Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 28 November 2026

C. Rehfeld

27 May 2026

APIX IoT Device Profile: Discovery and Presence for Connected Device Services
draft-rehfeld-apix-iot-01

Abstract

This document defines the APIX IoT Device Profile: an extension to the APIX Core Infrastructure specification that enables discovery and presence management for physical connected devices. It specifies the two-layer discoverability model (public device class, private device instance), the presence signal protocol, device instance token management, device class lifecycle, hub-mediated presence, ownership transfer, and the data retention and privacy rules applicable to device instance records. Autonomous agents that implement this profile can discover device capabilities at the class layer without any authentication, and retrieve live instance endpoint data at the instance layer subject to owner-granted authorisation.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 28 November 2026.

Table of Contents

1. Introduction

The APIX Core Infrastructure [APIX-CORE] defines the common foundation for agent-oriented service discovery: the APIX Manifest (APM) format, the three-dimensional trust model, commercial onboarding, sanctions compliance, and the base Index API. That document is the authoritative reference for all concepts and normative requirements shared across service types.

This document extends APIX Core for a fundamentally different service category: physical connected devices. API services have stable URLs and are verifiable by a crawler. IoT devices have dynamic network addresses, are physically present in private spaces, and signal their own liveness directly to the index. These differences require a specialised registration model, a presence protocol, and privacy rules that do not apply to web API services.

Implementors MUST satisfy all normative requirements in [APIX-CORE] before applying the additional requirements in this document.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. All JSON payloads defined in this document MUST use UTF-8 encoding as mandated by [RFC8259] Section 8.1.

1.1. Terminology

The following terms are defined in [APIX-CORE] and used here without redefinition: APIX, APM, APIX Manifest, service_id, trust model, organisation trust level (O-0 through O-5), service verification level (S-0 through S-4), liveness, commercial contract, sanctions screening, APIX Spider, consumer token, Connected Device.

Additional terms defined in this document:

Device Manufacturer — The organisation that produces a physical device type and registers its device class with APIX. A device manufacturer is a type of Service Owner as defined in [APIX-CORE]. The term "manufacturer" is used in this profile where the Service Owner is specifically a producer of physical hardware.

Device Class — An APM registration that describes a physical device type: its capabilities, manufacturer API versions, and presence model. Corresponds to spec.type: "device-class" or spec.type: "hub" in the APM. A device class is the public, Layer 1 record.

Device Instance — A record representing a single physical unit of a device class. Created and maintained by the device's presence signals. A device instance is the private, Layer 2 record.

Presence Signal — An authenticated message sent by a device (or its relay) to the APIX presence endpoint to establish or maintain a device instance record. Signal types: register, heartbeat, depart.

Device Instance Token — A per-device secret credential issued by APIX to the manufacturer at manufacture time. Used exclusively to authenticate presence signals. Distinct from consumer tokens and manufacturer API keys.

token_id — A stable, non-secret identifier assigned to a device instance token at issuance. Used in token management API calls without transmitting the token secret.

Presence Mode — The mechanism by which a device class's instances send presence signals to APIX. Values: push, cloud_relay, hub.

Hub — An internet-capable gateway device that relays presence signals for mesh-protocol devices that cannot reach APIX directly. Registered as spec.type: "hub".

Layer 1 — The public, unauthenticated view of a device class record. Contains capability metadata only. Never contains instance data.

Layer 2 — The private, authenticated view of device instance records. Accessible only to the device owner and explicitly delegated agents.

Reachable — An instance state flag. An instance is reachable: false when its current api_version is not present in the device class's current supported_api_versions list. Such instances are excluded from all query results.

Endpoint Confidence — An APIX-derived per-instance field that captures the reachability trust ceiling imposed by the device's network topology. Value "ipv6" indicates the device provided a valid global unicast IPv6 address in its most recent register signal and can be reached directly. Value "ipv4_observed" indicates no IPv6 address was provided; the device is reachable only via its manufacturer's cloud relay. APIX derives this field from the register signal and MUST NOT accept it from the device payload.

Law Enforcement Request (LER) — A formally submitted request from an authorised law enforcement authority to suppress the visibility of one or more device instances via the Law Enforcement Cooperation Interface defined in Section 9.8 of this document. LER processing obligations and the non-surveillance framework within which they operate are outlined in [APIX-CORE] Section 5.

1.2. Device Service Model

The IoT device trust model is architecturally distinct from the API service trust model defined in [APIX-SERVICES]. API service trust is pull-based: the APIX Spider probes the service. Device trust is push-based: devices signal their presence to the index. The manufacturer registration process, not the Spider, governs device class verification. The trust object structure in device class and device instance APMs differs accordingly from that of API service APMs. See [APIX-CORE] Section 6 (Trust Model) for the authoritative description of all three trust implementations.

APIX supports two fundamentally distinct service types: API services and Device services. The APM spec.type field determines which model applies. Services with spec.type set to device-class are governed by this section.

The two service types

API services have a stable URL, a machine-readable specification document, and are verified by the APIX Spider. Device services are physical or firmware-based entities with dynamic network addresses. They cannot be crawled. Their presence is inherently transient.

Two-layer discoverability model

Device services introduce a privacy and security requirement that does not apply to API services: individual device instances MUST NOT be publicly discoverable. A dishwasher's online presence is a household occupancy signal. An agent that can query which specific devices are online at a given time can infer whether a home is occupied. This is an unacceptable security and privacy exposure for both manufacturers and end users.

APIX enforces a strict two-layer model for device services:

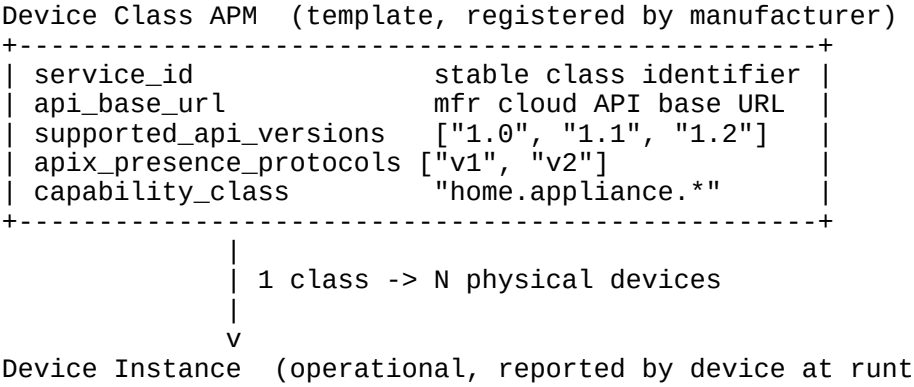
Layer 1 -- Device Class (public): The device class registration is publicly discoverable by any agent, including anonymous queries. It describes what the device type is, what it can do, and which protocol it uses. No instance data is present at this layer. This layer is safe to expose publicly and serves as the capability advertisement manufacturers want agents to find.

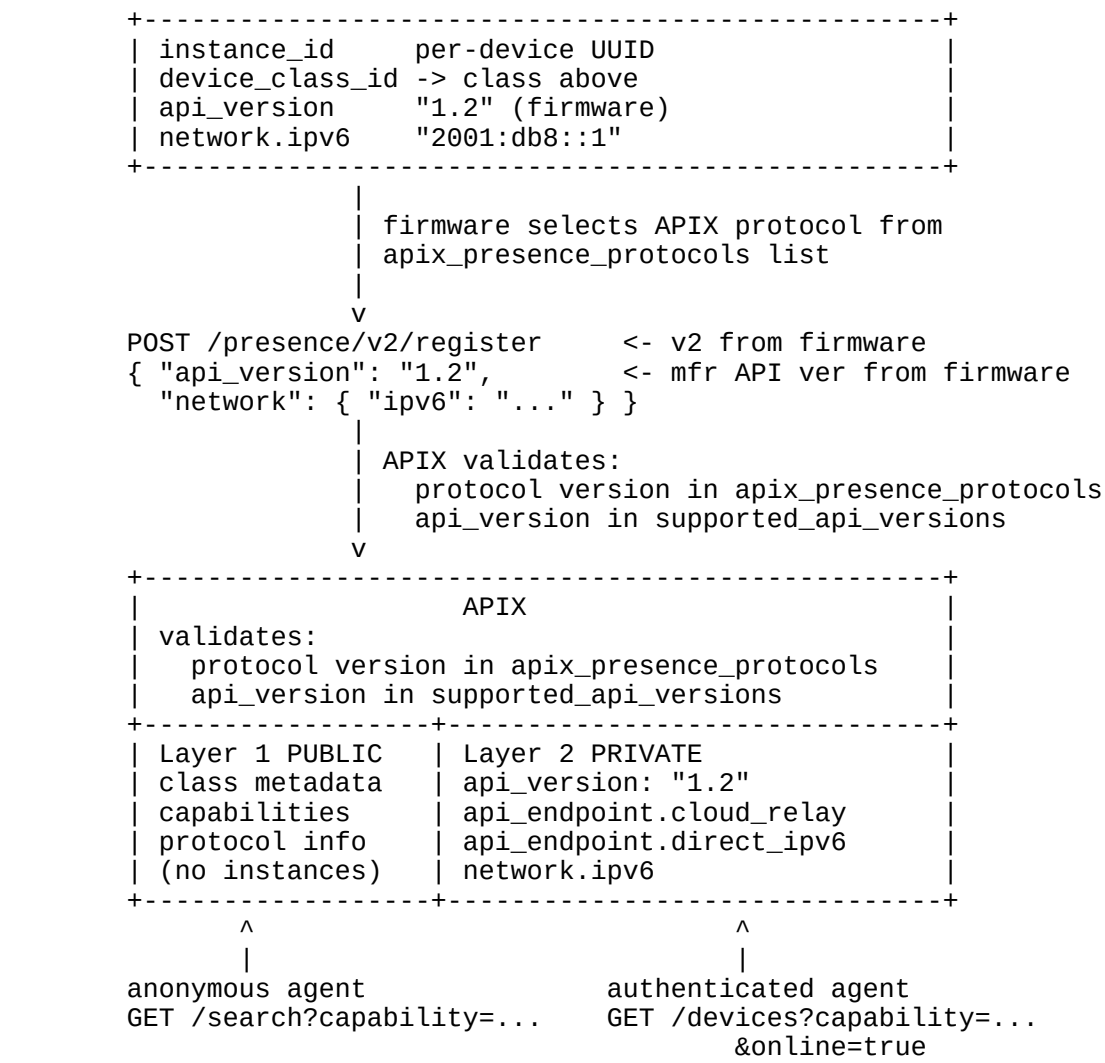
Layer 2 -- Device Instance (private): Individual device instance data -- whether a specific device is online, its current network endpoint, its operational state -- is NEVER returned to anonymous or unauthenticated queries. Instance-level data is accessible only to agents that have been explicitly authorised by the device owner, and only for instances owned by that principal.

```
Anonymous agent query: home.appliance.dishwasher
Result: device class record -- capabilities, protocol, manufacturer, APM
        NO instance data. NO online/offline. NO location. NO count.

Authenticated agent (device owner):
Result: instances owned by principal -- online status, endpoint, state
```

The APIX index MUST enforce this separation at the query layer. An index implementation that returns instance-level data to unauthenticated queries is non-conformant.





