Human Identity Protocol — Specification

Field	Value
Version	1.0-draft
Status	Draft
Date	2026-03-27
License	Apache 2.0 (this specification)

1. Abstract

Account creation on the internet has no cost. Bot farms, AI agents, and automated accounts exploit this to flood platforms with fake engagement, destroying trust at scale. Every platform fights this arms race independently, with no shared infrastructure.

The Human Identity Protocol (HIP) solves this at the identity layer. A person verifies once — using a government-issued document and a liveness check — and receives a certificate from a provider. Any platform can then query the provider to confirm the person is human. The provider returns a signed attestation containing a status and a confidence score. No personal data crosses the boundary.

HIP proves humanness, not citizenship or legal identity. A government document is used as the strongest available signal that the bearer is a real person, not as a citizenship check. The protocol does not serve as a government enforcement mechanism.

2. Conventions

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 RFC 2119.

The protocol version described in this document is **HIP/1.0**.

Sections marked **[v2]** describe planned extensions and are informative, not normative.

3. Terminology

Term	Definition
Provider	An entity that operates the identity verification pipeline, issues certificates, and responds to verification requests. Registered in the registry.
Platform	An entity that integrates HIP to verify its users. Identified by a canonical platform ID assigned by the registry.
User	A natural person who has completed identity verification with a provider.
Registry	A public, append-only, cryptographically signed log of providers and platforms. The only shared infrastructure in the protocol.
Certificate	An Ed25519 keypair issued to a user upon successful identity verification, binding the user's identity to a cryptographic key.
Subject ID	A per-platform, per-user opaque identifier derived from the user's master secret and the platform's canonical ID. Format: `{derived_id}@id.{provider_domain}`. The `id.` prefix is a namespace marker that prevents collision with real email addresses.
Master Secret	A cryptographically random 256-bit secret unique to each user, held by the provider, used to derive per-platform subject IDs.
Canonical Platform ID	A globally unique, registry-assigned identifier for a platform. Stable across rebrands.
Confidence Score	An integer 0–100 representing the provider's confidence that the user is a real, non-compromised human. Decays over time; drops on risk events.
Score State	One of `stable`, `recently_dropped`, or `recovering`. Indicates the trajectory of the score.
Identity Verification	The process by which a user proves their humanness to a provider: document verification, active liveness check, and human review. Also referred to as "verification" in user-facing contexts.
Platform Verification	The process by which a platform queries a provider about a user's status and score.
Attestation	A signed verification response from a provider, formatted as a JWS compact serialization (RFC 7515).
Content Hash	An irreversible SHA-256 hash of a normalized document field, stored for deduplication and fraud detection.
Conflict	A condition where two verification attempts match on identity signals (document number, name + date of birth, or face similarity).
Nonce	A random string included in verification requests and echoed in responses, used for replay prevention.
Human Verifier	A trained specialist who conducts live video review of verification cases, using physical-world challenges to defeat deepfake attacks.

4. Identity Model

4.1 Identifier Format

A HIP identifier has the form:

```
{derived_id}@id.{provider_domain}
```

- `derived_id` is a 22-character base64url string (128-bit truncated HMAC output, no padding).
- `id.` is a fixed namespace prefix indicating this is a HIP identity (not a real email address).
- `provider_domain` is the provider's registry-assigned domain (e.g., `humanidentity.io`).

Example: `xK7mN2pR9sT4vW6yBQw@id.humanidentity.io`

The `id.` prefix ensures HIP identifiers cannot collide with real email addresses at the provider's domain. It is a namespace marker only — not a resolvable subdomain.

This is the identifier that users present to platforms and that platforms include in verification requests.

4.2 Subject ID Derivation

A provider MUST derive the subject ID for a given user and platform as follows:

```
derived_id = Base64URL(HMAC-SHA256(user.master_secret, canonical_platform_id ":" country)[0:16])
```

Where:

- `user.master_secret` is a cryptographically random 256-bit (32-byte) secret generated at user creation and stored by the provider.
- `canonical_platform_id` is the platform's registry-assigned identifier.
- `country` is the ISO 3166-1 alpha-2 code of the user's verified document-issuing country (e.g., `US`, `NG`).
- The separator is a literal colon character (U+003A).
- The HMAC-SHA256 output (32 bytes) is truncated to the first 16 bytes (128 bits).
- The truncated output is base64url-encoded without padding (RFC 4648 §5) to produce a 22-character string.

The full subject ID presented to the platform is `{derived_id}@id.{provider_domain}`.

Truncation rationale: 128 bits provides 2^64 collision resistance, which is more than sufficient for the expected user population. The shorter identifier is more user-friendly and resembles an email address format.

Properties:

- The same user produces a different `derived_id` for every platform (different `canonical_platform_id`).
- The same user verified in a different country produces a different `derived_id` for the same platform (different `country`).
- The provider can recompute any user's subject ID for any platform without storing a mapping table.
- Platforms cannot correlate users across platforms without access to the master secret.

4.3 Privacy Guarantees and Limitations

No two platforms receive the same `derived_id` for the same user. Cross-platform tracking by identifier is not possible without the provider's cooperation.

The provider holds the master secret and can compute all per-platform IDs for a user. This is an honest limitation of the server-side key storage model used in HIP/1.0.

4.4 Key Storage Tiers [v2]

Future versions of the protocol define three key storage tiers that progressively reduce the provider's ability to correlate users:

Tier	Storage	Provider Can Correlate?	Recovery
1 (default, v1)	Server-side	Yes, if compelled	Standard re-verification
2 (opt-in)	Device + encrypted backup	Only with user authentication	Device backup + re-verification
3 (advanced)	Device only, no backup	No, under any circumstance	User-managed seed phrase only

Tier selection is invisible to platforms.