Manufacturer registration

A device manufacturer registers a device class once. The registration defines the capability profile, the communication protocol, and the presence model for all physical instances of that class. Individual device instances are not registered by end users -- they are provisioned by the manufacturer at the point of manufacture.

```
{
  "apm_version": "1.0",
  "service_id": "UUID v4 -- stable identifier for this device class",
  "name": "Haustec Pro 8 Dishwasher",
  "description": "Connected dishwasher -- energy optimisation",
  "lifecycle_stage": "stable",
  "owner": {
    "organisation_name": "BSH Home Appliances GmbH",
    "jurisdiction": "DE",
    "registration_number": "HRB 123456",
    "contacts": {
      "operations": "apix-ops@example.com",
      "escalation": "apix-escalation@example.com"
    }
  },
  "spec": {
    "type": "device-class",
    "protocol": "home-connect",
    "api_base_url": "https://api.haustec.example/api",
    "supported_api_versions": ["1.0", "1.1", "1.2"],
    "capability_class": "home.appliance.dishwasher",
    "presence_mode": "push",
    "apix_presence_protocols": ["v1"],
    "heartbeat_interval_seconds": 300,
    "max_offline_seconds": 900
  },
  "capabilities": ["home.appliance.dishwasher", "home.energy"],
  "pricing": {
    "model": "freemium",
    "pricing_url": "https://developer.haustec.example/pricing"
  },
  "custom": [
    "com.haustec.energy_class",
    "com.haustec.max_water_temp_c"
  ],
  "trust": {
    "organisation_level": "O-4",
    "service_level": "S-1",
    "spec_consistency": null,
    "liveness": {
      "presence_mode": "push",
      "heartbeat_interval_seconds": 300,
      "max_offline_seconds": 900
    }
  }
}
```

Note that the public device class record contains no instance count, no online status aggregate, and no presence timestamps. All such data is confined to the authenticated instance layer.

Device class APMs MAY include a pricing field using the schema defined in [APIX-SERVICES]. For IoT device classes, the "dynamic" pricing model is particularly relevant: a cloud relay service may price per connected device-hour at rates that vary with platform load. When model is "dynamic", the pricing_endpoint MUST be queried by the agent before initiating a command session, not on every individual command within an established session.

Device class APMs MAY include a custom field using the schema and normative rules defined in [APIX-SERVICES]. The custom array is a capability declaration list: it signals the existence of hardware-specific properties (energy classification, operating temperature ranges, protocol variant details) that frequently precede standardisation. Values are not stored in the index; they are defined in the manufacturer’s own API documentation and retrieved directly from the device class service. The same rules apply as in [APIX-SERVICES]: entries MUST use reverse-domain notation, the array MUST NOT exceed 20 entries, each entry MUST NOT exceed 128 characters, and declared key names are indexed and exposed via the custom_key search parameter.

Heartbeat interval and offline tolerance contract

Two companion fields in the device class APM together define the liveness contract for all instances of the class. Both are set at class registration time by the manufacturer and are binding on all instances.

heartbeat_interval_seconds — the target rate at which conformant instances MUST send presence signals. This is the expected steady-state cadence. An instance that signals less frequently than this value is non-conformant.

max_offline_seconds — the maximum age of the most recently received heartbeat before the index MUST mark the instance offline. This value encodes the manufacturer’s knowledge of normal connectivity gaps for the device type and MUST be greater than or equal to heartbeat_interval_seconds. APIX MUST reject device class registrations where max_offline_seconds is less than heartbeat_interval_seconds.

Manufacturers MUST choose values appropriate to the device’s deployment context and connectivity model. A mains-powered, always-connected field device (for example, a traffic surveillance camera)

SHOULD declare a short heartbeat_interval_seconds (for example, 30 seconds) and a max_offline_seconds only modestly larger (for example, 90 seconds), to enable fast offline detection. A battery-powered or intermittently connected sensor (for example, a forest fire smoke detector in a remote area) SHOULD declare a long heartbeat_interval_seconds (for example, 3600 seconds) and a max_offline_seconds that accommodates known connectivity gaps in the deployment environment (for example, 86400 seconds for a device in an area with intermittent network coverage).

The APIX index uses the class-declared values, never any instance-reported value, to determine liveness.

Notification channels for device classes:

Device class APMs MAY include a notifications field to allow consuming agents and device owners to receive push notifications about class-level changes (spec updates, trust level changes, lifecycle transitions). The permitted channel types for device class registrations are restricted to webhook only. sse and websocket channel types defined in [APIX-SERVICES] MUST NOT be used in device class APMs.

This restriction exists because device class changes are sparse, infrequent events. A persistent open connection (SSE or WebSocket) held by thousands of agents waiting for a class manifest change that occurs at most a few times per year is architecturally wasteful. A webhook is the correct model: the service calls the agent when an event occurs; no connection is held between events.

Device instance liveness (online/offline transitions for individual devices) is served by the presence signal mechanism and by the devices.presence delegation scope (Section 6), not by notification channels.

Device version contract

The device class APM declares all manufacturer API versions the device class supports in supported_api_versions. This is the exhaustive version contract between the manufacturer and APIX. When a physical device sends a presence signal it MUST declare exactly one of these versions as its currently running version; APIX MUST reject presence signals that declare a version outside the registered list.

The apix_presence_protocols field declares the list of APIX Presence Protocol versions (Section 6) that devices of this class may use when sending presence signals. At initial registration the list typically contains a single version. When the manufacturer releases a firmware

update that moves devices to a newer APIX presence protocol, the manufacturer updates the device class registration to add the new version to the list. Both versions remain valid simultaneously: devices that have received the firmware update call the newer presence endpoint; devices not yet updated continue calling the older one. APIX MUST accept presence signals from any endpoint version present in apix_presence_protocols.

The APIX presence protocol version a device uses is implicit in the endpoint URL it calls (e.g. POST /presence/v1/register vs POST /presence/v2/register). It is not repeated in the signal body.

The device version model provides two independent guarantees:

Reliability axis (device to APIX): Every device firmware version knows exactly which APIX presence endpoint URL to call. This URL is stable for the full support window of that protocol version (Section 6). A device running a given firmware version will always be able to signal its presence to APIX without modification, as long as the protocol version it was built against remains in its support window. A firmware update that introduces a newer APIX protocol version is the device’s own upgrade path, not a requirement.

Flexibility axis (agent through APIX to device): The manufacturer API version (api_version) running on a physical device instance is reported by the device itself and may change when the device performs a firmware upgrade through the manufacturer’s own update process. A firmware upgrade that changes the running API version MUST be signalled to APIX by the device sending a new register signal carrying the updated api_version. APIX updates the instance record immediately. Consuming agents that re-fetch the instance record after a connection failure automatically receive the current API version and the correct versioned endpoint. The firmware update process is out of scope for this specification; APIX is version-aware only at the presence signal boundary.

Example: a device class initially registered with firmware D1.0 using APIX presence protocol v2 ships with apix_presence_protocols: ["v2"]. When a firmware update introduces D1.3 with support for APIX presence protocol v3, the manufacturer adds v3 to the device class registration: apix_presence_protocols: ["v2", "v3"]. Updated devices call /presence/v3/register; unupdated devices continue calling /presence/v2/register. Both are accepted.

Device instance tokens

Each physical device is provisioned at manufacture with a device instance token. This token is used exclusively to authenticate presence signals sent to the APIX presence endpoint. Tokens are issued by APIX to the manufacturer as part of the commercial contract and are scoped to a specific device class registration.

Token requirements: tokens MUST contain a minimum of 256 bits of cryptographic randomness and MUST be encoded as base64url without padding. Tokens MUST NOT be derivable from device serial numbers, MAC addresses, or any other publicly accessible or predictable identifier. APIX stores only the SHA-256 hash of each token; the plaintext is delivered once at issuance and cannot be re-retrieved.

Each token is assigned a stable, non-secret token_id by APIX at issuance. The token_id uniquely identifies a token in API operations (rotation, revocation) without requiring the token secret to be transmitted. The token_id is returned alongside the token in the batch delivery payload and appears in the device instance record, allowing the manufacturer to correlate their production records with APIX instance data.

Token rotation

A device instance token MUST be rotatable without changing the device’s instance_id. The rotation preserves the device’s ownership, history, and position in any consuming agent’s workflow; only the credential changes.

Rotation is initiated by the manufacturer via the Token Management API, referencing the token_id of the token to be replaced. APIX MUST issue a single replacement token and enter a dual-valid state for that instance: both the existing token and the replacement token are accepted for presence signals for a configurable handover window (default: 24 hours; manufacturer may request shorter or longer within operator limits).

The manufacturer delivers the replacement token to the physical device via the manufacturer’s OTA channel (out of scope for this specification). When the device sends its first presence signal using the replacement token, APIX MUST atomically revoke the previous token. Subsequent signals using the old token MUST be rejected with HTTP 401 token_revoked.

If the handover window expires before the replacement token has been used in a successful presence signal, APIX MUST revoke the old token at window expiry. The device transitions to offline and enters the HTTP 401 dormant state until the manufacturer resolves the delivery failure via a further OTA cycle or re-provisioning.

Replacement tokens MUST meet the same entropy and format requirements as initial tokens. Previously issued tokens MUST NOT be reissued after rotation or revocation.

Device owner authorisation

A device owner associates their device instances with their APIX identity through a one-time ownership claim process defined by the manufacturer. Once claimed, the device owner MAY delegate access to specific agents -- for example, a home energy management agent authorised to query the dishwasher's availability and schedule a wash cycle during off-peak tariff windows. Delegation is scoped, revocable, and does not expose the device to any other agent.

Ownership transfer and release

Device ownership is transferable without changing the device's instance_id. A device owner MAY release their ownership claim at any time by calling DELETE /devices/{instance_id}/claim. APIX MUST remove the owner_id association and revoke all agent delegations scoped to that device instance. The instance record and its presence history are retained. The device continues to signal presence and remains online, but is not visible in Layer 2 queries until a new owner completes a claim.

A manufacturer MAY issue a new claim token for a device that already has an active ownership claim -- for example, to facilitate a resale or to respond to a factory reset. When a new claim is completed, APIX MUST atomically replace the existing owner_id with the new claimant's identity and revoke all delegations from the previous owner. The previous owner receives no notification; their Layer 2 access ceases immediately upon the new claim being recorded.

A device that is factory-reset at firmware level SHOULD send a depart signal with "reason": "factory_reset" before clearing its operational state. On receipt, APIX MUST release the ownership association (clear owner_id) and mark the instance offline, in addition to the standard depart processing. If no factory-reset depart signal is received (e.g., the reset is hardware-triggered), the ownership association persists until the manufacturer issues a new claim token or the previous owner explicitly releases their claim.

Presence signal mechanism

Device liveness is maintained by push signals, not Spider crawls. A presence signal is not a bare keepalive: it carries the device's currently running manufacturer API version and its network addresses, allowing APIX to construct the full instance record at registration time.

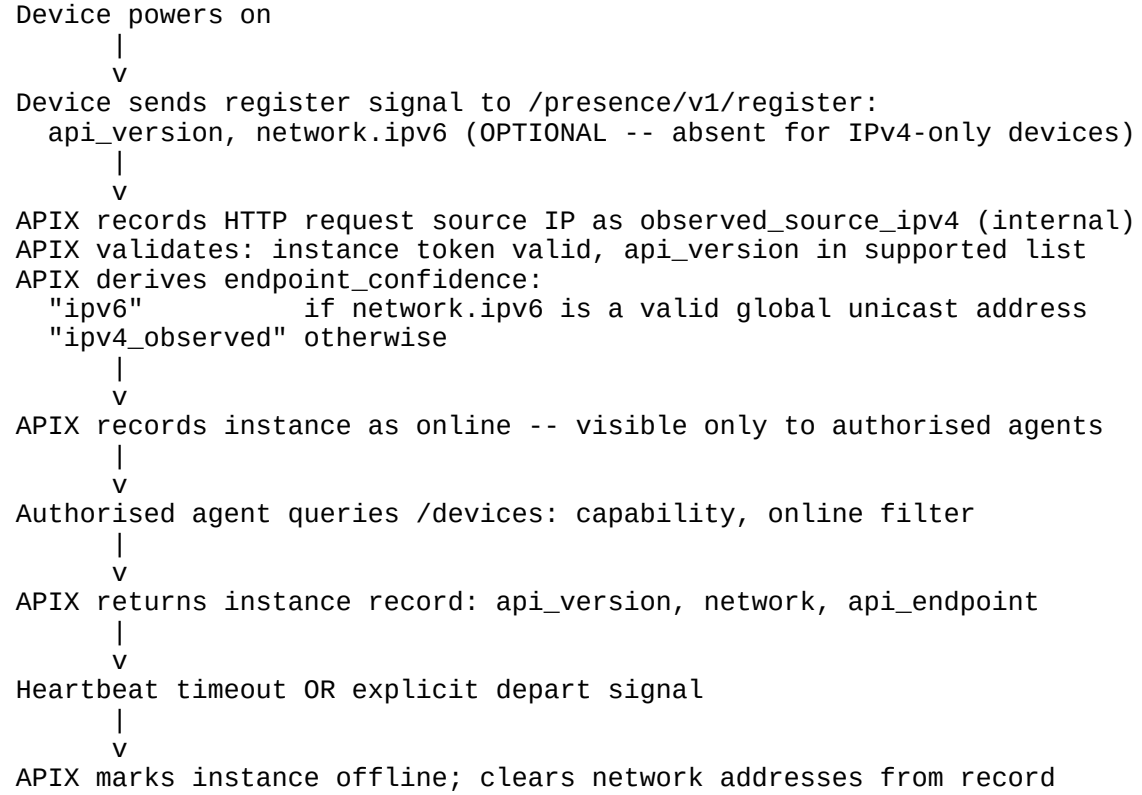
Signal types: register (first contact after power-on or reconnect), heartbeat (periodic keepalive, MUST NOT change api_version or network addresses without re-registering), depart (graceful shutdown or factory reset). The depart signal MAY carry an optional "reason" field; the only defined value in this specification is "factory_reset", which triggers ownership release in addition to standard offline processing. Unrecognised reason values MUST be ignored by APIX.

```
POST https://apix.example.org/presence/v1/register
Authorization: Bearer device-instance-token
Content-Type: application/json

{
  "device_class_id": "dc-haustec-pro8-dishwasher",
  "signal_type": "register",
  "api_version": "1.2",
  "network": {
    "ipv6": "2001:db8:85a3::8a2e:370:7334"
  }
}
```

When a register signal carries an api_version not present in the device class's supported_api_versions list, APIX MUST return HTTP 422 api_version_not_supported and MUST create or update the instance record with reachable: false. The record is not visible in capability searches but IS visible in the device owner's Layer 2 query, allowing the owner to observe and diagnose the version mismatch without direct device access. APIX MUST reject a presence signal received at a protocol endpoint version not listed in the device class's apix_presence_protocols with HTTP 400; no instance record is created for endpoint version mismatches.

Presence signal flow:



The index determines instance liveness by comparing the age of the most recently received heartbeat against the max_offline_seconds value declared in the device class APM:

online IF: (current_time - last_heartbeat_at) <= max_offline_seconds

The threshold is always derived from the class template, never from any instance-reported value. An instance whose condition evaluates to false MUST be marked offline by the index.

Manufacturers SHOULD implement a departure signal on graceful shutdown; relying solely on heartbeat timeout is permitted but results in a delayed offline transition.

Presence relay options

Three presence_mode values are defined for device classes:

push — the device contacts the APIX presence endpoint directly over HTTPS without any intermediary. This is the highest-integrity path: the signal originates from the device itself. Device classes using push may achieve any service trust level up to S-4.

cloud_relay — the manufacturer's cloud infrastructure relays presence signals on behalf of the device. The device sends signals to a manufacturer-operated endpoint; the manufacturer's cloud forwards them to APIX using the device's instance token. This model is appropriate for devices on constrained networks or behind carrier-grade NAT that cannot reach APIX directly. Device classes using cloud_relay are bounded at service trust level S-1. The manufacturer's cloud intermediary holds all instance tokens for devices of that class, and can attest any device state value (online status, api_version, network addresses) without independent verification by APIX. A consuming agent applying a trust policy that requires S-2 or higher MUST NOT rely on cloud_relay device instances.

hub — a hub gateway device mediates presence for devices that communicate exclusively over short-range mesh protocols, as specified in Section 3.6. Service trust level is bounded at S-1 for the same reasons as cloud_relay.

The presence_mode MUST be declared in the device class APM. The presence_mode determines the trust ceiling for all instances of that class; it cannot be overridden per-instance.

Endpoint confidence

presence_mode captures signal-integrity trust — how reliably APIX can attribute a presence signal to the physical device. A separate per-instance field, endpoint_confidence, captures reachability trust — how reliably a consuming agent can form a direct connection to the device.

APIX MUST derive endpoint_confidence from each register signal:

Value	Condition	Direct endpoint available
ipv6	Device provided a valid global unicast IPv6 address in network.ipv6	api_endpoint.direct_ipv6 is present
ipv4_observed	No valid global unicast IPv6 provided; APIX observed a source IPv4 from the HTTP request	No direct_ipv6; no direct_ipv4; cloud_relay or hub_gateway only

Table 1

IPv4 addresses self-reported by a device MUST NOT be accepted in the network payload. APIX records the HTTP request source address as observed_source_ipv4 for internal operational use (abuse detection, geo-routing) and MUST NOT surface it in any instance record or API response. The reason is structural: a device behind NAT knows only its private LAN address; the public IP APIX observes is the NAT gateway, not the device. Neither value reliably supports direct agent-to-device connectivity without port forwarding, which cannot be assumed.

A consuming agent MUST read endpoint_confidence before selecting an endpoint. Agents performing consequential operations (device commands, ownership actions) MUST NOT assume a direct endpoint is available for instances with endpoint_confidence: "ipv4_observed"; they MUST route through cloud_relay or hub_gateway.

APIX Presence Endpoint Versioning

The APIX Presence Protocol is versioned independently of the Index API. The version identifier is embedded in the endpoint path:

https://apix.example.org/presence/v1/register
https://apix.example.org/presence/v1/heartbeat
https://apix.example.org/presence/v1/depart

A device class declares all presence protocol versions its devices may use in apix_presence_protocols. The list starts with the version supported at initial registration. When a firmware update introduces

support for a newer protocol version, the manufacturer adds it to the list -- the older version remains valid for devices not yet updated. Multiple versions in apix_presence_protocols are all simultaneously active for that device class.

Long-term support guarantee: APIX MUST maintain each presence protocol version in full operation for a minimum of ten (10) years from the date on which the first device class registration including that version is accepted. After this window, a five (5) year deprecation period applies during which the version continues to operate but no new device class registrations may add it to their apix_presence_protocols list. Removal of a presence protocol version before the combined fifteen-year window has elapsed is a specification violation.

This guarantee exists because IoT appliances frequently have commercial service lifespans of ten to twenty years, and not all deployed units receive firmware updates on a predictable schedule. A manufacturer must be able to guarantee to their customers and channel partners that presence signalling will remain operational for the full product life, independently of which firmware version a specific unit happens to run.

Protocol version sunset

APIX presence protocol versions do not remain operational indefinitely. After the minimum support window (Section 3.5), APIX may sunset a version. Sunset proceeds in two phases:

Deprecation phase: APIX marks the version as deprecated. All responses from the deprecated endpoint version MUST include the Deprecation response header ([RFC9745]) and the Sunset response header ([RFC8594]) carrying the planned removal date. APIX MUST notify the registered operations contact of every device class that lists the deprecated version in apix_presence_protocols. Device class records that still include the deprecated version receive a standard_warnings entry indicating the sunset date. The deprecation phase MUST last a minimum of five (5) years.

Removal: On the sunset date, the endpoint version is removed. Requests to the removed endpoint MUST return HTTP 410 Gone with a response body referencing current migration documentation. Device instances that have not migrated to a supported protocol version will cease to appear in APIX as online. Their device class registrations remain valid; only presence signalling is affected.

The sunset obligation runs between APIX and the device manufacturer. The manufacturer accepted the APIX terms of service at registration and is contractually responsible for managing their fleet's migration path. APIX is not liable to individual device owners whose devices lose APIX connectivity because the manufacturer did not deploy a firmware update within the deprecation window. The manufacturer may satisfy this obligation by updating device firmware to a supported protocol version, by routing presence signals through a manufacturer-side relay that performs protocol translation, or by retiring the device class and communicating end-of-APIX-support to customers.

IPv6 as primary network addressing

The APIX Presence Protocol uses IPv6 as the sole self-reported network address in device presence signals. Physical devices SHOULD include a global unicast IPv6 address in the network.ipv6 field of their presence signals. Devices that cannot provide a global unicast IPv6 address send presence signals without a network field; APIX observes their source IPv4 address from the HTTP request for internal operational use only (see endpoint_confidence semantics above). IPv4 MUST NOT be included in the network payload of a presence signal.

IPv6 addresses MUST be provided as unbracketed strings in the network.ipv6 field of the presence signal. APIX stores and returns them as unbracketed strings; consuming agents construct RFC 3986-compliant bracketed URIs (e.g. https://[2001:db8::1]/api/1.2/) when building endpoint URLs from instance records.

Spider exclusion

Device services with spec.type: device-class MUST NOT be crawled by the APIX Spider. The Spider specification in [APIX-CORE] does not apply to device services. Spec consistency checking (spec_consistency) is not applicable and MUST be set to null for device class registrations.

Trust levels for device services

Organisation trust levels (O-0 through O-5) apply to device manufacturers in the same way as API service owners. Service verification levels S-0 through S-4 (as defined in [APIX-CORE]) apply in full, with the following device-class-specific meaning for S-2: S-2 (Spec Verified) means the declared capability class is registered in the APIX capability taxonomy and the device class has completed the manufacturer registration process including KYC/AML screening. The Spider does not contribute to device service verification.

The maximum S-level achievable by a device class is constrained by its presence_mode. The full S-0 through S-4 range is defined; the table below states the ceiling imposed by each mode:

Presence mode	Maximum service trust level	Reason
push	S-4	Direct device signal path; no intermediary
cloud_relay	S-1	Manufacturer cloud attests device state; not independently verifiable
hub	S-1	Hub gateway attests device state; not independently verifiable

Table 2

A consuming agent that requires S-2 or higher for its trust policy MUST exclude device classes with presence_mode: cloud_relay or presence_mode: hub from its query results. Agents SHOULD filter by service_level_min when querying the Device Instance Query API.

1.2.1. Hub-Mediated Device Services

Many IoT devices communicate exclusively over short-range mesh protocols (Matter over Thread, Zigbee, Z-Wave) and have no direct path to the public internet. These devices are physically incapable of sending presence signals to APIX themselves. A hub gateway -- an internet-capable device that bridges the local mesh to the IP network -- must act as their APIX presence relay.

APIX supports this through two complementary registrations: a hub device class (type hub) for the gateway hardware, and a hub-mediated device class (presence_mode hub) for the devices the hub serves.

Hub entity

A hub is registered in APIX as a device class with spec.type: "hub". Hub physical units receive their own instance tokens at manufacture, identical in structure to device instance tokens. Hubs signal their own presence directly to the APIX presence endpoint (presence_mode push). A hub instance that is offline cannot relay presence for any of its connected devices.

```
{
  "apm_version": "1.0",
  "service_id": "UUID v4 -- stable hub class identifier",
  "name": "Haustec Connect Bridge v2",
  "description": "Matter/Thread hub for BSH home appliances",
  "lifecycle_stage": "stable",
  "owner": {
    "organisation_name": "BSH Home Appliances GmbH",
    "jurisdiction": "DE",
    "registration_number": "HRB 123456",
    "contacts": {
      "operations": "apix-ops@example.com",
      "escalation": "apix-escalation@example.com"
    }
  },
  "spec": {
    "type": "hub",
    "hub_protocols": ["matter"],
    "supported_device_classes": ["dc-haustec-pro8-dishwasher"],
    "presence_mode": "push",
    "apix_presence_protocols": ["v1"],
    "heartbeat_interval_seconds": 60,
    "max_offline_seconds": 180
  }
}
```

supported_device_classes is the exhaustive list of device class IDs for which this hub type may relay presence signals. APIX MUST reject hub-relay signals for device classes not in this list.

Hub-mediated device class

A device class whose physical instances cannot signal directly sets presence_mode: "hub" in its APM. It also declares hub_protocols to identify how the hub-to-device local channel carries the instance token during the pairing process.

```
{
  "spec": {
    "type": "device-class",
    "presence_mode": "hub",
    "hub_protocols": ["matter"],
    "permitted_hub_classes": ["hc-haustec-connect-bridge-v2"],
    "api_base_url": "https://api.haustec.example/hub-gateway/api",
    "supported_api_versions": ["1.0", "1.1"],
    "apix_presence_protocols": ["v1"],
    "heartbeat_interval_seconds": 300,
    "max_offline_seconds": 600
  }
}
```

hub_protocols informs APIX and consuming agents which local protocol carries the device's instance token to the hub at pairing time. The mechanism for token transfer is hub-protocol specific and outside the scope of this specification.

permitted_hub_classes is an optional field. When present, it is a manufacturer capability declaration that restricts which hub classes may relay presence signals for instances of this device class. APIX MUST reject relay signals from hub instances whose class ID is not in the list. When absent, any hub whose supported_device_classes includes this device class may relay.

This field is a capability and interoperability declaration, not a per-instance pairing or authentication mechanism. It encodes the manufacturer's knowledge of which hub types their device is designed to work with correctly. Instance-level device-to-hub pairing is governed by the local network provisioning process (for example, Matter commissioning) and the manufacturer's device management infrastructure; it is outside the scope of this specification.

Hub-relay presence signal

A hub signals presence for a connected device using a dedicated hub-relay endpoint. Each signal carries two credentials: the hub's own instance token in the Authorization header (authenticating the hub), and the paired device's instance token in the request body (authenticating the specific device). APIX validates both independently.

```
POST https://apix.example.org/presence/v1/hub-relay/register
Authorization: Bearer hub-instance-token
Content-Type: application/json

{
  "hub_instance_id": "di-hub-a4c27f...",
  "signal_type": "register",
  "device": {
    "instance_token": "device-instance-token",
    "device_class_id": "dc-haustec-pro8-dishwasher",
    "api_version": "1.0"
  },
  "hub_gateway_endpoint":
    "https://api.haustec.example/hub-gw/api/1.0/di-f47ac1..."
}
```

Note that hub-mediated device presence signals carry no network.ipv6 field: the device has only a local mesh address, which is not internet-routable and MUST NOT be included. The agent reaches the device exclusively via the hub_gateway_endpoint at the manufacturer's cloud, which routes requests to the device through the hub.

APIX MUST reject a hub-relay signal where: (a) the hub instance token is invalid or revoked, (b) the device instance token is invalid or revoked, (c) the device class is not in the hub class's supported_device_classes, (d) the hub instance is not itself currently online, or (e) the device class declares permitted_hub_classes and the relaying hub's class ID is not in that list.

Layer 2 instance record for hub-mediated devices

```
{
  "instance_id": "di-f47ac10b-58cc-4372-a567-0e02b2c3d479",
  "device_class_id": "dc-haustec-pro8-dishwasher",
  "api_version": "1.0",
  "online": true,
  "presence_mode": "hub",
  "hub_instance_id": "di-hub-a4c27f...",
  "last_seen_at": "2026-04-24T09:00:00Z",
  "api_endpoint": {
    "hub_gateway":
      "https://api.haustec.example/hub-gw/api/1.0/di-f47ac1..."
  },
  "owner_id": "usr-7e8a9b2c...",
  "claimed_at": "2026-01-10T10:00:00Z",
  "_links": {
    "self": {
      "href": "https://apix.example.org/devices/di-f47ac1..."
    },
    "hub": {
      "href": "https://apix.example.org/devices/di-hub-a4c27f..."
    },
    "device_class": {
      "href": "https://apix.example.org/device-classes/dc-haustec-p8"
    }
  }
}
```

Cascading offline

When a hub instance is marked offline -- whether by heartbeat timeout or explicit departure signal -- APIX MUST mark all device instances currently attributed to that hub as offline and MUST clear their api_endpoint.hub_gateway fields. APIX MUST NOT return online: true for any device instance attributed to an offline hub.

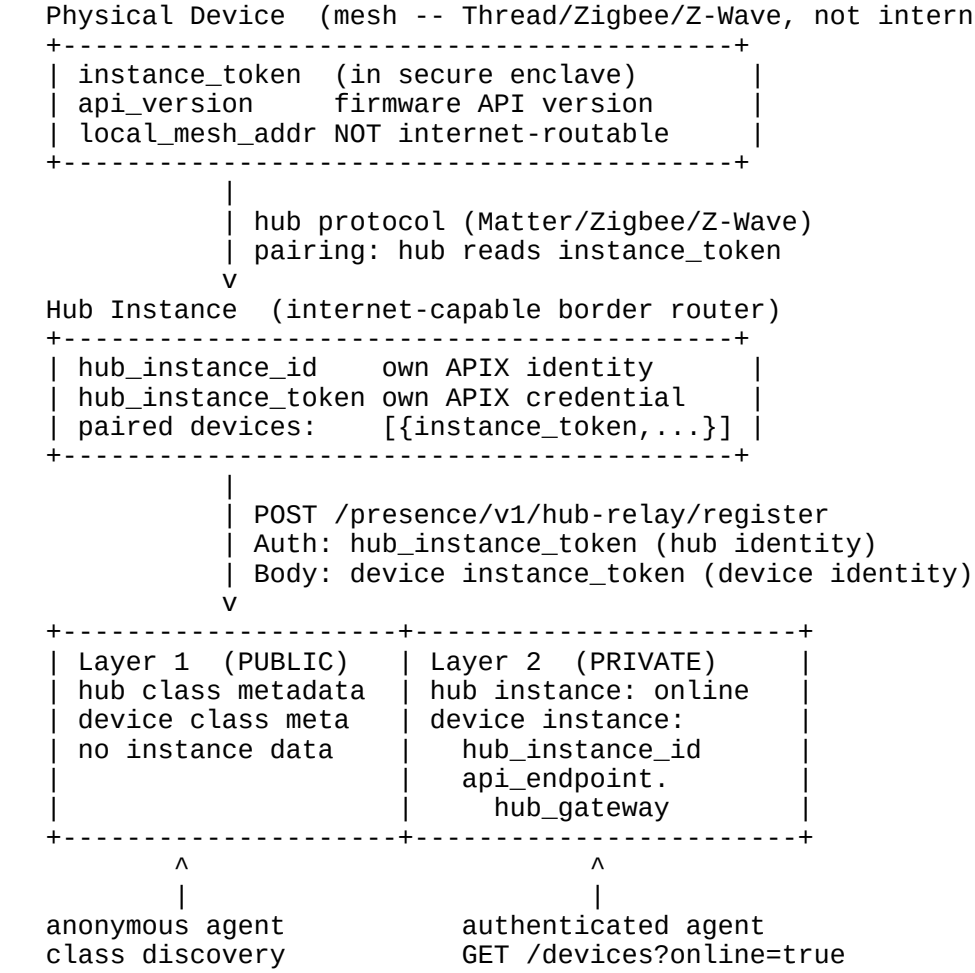
Hub migration

A device may move from one hub to another. When a hub-relay register signal is received for a device whose instance record is currently attributed to a different hub, APIX MUST update the device instance record: hub_instance_id is set to the new hub's instance_id and api_endpoint.hub_gateway is replaced with the value from the new signal. The device instance_id MUST be preserved. APIX MUST return HTTP 409 device_not_attributed_to_hub on any subsequent hub-relay signal from the previous hub for that device.

Trust boundary

Hub relay introduces an intermediary trust boundary absent from the push model. The device's instance token validates that the specific device was genuinely provisioned for this class and class owner, but the hub's reported api_version and hub_gateway_endpoint are attested by the hub and cannot be independently verified by APIX. A compromised hub can misrepresent the state of devices it genuinely controls but cannot forge presence signals for devices whose instance tokens it does not hold.

Device classes using presence_mode: "hub" are bounded at service trust level S-1. S-2 (Spec Verified) requires a direct device signal path and is not achievable via hub relay in this specification version.



1.2.2. Device Class Lifecycle

Device class lifecycle is defined by this profile. Valid lifecycle_stage values for device classes are stable, deprecated, and end_of_life. These are the complete set for this service type.

A device class progresses through three lifecycle stages that the manufacturer sets via registration updates. Transitions are unidirectional: stable → deprecated → end_of_life.

stable — Full operation. All normative rules apply without modification.

deprecated — The manufacturer is winding down the class. APIX MUST continue accepting presence signals from existing device instances. Deprecated classes MUST be excluded from Layer 1 capability searches by default; they remain directly retrievable by service_id. Device instances of deprecated classes MUST be excluded from Layer 2 query results by default; a consuming agent MAY include them by adding include_deprecated=true to the query. Instance records MUST carry "class_lifecycle_stage": "deprecated".

_end_of_life_ — APIX MUST reject all presence signals for the class with HTTP 410 Gone. All existing instances are marked online: false and reachable: false. Class and instance records are retained for historical query by the device owner but MUST NOT appear in any capability or instance search.

1.2.3. Instance Reachability

A device instance MUST be marked "reachable": false when its current api_version is absent from the device class’s current supported_api_versions list. This condition arises when a manufacturer removes a version from the supported list -- for example, as part of a class deprecation cycle -- while one or more physical devices remain on that version. Such a device is still online (liveness signals continue to be accepted) but its API version is no longer endorsed, so no valid endpoint can be constructed by a consuming agent.

APIX MUST exclude instances with reachable: false from query results regardless of the online filter value. A query for online=true devices MUST NOT include an instance that is reachable: false.

When reachable: false, instance records MUST omit api_endpoint fields. The api_version field is retained to allow the owner to identify the version mismatch.

Instance records MUST include "reachable": false and "class_lifecycle_stage" (with its non-stable value) whenever those conditions apply. Neither field is present in the response when the instance is reachable and its class is stable.

1.3. Data Retention and Privacy

Personal data in device instance records

Device instance records contain personal data fields. The index operator is the data controller for these fields within the meaning of applicable data protection law (including GDPR where applicable). The following fields are classified as personal data:

* owner_id — links the instance to a natural person's consumer account

* network.ipv6 — the device's IPv6 address, which may identify a natural person's home network or device location. Note: APIX observes the device's source IPv4 address internally at registration for abuse detection and geo-routing purposes; this address is not stored in the instance record and is not classified as instance-level personal data.

* api_endpoint fields — URLs that may carry device-specific identifiers attributable to a natural person

* claimed_at and last_seen_at — timestamps that, in combination with network addresses, constitute behavioural data

instance_id and device_class_id are pseudonymous technical identifiers and are not classified as personal data on their own.

Network addresses and api_endpoint fields MUST be cleared from the instance record when a device transitions to offline, as specified in the presence signal processing rules. This is both an operational requirement and a data minimisation obligation.

Offline record retention

An instance record that has been offline without re-registration for more than ninety (90) days MUST be transitioned to archived state. In archived state:

* owner_id, claimed_at, last_seen_at, and went_offline_at are cleared.

* instance_id, device_class_id, and api_version are retained for operational audit purposes (legitimate interest).

* The record MUST NOT appear in any Layer 2 query result.

An archived record that has not been re-activated by a new registration within twelve (12) months of archival MUST be permanently deleted.

Right to erasure

When a device owner submits an erasure request for a device instance (or when the owner deletes their consumer account), APIX MUST:

1. Clear owner_id, claimed_at, and all network address and endpoint fields from the instance record.

2. Revoke all agent delegations scoped to the device instance.

3. Retain instance_id, device_class_id, api_version, and presence activity timestamps for the minimum period required by the operator’s legitimate interest (audit trail, dispute resolution). RECOMMENDED retention: 90 days from the erasure request date, after which the record transitions to the standard archived state.

The device’s instance token is not affected by an erasure request. The token is a device credential, not owner personal data. The device MAY continue to send presence signals; however, without an owner_id association the instance record will not appear in any Layer 2 query until re-claimed.

An erasure request does not affect device class records, which are owned by the manufacturer and do not contain end-user personal data.

1.4. Device Instance Query API

Device instance queries give an authorised agent access to Layer 2 -- the private, per-principal view of device instances. All endpoints in this section require authentication. The consumer token MUST carry the devices.read scope. APIX enforces ownership boundaries at query time: a principal MAY only retrieve instance records for devices they own or have been explicitly delegated access to. Queries that would match instances belonging to other principals MUST return an empty result, not an error, to prevent ownership enumeration.

The devices link in the Index root resource (defined in [APIX-CORE]) provides the templated entry point to this API. Agents MUST follow this link rather than constructing the URL independently.

Query endpoint:

```
GET /devices?capability=home.appliance.dishwasher
            &online=true
            &api_version=1.2
            &page=1
            &page_size=20
Authorization: Bearer consumer-token-with-devices.read-scope
```

Normative query parameters:

Parameter	Type	Default	Description
capability	string	—	Capability taxonomy term. Filters to instances of device classes declaring this capability
online	boolean	—	true: online instances only. false: offline only. Absent: all instances
api_version	string	—	Filter by manufacturer API version currently running on the instance
include_deprecated	boolean	false	When true, results include instances of deprecated classes. Instances with reachable: false are excluded regardless of this parameter
page	integer	1	Result page number
page_size	integer	20	Results per page. Maximum: 100

Table 3

Device Instance Summary (Level 1):

```
{
  "instance_id": "di-f47ac10b-58cc-4372-a567-0e02b2c3d479",
  "device_class_id": "dc-haustec-pro8-dishwasher",
  "device_class_name": "Haustec Pro 8 Dishwasher",
  "api_version": "1.2",
  "online": true,
  "last_seen_at": "2026-04-24T08:01:00Z",
  "_links": {
    "self": {
      "href": "https://apix.example.org/devices/di-f47ac1..."
    },
    "device_class": {
      "href": "https://apix.example.org/device-classes/dc-haustec-p8"
    }
  }
}
```

The fields reachable and class_lifecycle_stage are present only when their value is non-default (reachable: false; class_lifecycle_stage other than stable). Network addresses and resolved endpoint options are not returned at Level 1. Agents retrieve them by fetching the detail resource via the self link.

Device Instance Record (Level 2):

```
{
  "instance_id": "di-f47ac10b-58cc-4372-a567-0e02b2c3d479",
  "device_class_id": "dc-haustec-pro8-dishwasher",
  "device_class_name": "Haustec Pro 8 Dishwasher",
  "api_version": "1.2",
  "online": true,
  "endpoint_confidence": "ipv6",
  "last_seen_at": "2026-04-24T08:01:00Z",
  "network": {
    "ipv6": "2001:db8:85a3::8a2e:370:7334"
  },
  "api_endpoint": {
    "cloud_relay": "https://api.haustec.example/api/1.2/di-f47ac1",
    "direct_ipv6": "https://[2001:db8:85a3::8a2e:370:7334]/api/1.2/"
  },
  "owner_id": "usr-7e8a9b2c-1234-5678-90ab-cdef01234567",
  "claimed_at": "2026-01-10T10:00:00Z",
  "_links": {
    "self": {
      "href": "https://apix.example.org/devices/di-f47ac1..."
    },
    "device_class": {
      "href": "https://apix.example.org/device-classes/dc-haustec-p8"
    }
  }
}
```

The endpoint_confidence field governs which api_endpoint keys are present in the record. When endpoint_confidence is "ipv6", the device provided a valid global unicast IPv6 address in its most recent register signal; both api_endpoint.cloud_relay and api_endpoint.direct_ipv6 are present. direct_ipv6 is the preferred endpoint for agents on IPv6-capable networks as it avoids the manufacturer’s relay infrastructure.

When endpoint_confidence is "ipv4_observed", no IPv6 address was provided; only api_endpoint.cloud_relay is present. APIX observes the device’s source IPv4 address internally for abuse detection and geo-routing purposes but MUST NOT surface it in instance records: the observed IPv4 is a NAT gateway address and is not a usable direct endpoint.

api_endpoint.cloud_relay is always present when the manufacturer’s device class APM declares api_base_url; it is the mandatory fallback for all endpoint_confidence values.

Device Instance Record (Level 2 — endpoint_confidence: "ipv4_observed"):

```
{
  "instance_id": "di-a1b2c3d4-0001-0002-0003-000400050006",
  "device_class_id": "dc-haustec-pro8-dishwasher",
  "device_class_name": "Haustec Pro 8 Dishwasher",
  "api_version": "1.1",
  "online": true,
  "endpoint_confidence": "ipv4_observed",
  "last_seen_at": "2026-04-24T09:15:00Z",
  "api_endpoint": {
    "cloud_relay": "https://api.haustec.example/api/1.1/di-a1b2c3d4"
  },
  "owner_id": "usr-1a2b3c4d-5678-90ab-cdef-012345678901",
  "claimed_at": "2025-11-03T14:30:00Z",
  "_links": {
    "self": {
      "href": "https://apix.example.org/devices/di-a1b2c3d4..."
    },
    "device_class": {
      "href": "https://apix.example.org/device-classes/dc-haustec-p8"
    }
  }
}
```

No network object is present: the device did not report an IPv6 address and APIX does not surface the observed NAT gateway IPv4. Agents MUST route all control traffic through api_endpoint.cloud_relay for such instances.

When a device transitions to offline, network addresses and api_endpoint values are cleared from the instance record. An agent that cached endpoint data from a previous online period MUST re-fetch the instance record after a connection failure to obtain fresh network addresses, as dynamic IP assignments may have changed.

1.5. Manufacturer Fleet Summary API

Manufacturers require aggregate visibility into the deployment status of their device classes — for version migration planning, token rotation batch operations, and protocol deprecation compliance. The Device Instance Query API is scoped to device owners and their delegated agents; it does not serve the manufacturer’s operational needs.

The Manufacturer Fleet Summary API exposes aggregate, per-class statistics to the registering manufacturer. It MUST NOT return any per-instance data: no instance_id values, no network addresses, no owner_id associations, and no individual presence timestamps. Returning such data would constitute a Layer 2 enumeration violation (Section 8.2).

Authentication: Requests MUST be authenticated using the manufacturer's APIX API key, which is the same credential used for device class registration updates and token management operations (Section 4.2). APIX MUST reject requests whose API key does not match the registering organisation of the requested device class.

Endpoint:

```
GET https://apix.example.org/device-classes/{class_id}/fleet-summary
Authorization: APIX-Key {manufacturer-api-key}
```

Response:

```
{
  "class_id": "dc-haustec-pro8-dishwasher",
  "class_lifecycle_stage": "stable",
  "total_registered": 47832,
  "online_count": 31209,
  "unclaimed_count": 412,
  "api_version_distribution": {
    "1.0": 8201,
    "1.1": 15447,
    "1.2": 24184
  },
  "as_of": "2026-04-25T12:00:00Z"
}
```

Field	Description
total_registered	Count of all instances of this class currently in the index, regardless of online status or ownership
online_count	Instances that have sent a presence signal within the configured heartbeat timeout window
unclaimed_count	Instances present in the index that have no owner_id association
api_version_distribution	Map of manufacturer API version string to count of instances currently reporting that version. Values are sourced exclusively from device presence signals
as_of	Timestamp of the aggregate computation. MAY be up to 15 minutes stale

Table 4

The online_count field is a scalar count only; it does not identify which instances are online. An implementation MUST ensure that no derivation path from fleet summary data can identify the presence or absence of any individual device instance.

1.6. Agent Delegation

1.6.1. Delegation Model

A device owner MAY delegate access to specific device instances to an autonomous agent. Delegation is explicit, scoped, time-bounded, and revocable. It does not expose the instance to any other agent and does not modify the instance record visible to other principals.

Delegation is distinct from consumer token issuance. A consumer token identifies a principal (human or organisation). A delegation grant authorises a specific agent, identified by its own consumer token, to act on behalf of the owner for a defined set of operations on a defined set of instances.

1.6.2. Delegation Scopes

The following delegation scope values MUST be supported:

Scope	Description
devices.read	Read device instance records (Level 2) for delegated instances, including network addresses and api_endpoint fields
devices.presence	Receive presence change notifications (online/offline transitions) for delegated instances via webhook
devices.command	Authorised to call the manufacturer's device API on behalf of the owner, using the api_endpoint from the instance record

Table 5

Scopes are additive. A delegation grant MUST specify at least one scope. devices.command MUST NOT be granted without also granting devices.read, as command routing requires knowledge of the current api_endpoint.

1.6.3. Delegation Grant API

The owner creates a delegation grant via the APIX delegation endpoint:

```
POST /devices/{instance_id}/delegations
Authorization: Bearer owner-consumer-token
Content-Type: application/json

{
  "agent_token_id": "token_id of the agent's consumer token",
  "scopes": ["devices.read", "devices.command"],
  "expires_at": "ISO 8601 timestamp -- REQUIRED",
  "note": "human-readable description -- OPTIONAL"
}
```

agent_token_id is the non-secret token identifier of the agent's registered consumer token. The owner does not need to hold or transmit the agent's token secret; the token_id is sufficient for APIX to resolve the delegation at query time.

expires_at is REQUIRED. Perpetual delegation grants are not permitted. Maximum delegation duration MUST be enforced by the index operator; the RECOMMENDED maximum is 365 days. Owners SHOULD grant the minimum duration sufficient for the agent’s task.

APIX MUST return the delegation grant identifier (delegation_id) in the response. The owner uses delegation_id to list, inspect, and revoke grants.

1.6.4. Delegated Agent Query

An agent that holds a valid delegation grant queries device instances using its own consumer token:

```
GET /devices/{instance_id}
Authorization: Bearer agent-consumer-token
```

APIX MUST resolve the delegation at query time: if a valid, non-expired delegation grant exists for the requesting token_id covering the requested instance_id with the devices.read scope, APIX MUST return the full Level 2 instance record as if the agent were the owner.

If no valid delegation exists, APIX MUST return an empty result (consistent with the enumeration prevention requirement in Section 6.2).

1.6.5. Delegation Revocation

An owner revokes a delegation by calling:

```
DELETE /devices/{instance_id}/delegations/{delegation_id}
Authorization: Bearer owner-consumer-token
```

Revocation is immediate. Any in-flight agent request that has not yet received a response at the time of revocation MUST return an empty result. APIX MUST NOT return instance data to a revoked delegation.

Delegation grants are also revoked automatically when: - The delegation expires_at timestamp is reached - The owner releases their ownership claim on the instance - The agent’s consumer token is revoked by the agent itself or by APIX - The instance is placed in quarantine state (fleet circuit breaker)

1.6.6. Delegation and Privacy

Delegation grants are personal data: they link a device owner identity to an agent identity. They MUST be subject to the same data minimisation and retention rules as other personal data in device instance records. On receipt of a device owner erasure request, all delegation grants for that owner's instances MUST be revoked and deleted.

1.7. Open Questions

The following questions are unresolved and explicitly invited for community input:

1. *Human operator as a first-class entity in the IoT service chain*: This specification recognises two human roles: the device owner (owner_id, a passive claimant) and the authorised human approver in the LER dual-control protocol (Section 9). Neither constitutes a first-class, addressable participant in the automated service chain.

   The AINS Internet-Draft ([I-D.vandemeent-ains-discovery]) defines a "human" entity type whose identity is bound to the person rather than to any device — a human operator who can be directly addressed within an agent ecosystem, for example via a phone number, a push endpoint, or a terminal identifier. This raises the question of whether APIX IoT should formalise an equivalent concept: a *Human Operator Profile* that allows an automated IoT service chain to include a callable human participant as an official link.

   Concrete motivating scenarios include: an agent requiring human confirmation before executing a high-impact device command; a device entering a fault state that requires human attention before the automated chain can continue; and regulatory human-oversight requirements for AI-driven device control (such as those implied by EU AI Act Article 50).

   The existing LER dual-control protocol demonstrates that APIX already supports the human-in-the-loop pattern in a specific context. The open question is whether this pattern should be generalised into a first-class Human Operator entity type with its own presence, delegation, and contact-endpoint semantics — and what the privacy, GDPR, and contact-protocol implications of doing so would be. A future Human Operator Profile is the candidate home for this work. Community input is invited on the scope, contact protocol requirements, and governance model for such a profile.

2. *Device API call budget as a capability descriptor*: Adjacent discovery specifications introduce rate limit fields on registered services and devices. This specification deliberately omits a rate limit field. Rate limiting as a defence against malicious or abusive callers is not a discovery-layer concern: APIX's trust model (O-levels, S-levels, consumer token system) addresses bad actors systemically at the index layer, and individual device registrations MUST NOT be designed around the assumption of hostile consumers. Surfacing a device-level rate limit as a security measure would reintroduce the adversarial framing that APIX's trust model is specifically designed to retire.

   There is, however, a distinct and legitimate use case that is unrelated to security: *agent workflow planning against hardware constraints*. A resource-constrained embedded device — limited processor, limited memory, firmware-defined command queue — has a physical maximum throughput that is a factual property of the device class, no different from its supported_api_versions or presence_mode. An agent orchestrating a multi-step workflow across a fleet of such devices needs this information before execution begins, not as a wall it discovers mid-run. A future api_call_budget object on the device class APM — framed explicitly as a *capability descriptor*, not a security mechanism — could serve this planning use case. Example fields: max_commands_per_minute (hardware ceiling), typical_response_ms (planning latency), concurrent_command_limit (firmware queue depth). Community input is invited on whether this capability descriptor should be standardised in a future revision, and on the correct schema to represent hardware-bound throughput constraints without reintroducing defensive rate-limit semantics.

1.8. Security Considerations

1.8.1. Instance Token Security

Device instance tokens are long-lived secrets stored on constrained hardware. Implementors MUST store tokens in hardware-backed secure storage (secure enclave, TPM, or equivalent) where the device platform supports it. Tokens MUST NOT be transmitted in log output, diagnostic APIs, or any channel other than the APIX presence endpoint Authorization header.

The token_id is a non-secret correlation handle. It MAY be logged and MAY appear in diagnostic interfaces without security risk.

1.8.2. Layer 2 Enumeration Prevention

APIX MUST return an empty result set (not HTTP 404 or HTTP 403) for authenticated queries that would match instances belonging to other principals. An attacker with a valid consumer token MUST NOT be able to determine whether a given instance_id or device_class_id exists in the system by observing error responses.

1.8.3. Hub Compromise Scope

A compromised hub instance can misrepresent the state (online status, api_version, hub_gateway_endpoint) of devices it genuinely controls. It cannot forge presence signals for devices whose instance tokens it does not hold. The blast radius of a hub compromise is bounded to the set of devices paired to that hub. This is a smaller blast radius than cloud_relay compromise, where the manufacturer’s cloud holds all fleet tokens.

1.8.4. Presence Signal Replay

APIX MUST enforce that presence signals are processed at-most-once within the heartbeat window. Duplicate signals within the same heartbeat_interval_seconds window MUST be idempotently accepted (no state change) rather than counted as separate registrations.

1.8.5. Physical Device Privacy

The two-layer model (Section 3) is a security requirement, not only a privacy preference. Index implementations MUST enforce it at the infrastructure layer. Returning instance-level data to unauthenticated queries is a critical vulnerability, not a configuration error.

1.8.6. APIX as a High-Value Attack Target

APIX occupies a structurally unique position: it is the single presence endpoint for potentially millions of physical devices and holds the mapping between device identifiers, network addresses, and owner identities at global scale. A full compromise of the APIX index is not a data breach in the conventional sense — it is a global occupancy intelligence leak combined with the ability to disable an arbitrary set of connected devices by mass-revoking their instance tokens. This threat profile requires architectural mitigations beyond standard application security.

APIX implementations MUST satisfy the following architectural requirements:

* The presence endpoint (which receives device signals) MUST be architecturally separated from the token store. A compromise of the presence endpoint infrastructure MUST NOT yield access to the token hash store or the Layer 2 owner-to-network-address mappings.

* APIX MUST NOT store plaintext device instance tokens at any point in the live system. Only the SHA-256 hash of each token is stored, as specified in Section 3. Hash storage and the presence endpoint MUST run in separate trust boundaries.

* Token issuance (the process by which new tokens are generated and delivered to manufacturers) MUST be performed in an air-gapped or hardware-security-module (HSM) isolated process. Token issuance infrastructure MUST NOT be reachable from the public internet.

* The presence endpoint MUST be geographically distributed across independent infrastructure regions. No single region outage or compromise MUST affect global device presence signalling.

* All administrative operations that affect token validity (revocation, suspension, fleet circuit breaker actions) MUST be recorded in a tamper-evident append-only audit log. This log MUST be replicated to at least two geographically separate locations. Log entries MUST be individually signed.

* APIX MUST publish an annual infrastructure security report disclosing the number of tokens issued, rotated, and revoked; the number of presence signals processed; and any confirmed security incidents.

1.8.7. Fleet Circuit Breaker

An emergency mechanism MUST be available to index operators to quarantine a compromised manufacturer fleet or a rogue device class at the index layer, without requiring physical access to the affected devices.

Two distinct compromise scenarios are addressed:

Scenario A — Manufacturer systems compromised. An adversary has gained access to the manufacturer's cloud infrastructure and is forging presence signals on behalf of real devices (false liveness, false network addresses, false api_version). The appropriate response is to suspend the manufacturer account and revoke all fleet tokens for the affected device classes, following the same atomic four-step procedure defined for sanctions suspension in [APIX-CORE]. Token revocation MUST trigger HTTP 401 token_revoked at the presence endpoint, and all affected instances MUST be marked offline immediately.

Scenario B — Devices themselves are weaponised. The physical devices are operating normally from the device credential perspective (valid tokens, authentic presence signals) but are under adversary control and are being used maliciously. In this scenario, revoking device tokens would brick legitimate consumer devices without stopping the threat. The appropriate response is presence signal suppression without token revocation:

APIX MUST support a quarantine state for individual device instances or entire device classes. In quarantine state:

* Presence signals from quarantined instances MUST continue to be accepted at the presence endpoint. The device is unaware of the quarantine; presence processing appears normal.

* api_endpoint fields MUST be withheld from all Layer 2 query responses for quarantined instances. The consuming agent receives an instance record showing the device as online but with no reachable endpoint. No agent can connect to the device through APIX.

* Quarantined instances MUST be excluded from Layer 1 capability searches, as if lifecycle_stage were end_of_life.

* The quarantine state MUST be reversible. When the quarantine is lifted, api_endpoint fields are restored and the instance returns to normal Layer 2 visibility.

* All presence signals received while an instance is quarantined MUST be logged with enhanced detail (source IP, signal content, timing) and retained for a minimum of twelve (12) months for law enforcement purposes.

The fleet circuit breaker MUST be operable at three scopes: instance (single device), device_class (all instances of a class), and manufacturer (all classes and instances under an organisation account). The scope MUST be explicitly declared in the activation record. Activation MUST require authorisation from at least two independent APIX operator roles (dual-control requirement).

1.8.8. Law Enforcement Cooperation Interface

APIX MUST provide a Law Enforcement Request (LER) interface: a restricted-access channel through which authorised public authorities may submit IP addresses of device instances identified as members of a malicious botnet, together with a suppression action request.

Accepted jurisdictions:

Two authority tiers are defined. Tier I covers supranational judicial instruments; Tier II covers national law enforcement requests. Both tiers require a judicial authorisation document. Administrative orders without judicial oversight MUST NOT be accepted at either tier.

Tier I — Supranational (automatically accepted):

Orders and provisional measures issued by the *International Court of Justice (ICJ)* are accepted automatically and are not subject to the Accepted Jurisdiction Whitelist. The ICJ is the principal judicial organ of the United Nations (ICJ Statute, June 1945). Its provisional measures under Article 41 of the ICJ Statute are legally binding on states and represent the highest-authority legal instrument available for cross-jurisdictional cases where no single national court has clear competence over a multi-state botnet operation. An LER citing a verified ICJ provisional measure or final judgment MUST be processed within 4 hours of receipt and is not subject to the 72-hour provisional reversal window; the suppression remains in force for the duration stated in the ICJ instrument.

Tier II — National law enforcement (whitelist-governed):

The LER interface MUST only accept requests from public authorities in jurisdictions that satisfy ALL of the following criteria:

* The jurisdiction has ratified the Budapest Convention on Cybercrime, OR is the domicile jurisdiction of the APIX governing body (currently Switzerland).

* The request is accompanied by a valid judicial authorisation document: a court order, magistrate warrant, or equivalent judicial instrument issued by a court of competent jurisdiction in the requesting state.

* The jurisdiction is included in the governing body's Accepted Jurisdiction Whitelist, maintained as a public document and updated by governing body board decision.

Jurisdiction Contact Registry:

The governing body MUST maintain a Jurisdiction Contact Registry (JCR) as part of the APIX operational infrastructure. The JCR is established during system ramp-up, before the LER interface is opened for submissions. A jurisdiction MUST NOT be added to the Accepted Jurisdiction Whitelist until all three mandatory contact roles have been confirmed for that jurisdiction.

Three contact roles MUST be registered per Tier II jurisdiction:

Operational contact — the national authority responsible for submitting LERs: typically the national cybercrime unit, CERT, or equivalent law enforcement body with cybercrime mandate (e.g., Bundeskriminalamt in Germany, ANSSI in France, FBI Cyber Division in the United States). This contact submits LER requests and provides the IP address lists.

Judicial contact — the authority responsible for obtaining and validating judicial authorisation: the Ministry of Justice, Attorney General, Prosecutor General, or equivalent government legal division with authority to commission court orders in the requesting jurisdiction. This contact is the issuing or co-signing party for the judicial authorisation document accompanying each LER. Including the judicial/legal division of government in the contact structure is REQUIRED — law enforcement submission without a validated judicial pathway is insufficient to activate the LER process.

Emergency contact — a named duty officer or on-call role reachable 24 hours a day, 7 days a week, for Tier I (ICJ) and emergency provisional national requests. MUST be distinct from the operational contact and MUST have authority to act on behalf of the jurisdiction without waiting for business-hours approval.

For the *ICJ (Tier I)*, the governing body MUST establish contacts with the following bodies during system ramp-up:

* *ICJ Registry (Greffe de la Cour)*: the administrative organ of the Court, responsible for receiving and communicating all instruments. Contact for standard ICJ instrument submission.

* *President's Office of the ICJ*: the authority responsible for ordering provisional measures under Article 41. Contact for time-critical ICJ provisional measure cases requiring 4-hour processing.

The governing body MUST formally notify each registered contact of the following at initial establishment and whenever the LER framework is materially updated:

1. The existence and purpose of the LER interface.

2. The legal basis under which APIX operates (Swiss Stiftung, Budapest Convention obligations, applicable data protection law).

3. The contact's specific role in the LER process and what actions are expected of them.

4. The APIX operational security contact and escalation path.

5. The jurisdiction's reference code in the JCR, to be cited in all LER submissions from that jurisdiction.

The governing body MUST review and re-confirm all JCR contacts at least annually. Contacts that cannot be confirmed MUST be flagged; the associated jurisdiction MUST be placed in suspended status on the Accepted Jurisdiction Whitelist until confirmed contacts are restored. A jurisdiction in suspended status MUST NOT submit new LERs; active suppressions from that jurisdiction remain in force.

Process tiers:

Tier	Authorisation required	Processing time	Effect
ICJ instrument	Verified ICJ provisional measure or judgment	MUST be processed within 4 hours	Fleet circuit breaker per instrument scope; no 72-hour reversal window
Emergency provisional (national)	Written agency declaration with case reference; judicial authorisation to follow within 72 hours	MUST be processed within 4 hours of receipt	Fleet circuit breaker Scenario B (endpoint suppression); full review opens automatically
Standard (national)	Verified judicial authorisation document	MUST be processed within 48 hours of verification	Fleet circuit breaker action per request scope
Refusal	Authorisation absent, invalid, jurisdiction suspended in JCR, or contact roles not established	N/A	Request refused; attempt logged; aggregate count included in transparency report

Table 6

A national emergency provisional suppression that is not followed by verified judicial authorisation within 72 hours MUST be automatically reversed. The governing body MUST notify both the operational contact and the judicial contact of the reversal. ICJ-based suppressions are not subject to automatic reversal; they run for the duration specified in the ICJ instrument.

LER submission protocol:

The LER interface is a restricted-access HTTPS endpoint. It is not publicly documented or reachable from the open internet. Access is limited to network addresses registered for the jurisdiction's operational contact in the JCR.

Authentication: LER submissions MUST use mutual TLS (mTLS). Each registered operational contact is issued a client certificate by the APIX governing body at the time their jurisdiction is onboarded to the JCR. The certificate is scoped to the submitting jurisdiction's reference code. Certificate issuance and renewal is the governing body's operational responsibility.

Each JCR contact record MUST include a preferred_language field: a single BCP 47 language tag ([RFC5646]) indicating the language in which APIX conducts the voice callback verification with that contact. APIX MUST use this language for all spoken exchanges during the callback procedure, including reading the APIX TAN and requesting the personal verification code and uploader auth TAN. If the contacted person responds in a different language, APIX MUST treat this as an anomaly and log it, but MUST NOT reject the callback on language grounds alone. Example values: "de" for Bundeskriminalamt, "en" for FBI Cyber Division, "fr" for ANSSI.

In-jurisdiction dual authorisation: Each jurisdiction MUST register at least two named operational contacts in the JCR, each holding an independently issued client certificate. A single individual MUST NOT hold both certificates; APIX MUST enforce this by verifying that the countersigning contact's certificate subject is distinct from the submitting contact's certificate subject.

A valid LER submission MUST be countersigned by a second registered contact from the same jurisdiction before it enters the human review queue. The protocol uses a two-step commit:

1. The primary contact submits the LER via their authenticated mTLS connection. The submission is recorded with status pending_countersign and no review is initiated.

2. A second registered contact from the same jurisdiction MUST authenticate via their own mTLS certificate and issue a countersign request referencing the ler_id returned in the initial submission response. The countersign window is 60 minutes for standard and ICJ tier; 20 minutes for emergency_provisional tier.

3. If the countersign is not received within the window, the submission MUST be automatically rejected and logged with reason countersign_timeout. No review is initiated and no suppression is applied.

4. For emergency_provisional tier, the 4-hour processing commitment runs from the timestamp of the countersign, not the initial submission.

Submission endpoint:

The LER interface endpoint URL is not publicly published. It is communicated to each jurisdiction's registered operational contact as part of the JCR onboarding process. The interface requirements are:

```
POST <ler-endpoint-url>
Authorization: mTLS client certificate (jurisdiction contact)
Content-Type: application/json
```

The <ler-endpoint-url> is operator-specific and MUST NOT be discoverable from the public Index API. Implementations MUST ensure the LER endpoint is not reachable from network addresses outside those registered for the jurisdiction's operational contact in the JCR.

Request format:

```
{
  "jurisdiction_ref": "JCR reference code -- issued by APIX",
  "tier": "emergency_provisional | standard | icj",
  "judicial_ref": "court order or ICJ instrument reference number",
  "judicial_doc_url": "HTTPS URL -- APIX-verifiable document",
  "case_ref": "internal case reference of the submitting authority",
  "device_addresses": [
    {"type": "ipv4", "address": "203.0.113.42"},
    {"type": "ipv6", "address": "2001:db8::1"}
  ],
  "requested_scope": "instance | device_class | manufacturer",
  "requested_duration_hours": 168,
  "emergency_declaration": "tier: emergency_provisional only"
}
```

device_addresses MUST contain at least one entry. Bulk requests without specific device addresses (e.g., "all devices in jurisdiction X") MUST be refused per the Non-Surveillance Commitment in [APIX-CORE] Section 5.

requested_duration_hours is advisory. APIX MUST apply the suppression for the duration specified in the judicial instrument if it differs. For emergency_provisional, the suppression is automatically reversed after 72 hours unless judicial authorisation is verified (Section above).

Response format:

```
{
  "ler_id": "APIX-assigned unique LER identifier",
  "status": "accepted | refused | pending_countersign | ...",
  "estimated_processing_seconds": 14400,
  "refusal_reason": "null | invalid_jurisdiction | ... (see spec)",
  "apix_tan": "one-time code (pending_countersign only; null)",
  "uploader_auth_tan": "one-time code (countersign only; null)"
}
```

ler_id is the only external reference APIX issues for an LER case. It MUST be used in all subsequent communications about the case and is included in the annual transparency report aggregate statistics.

apix_tan and uploader_auth_tan are one-time codes generated per submission and returned only in the initial POST response when the submission is accepted for the countersign step. They are shown once and never again.

APIX MUST NOT provide any endpoint or mechanism to retrieve the TAN values after the initial POST response. The fields MUST be absent — not null, not redacted — from all subsequent API responses including LER status queries, countersign responses, and any other interface. No GET path to TAN values exists. This follows the same principle as one-time API key generation: if the recipient does not retain the values from the initial response, they are permanently unavailable.

Both codes expire upon completion of any verification exchange in which credentials are requested from the contacted person, regardless of whether the exchange results in confirmation or failure. An unanswered call attempt does not constitute a verification exchange and does not consume the codes. A new submission generates new codes; expired codes cannot be reissued for the same ler_id.

APIX MUST store both codes internally in a form that permits verification during the callback (hashed) but does not expose plaintext values through any interface after the initial POST response.

Countersign endpoint:

The second registered contact MUST submit the countersign via an authenticated request to the same LER interface:

```
POST <ler-endpoint-url>/countersign
Authorization: mTLS client certificate (second jurisdiction contact)
Content-Type: application/json

{
  "ler_id": "APIX-assigned LER identifier (initial submission)",
  "countersign_statement": "confirmation statement (see spec)"
}
```

Response:

```
{
  "ler_id": "...",
  "status": "countersigned | rejected",
  "rejection_reason": "null | window_expired | same_contact | ..."
}
```

On countersigned status, the submission transitions to the callback verification step (see below). On rejected, the submission is terminated and MUST be re-submitted from the beginning.

Submission callback verification:

After countersign is confirmed and before the submission enters the human review queue, APIX MUST perform a voice callback to the jurisdiction’s registered operational contact. The callback uses three independent verification elements:

* *APIX TAN* — a one-time code returned in the initial submission response. APIX reads this to the contacted person at the start of the call, proving the caller is the legitimate APIX system.

* *Personal verification code* — a standing secret established per registered contact during JCR onboarding, known only to that individual and APIX. This code is never transmitted via any API or electronic channel.

* *Uploader auth TAN* — a one-time code returned in the initial submission response alongside the APIX TAN. The contacted person reads this back to APIX, binding the answering individual to the specific submission.

APIX MUST require both the personal verification code and the uploader auth TAN together in a single exchange. Neither alone constitutes confirmation.

The callback procedure is:

1. APIX retrieves the stored callback number from the JCR record for the submitting jurisdiction.

2. Immediately before initiating the callback (within 5 minutes of countersign confirmation), APIX MUST re-validate the stored number against the independent authoritative public source established at onboarding time.

3. If the stored number does not match the current public record, APIX MUST NOT call either number. The submission MUST be immediately suspended and an incident procedure initiated. No suppression is applied during suspension. The submission remains suspended until the incident is resolved and the callback number is re-established with confidence.

4. If the stored number matches the current public record, APIX initiates the call to the stored number and reads the APIX TAN. This allows the contacted person to verify they are speaking to the legitimate APIX system before providing any credential.

5. APIX then requests, in a single exchange: the contacted person’s full name, personal verification code, and uploader auth TAN.

6. APIX verifies all three against its stored records. Any mismatch on any element MUST result in immediate termination of the call. No partial credit applies. The failure mode then determines the disposition (see below).

7. If the call is not answered on the first attempt, APIX MUST make one further attempt before permanently rejecting the submission. Two unanswered attempts exhaust the callback allowance. If the contact answers and a verification exchange begins, no further attempt is permitted regardless of outcome — the exchange result is final. If the contacted person explicitly denies any knowledge of the submission, the LER is permanently rejected with reason callback_verification_failed (see below).

Callback failure dispositions differ by cause and MUST be handled distinctly:

callback_verification_failed — operational failure, permanent rejection: Triggered when credential verification fails (wrong or absent uploader auth TAN or personal verification code), when the call is not answered after two attempts, or when the contact denies the submission. The LER is permanently rejected. This is not a security event: no Security Officer escalation is initiated. The jurisdiction MUST file a new LER; the rejected ler_id cannot be recovered. Logged as operational failure.

callback_number_mismatch — security event, submission suspended: Triggered when the stored callback number does not match the current public record at re-validation time (step 2 above). The submission is suspended pending incident resolution. Security Officer is notified. No suppression is applied. The submission remains suspended until the callback number is re-established with confidence and the incident is closed.

callback_interception_reported — confirmed security incident, submission suspended: Triggered when the contacted person reports having already received a call from a party claiming to be APIX before this call. The submission is immediately suspended. The Security Officer and a governing body board member MUST be notified. The incident is treated as a confirmed attempt to compromise the LER verification process and handled under the security incident response procedure in [APIX-CORE] Section 12.

The callback confirmation is a mandatory pre-condition for entering the human review queue. It does not substitute for the human review and dual-control activation procedure.

APIX MUST NOT acknowledge by any means that a suppression is active on a specific device or instance when responding to queries from parties other than the submitting jurisdiction. Device owner queries for affected instances MUST receive responses indistinguishable from an offline or unreachable instance.

Audit logging obligations:

Every event in the LER lifecycle MUST be written to the tamper-evident audit log defined in [APIX-CORE] Section 12. Log entries MUST be individually signed and immutable. The following events are individually logged:

Event	Logged fields
LER received	ler_id, timestamp, jurisdiction_ref, tier, device_addresses count, receiving operator identity
Countersign received	ler_id, timestamp, countersigning contact certificate subject (distinct from submitting contact), window elapsed
Countersign timeout	ler_id, timestamp, window duration, disposition (auto-rejected)
Callback number validated	ler_id, timestamp, stored number hash, public record hash, match result (match / mismatch)
Callback phone mismatch — callback_number_mismatch	ler_id, timestamp, stored number hash, public record hash, incident_id; disposition: suspended
Callback confirmed	ler_id, timestamp, attempt count, name match (boolean), personal code match (boolean), uploader auth TAN match (boolean), governing body operator identity (caller)
Callback rejected — callback_verification_failed	ler_id, timestamp, attempt count, cause (credential_mismatch / no_answer / contact_denied), failed element if credential_mismatch (name / personal_code / uploader_auth_tan — do not log the submitted value); disposition: permanently rejected, no escalation

Callback suspended — callback_interception_reported	ler_id, timestamp, governing body operator identity (caller), contact statement summary; disposition: suspended, Security Officer + board member notified
LER refused	ler_id, timestamp, refusal_reason, reviewing operator identity
Suppression activated	ler_id, timestamp, scope, device instance count affected, authorising operator identities (both, per dual-control)
Suppression reversed	ler_id, timestamp, reason (72h expiry / judicial doc not verified / operator decision), reversing operator identity
Review completed	ler_id, timestamp, reviewer identity, decision (confirmed / escalated / reversed), notes

Table 7

The audit log MUST be the system of record for all LER activity. The annual transparency report derives its aggregate statistics from it; the log is the authoritative source and MUST be retained for a minimum of seven (7) years from the date of each entry.

Human authorisation and review:

APIX MUST NOT activate a suppression through a purely automated path. Every suppression activation requires human authorisation and post-hoc review by a named, accountable operator. The requirements differ by tier:

Standard and ICJ tiers:

1. Upon receipt of a valid LER submission (certificate valid, format valid, jurisdiction in accepted whitelist), the submission enters a human review queue. No suppression is active at this point.

2. A designated LER reviewer (a named APIX operator with the ler_review role) MUST read the submission and the referenced judicial document before authorising activation.

3. Activation MUST follow the dual-control requirement of the fleet circuit breaker ([APIX-CORE] Section 12): two independent operators with the ler_activate role MUST independently confirm before suppression is applied. Neither operator may be the same individual who reviewed the submission.

4. The activation record in the audit log MUST include the identities of both activating operators and the timestamp of each confirmation.

5. A post-activation review MUST be completed within 72 hours by a third operator (the LER oversight role) confirming that the activation was lawful, proportionate, and consistent with the judicial instrument scope. The review outcome MUST be recorded in the audit log.

Emergency provisional tier (4-hour window):

The 4-hour processing commitment creates time pressure. The following modified procedure applies:

1. Automated validation on receipt: certificate valid, jurisdiction in whitelist, emergency declaration field present, at least one device address specified. If validation passes, the submission immediately enters the emergency queue with an alert to the on-call LER reviewer.

2. The on-call LER reviewer MUST authorise activation within 4 hours. Activation requires only one authorising operator (single-control) in this tier due to the time constraint.

3. A second operator MUST complete a dual-control confirmation within 24 hours of activation. If the second confirmation is not received within 24 hours, the suppression MUST be automatically reversed.

4. The post-activation oversight review (step 5 above) MUST be completed within 24 hours of activation (not 72 hours) for emergency provisional cases.

5. Judicial authorisation MUST be verified within 72 hours (the automatic reversal window). If the judicial document review confirms the authorisation is invalid or the scope is exceeded, the suppression MUST be reversed immediately regardless of the dual-control confirmation in step 3.

Operator role structure:

The following named operator roles MUST be defined and staffed in the APIX operating model. No individual MUST hold more than two of these roles simultaneously (segregation of duties):

Role	Responsibility
ler_reviewer	Reviews LER submissions and judicial documents before authorisation. May not activate.
ler_activate	Provides one of the two required authorisation confirmations for activation. May not review the same submission.
ler_oversight	Performs post-activation review. Must be independent of both reviewer and activating operators for the case under review.
ler_emergency	On-call role, 24/7. Holds both ler_reviewer and single-control ler_activate authority for emergency provisional tier only.

Table 8

Role assignments MUST be published in the governing body's annual security report. Changes to role assignments MUST be logged in the audit log.

Any operator who receives an instruction — from any source, including governing body board members — to activate a suppression outside this procedure MUST refuse and MUST exercise the whistleblower protection right defined in [APIX-CORE] Section 12. This right is not waivable by role, contract, or board resolution.

Suppression effect:

LER-triggered suppression follows the fleet circuit breaker Scenario B model: presence signals continue to be accepted (the adversary controlling the botnet has no signal of identification), while

api_endpoint fields are withheld from Layer 2 responses. The suppression scope is specified by the requesting authority as a list of device IPv4 and/or IPv6 addresses; APIX matches these against active presence signal source addresses.

APIX MUST NOT disclose to any party other than the requesting authority that a specific device instance is under LER-triggered suppression. Device owners querying their instance records MUST receive the same response as for any offline or unreachable instance; the suppression reason MUST NOT be disclosed.

Transparency:

APIX MUST publish aggregate LER statistics in its annual transparency report, disclosing: number of LER requests received by requesting jurisdiction, number accepted, number refused, number of provisional suppressions reversed, and number of device instances affected (by jurisdiction, not by individual identity). No individual case details, case references, or device identifiers MAY appear in the transparency report.

1.8.9. Bad-Bot Graduated Response

APIX MUST maintain a Bad-Bot Registry: a classified list of device instance identifiers and consumer token identifiers that have been confirmed as operating maliciously. Sources for registry entries include: LER submissions (Section 5.7), manufacturer abuse reports, and APIX operator investigation of anomalous signal patterns (e.g., rapid multi-instance registration from a single IP, heartbeat flooding, or token reuse patterns inconsistent with legitimate device operation).

Three response tiers are defined. The active tier for each registry entry MUST be explicitly recorded in the Bad-Bot Registry and MUST be changeable by operator action.

Tier 1 — observe (investigation in progress; entry not yet confirmed hostile or confirmation is recent):

* Presence endpoint: Accept signal normally. Instance record updated. HTTP 200 returned. No signal to the device that it is under observation.

* Consumer query endpoint: Full response returned normally.

* Logging: Enhanced. All presence signals and consumer queries from this identity MUST be logged with full detail (IP, payload, timing, response) and retained for a minimum of twelve (12) months.

Tier 2 — degrade (confirmed hostile; investigation active; covert restriction required to limit harm without alerting adversary):

* Presence endpoint: Accept signal normally (HTTP 200). Instance record updated to reflect online status. However, api_endpoint fields MUST be withheld from all Layer 2 query responses for the affected instance (fleet circuit breaker Scenario B). If active blocking of the presence signal itself is additionally required during this phase, APIX MUST return HTTP 401 with error code token_revoked — identical to a normal token rotation response. This response MUST NOT include any indication that the device has been identified as malicious.

* Consumer query endpoint: Response returned with an intentional processing delay of 210 seconds (configurable per entry). Results MAY be limited to high-trust entries only (org_level_min: O-2, service_level_min: S-2). No error is returned; the delay and filtering are silent.

* Logging: Enhanced, as per observe.

Tier 3 — block (law enforcement action taken or imminent; covert restriction no longer required):

* Presence endpoint: MUST return HTTP 401 with error code device_quarantined. This response explicitly signals that the device's APIX access has been administratively suspended. This tier MUST be activated in coordination with the relevant law enforcement authority and SHOULD be timed to coincide with or follow an enforcement action (arrest, infrastructure seizure, botnet takedown).

* Consumer query endpoint: MUST return HTTP 403. No result data returned.

* Logging: Full detail, with real-time alert to APIX operator security team.

Tier transitions MUST follow the direction observe → degrade → block. Downgrade (e.g., block to degrade) is permitted only by explicit operator authorisation with documented justification. Removal from the Bad-Bot Registry requires dual-control operator authorisation and is logged permanently in the tamper-evident audit log.

The existence of the Bad-Bot Registry and its aggregate statistics (total entries by tier, by source type, by jurisdiction) MUST be disclosed in the annual transparency report. Individual registry entries MUST NOT be disclosed publicly.

2. References

2.1. Normative References

[APIX-CORE] Rehfeld, C., "API Index (APIX): Core Infrastructure for Autonomous Agent Service Discovery", Work in Progress, Internet-Draft, draft-rehfeld-apix-core-04, n.d., <https://datatracker.ietf.org/doc/draft-rehfeld-apix-core/>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/rfc/rfc5646>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

2.2. Informative References

[APIX-SERVICES] Rehfeld, C., "APIX Services Profile: Discovery Infrastructure for Web API and Bot Services", Work in Progress, Internet-Draft, draft-rehfeld-apix-services-01, n.d., <https://datatracker.ietf.org/doc/draft-rehfeld-apix-services/>.

[I-D.vandemeent-ains-discovery]
van de Meent, J. and R. AI, "AINS: AInternet Name Service - Agent Discovery and Trust Resolution Protocol", Work in Progress, Internet-Draft, draft-vandemeent-ains-discovery-01, 29 March 2026, <https://datatracker.ietf.org/doc/html/draft-vandemeent-ains-discovery-01>.

[RFC8594] Wilde, E., "The Sunset HTTP Header Field", RFC 8594, DOI 10.17487/RFC8594, May 2019, <https://www.rfc-editor.org/rfc/rfc8594>.

[RFC9745] Dalal, S. and E. Wilde, "The Deprecation HTTP Response Header Field", RFC 9745, DOI 10.17487/RFC9745, March 2025, <https://www.rfc-editor.org/rfc/rfc9745>.

Appendix A. Change Log

-00: Initial submission.

-01: Stream and intended status changed: the document moves from the Independent Submission stream (Informational) to the IETF stream (Standards Track), following IETF dispatch guidance. No other content changes.

A.1. IANA Considerations

A.1.1. Additions to the APIX Protocol Type Registry

This document requests the following additions to the Protocol Type Registry maintained by the governing body at apix.example.org/registry/protocols:

Registry value	Standard	Spider behaviour	Status
device-class	APIX IoT Device Profile (this document)	No Spider crawl -- presence-based liveness	active
hub	APIX IoT Device Profile (this document)	No Spider crawl -- presence-based liveness	active

Table 9

A.1.2. APIX Presence Mode Registry

This document defines the initial contents of the APIX Presence Mode Registry, maintained by the governing body at apix.example.org/registry/presence-modes. Values in this registry MUST be used in the presence_mode field of device class APMs.

Value	Description	Trust ceiling
push	Device signals directly to APIX presence endpoint without intermediary	S-4
cloud_relay	Manufacturer cloud relays signals on behalf of the device	S-1
hub	Hub gateway relays signals for mesh-protocol devices	S-1

Table 10

New values MAY be registered by the governing body through its registry extension process. Each new value MUST specify the trust ceiling it imposes and the conditions under which APIX can independently verify the signal’s authenticity.

A.1.3. APIX Device Delegation Scope Registry

This document defines the initial contents of the APIX Device Delegation Scope Registry, maintained by the governing body at apix.example.org/registry/delegation-scopes. Values in this registry MUST be used in the scopes field of delegation grant requests (Section 6.3).

Scope value	Description	Requires
devices.read	Read Layer 2 device instance records (network addresses, api_endpoint) for delegated instances	—
devices.presence	Receive presence change notifications (online/offline transitions) via webhook for delegated instances	—
devices.command	Call the manufacturer’s device API via api_endpoint on behalf of the device owner	devices.read

Table 11

New scope values MAY be registered by the governing body. Each new scope MUST specify: the operations it permits, any prerequisite scopes, and whether it implies any data access beyond what is already visible in the device instance record.

A.1.4. Additions to the APIX Capability Taxonomy Registry

This document requests the following additions to the Capability Taxonomy Registry maintained by the governing body at apix.example.org/registry/capabilities:

Term	Description	Status
iot	Sensor and device data	active
home	Smart home and connected appliance services	active
home.appliance	Connected home appliances (device-class services)	active
home.appliance.dishwasher	Dishwasher appliance	active
home.appliance.heating	Heating and climate control	active
home.appliance.washing	Washing machine and dryer	active
home.appliance.cooking	Oven, hob, and cooking appliances	active
home.appliance.refrigeration	Refrigerator and freezer	active
home.energy	Home energy management and tariff services	active
home.energy.tariff	Energy tariff and pricing APIs	active
home.energy.grid	Grid demand and load balancing signals	active

Table 12

A.2. References

A.2.1. Normative References

* [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

* [RFC9110] Fielding, R., et al., "HTTP Semantics", RFC 9110, June 2022.

* [APIX-CORE] Rehfeld, C., "API Index (APIX): Core Infrastructure for Autonomous Agent Service Discovery", draft-rehfeld-apix-core-04.

A.2.2. Informative References

* [RFC8594] Wilde, E., "The Sunset HTTP Header Field", RFC 8594, May 2019.

* [RFC9745] Cedik, A., et al., "The Deprecation HTTP Header Field", RFC 9745, March 2025.

* [APIX-SERVICES] Rehfeld, C., "APIX Services Profile: Discovery Infrastructure for Web API and Bot Services", draft-rehfeld-apix-services-01.

A.3. Author’s Address

Carsten Rehfeld Email: carsten@botstandards.org

Author’s Address

Carsten Rehfeld
Email: carsten@botstandards.org