---

5. Identity Verification Process

5.1 Requirements

A provider MUST verify the following before issuing a certificate:

1. **Email address** — verified via magic link or one-time code.
2. **Government-issued identity document** — passport, national ID card, or driver's license.
3. **Active liveness check** — a challenge-response liveness test (not passive detection). MUST be certified to at least iBeta PAD Level 2 or equivalent.
4. **Human verifier review** — a live video session with a trained specialist who applies physical-world challenges.

A provider SHOULD additionally collect a phone number as a convenience authentication factor.

5.2 Verification Tiers

Identity verification attempts are routed to one of three tiers based on risk signals and random selection:

Tier	Selection Rate	Method
Standard	~85–90%	Automated document + active liveness check, followed by remote human verifier with physical-world challenges
Enhanced	~8–12%	Standard tier, plus an extended human verifier session with a writing challenge
Physical	~2–5%	In-person verification at a partner location, with video recording reviewed by an independent remote verifier

Selection is risk-weighted. Higher-risk verification attempts — flagged by device fingerprint clustering, temporal patterns, or composite risk signals — are routed to Enhanced or Physical tiers at higher rates. A percentage of low-risk verifications are randomly selected for higher tiers as a deterrence mechanism.

5.3 Human Verifier Challenges

A human verifier MUST use at least one physical-world challenge that requires interaction with the physical environment. These challenges are designed to defeat deepfake injection attacks by requiring actions that are difficult to synthesize in real time.

Examples of compliant challenges:

- **Writing challenge**: Server generates a random 4–6 character code; user writes it on paper and holds it to the camera.
- **Hand-face interaction**: "Touch your left ear with your right hand" — exploits rendering glitches at the boundary

where hands cross the face.

- **Physical object**: "Hold up N fingers" or display a colored object.
- **Environment change**: "Move to a window" or "turn a light on/off."
- **Multi-angle rotation**: "Slowly rotate your phone around your head."

5.4 Data Handling

Category	Data	Storage Form	Retained?
Identity	Full name	Encrypted at rest (envelope encryption)	Yes
Identity	Date of birth	Encrypted at rest (envelope encryption)	Yes
Auth	Email address	Plaintext	Yes
Auth	Phone number	Plaintext	Yes (optional)
Dedup	Document number	SHA-256 hash only (irreversible)	Yes
Dedup	Normalized name + DOB	SHA-256 hash only (irreversible)	Yes
Audit	Vendor check reference ID	Plaintext	Yes
Biometric	Facial templates	**Never stored** — delegated to liveness vendor	Deleted after verification
Document	Raw document images	**Never stored** by provider	Deleted immediately after verification

A provider MUST delete all raw document images and biometric data immediately after the verification decision is made. No exceptions.

5.5 Consent

A provider MUST obtain explicit, affirmative consent before collecting biometric data (liveness check). The consent interaction MUST be standalone — not bundled with terms of service.

The consent record MUST include:

- Consent version identifier
- Timestamp (ISO 8601)
- IP address
- Device fingerprint

Consent records are append-only and MUST NOT be modified or deleted during the account's active lifetime. Providers MAY delete consent records after the legally required retention period following account closure (e.g., 3 years per BIPA).

5.6 Re-verification Triggers

A provider MUST require re-verification when any of the following occur:

- The user's identity document has expired.
- The user has been inactive for more than 12 months.

- The user's confidence score drops below 40.
- The user requests re-verification (to reset their score).

On successful re-verification, the provider issues a new certificate and resets the confidence score to 100.

5.7 Identity Verification Status Values

An identity verification attempt progresses through the following statuses:

```
pending → document_uploaded → verifying → completed | failed | conflict_detected
```

Status	Meaning
`pending`	Verification initiated, awaiting document upload
`document_uploaded`	Document received, awaiting liveness and verification
`verifying`	Liveness and/or human verifier review in progress
`completed`	Verification successful, certificate issued
`failed`	Verification rejected (document invalid, liveness failed, etc.)
`conflict_detected`	Identity signals match an existing account — see Section 12

---

6. Platform Verification Protocol

This section is normative. A compliant provider MUST implement the verification endpoint exactly as specified.

6.1 Endpoint

```
POST https://{provider_domain}/.well-known/hip/verify
```

The path `/.well-known/hip/verify` is fixed and MUST NOT vary.

6.2 Request Format

```
POST /.well-known/hip/verify
Content-Type: application/json
Authorization: Bearer {platform_api_key}
```

Headers:

Header	Required	Description
`Authorization`	REQUIRED	Bearer token. An API key issued by the provider to the platform. The provider identifies the platform from the key.
`Content-Type`	REQUIRED	MUST be `application/json`.

Request body:

```
{
  "subject_id":    "string",
  "minimum_score": 0,
  "purpose":       "string",
  "nonce":         "string"
}
```

Field	Type	Required	Description
`subject_id`	string	REQUIRED	The `derived_id` portion of the HIP identifier (22 base64url characters). Does not include the `@id.{provider_domain}` suffix.
`minimum_score`	integer	OPTIONAL	The platform's score threshold (0–100). Informational — the provider returns the actual score regardless. The platform makes the accept/reject decision.
`purpose`	string	OPTIONAL	Optional platform context. Common values: `account_creation`, `age_verification`, `re_authentication`, `high_value_transaction`.
`nonce`	string	REQUIRED	A cryptographically random string (minimum 16 characters, maximum 128 characters) for replay prevention.
`hip_version`	string	OPTIONAL	The protocol version the platform expects. Defaults to `"1.0"`. Providers MAY ignore this field in HIP/1.0.

6.3 Response Format

The response is a JWS (JSON Web Signature) compact serialization as defined in RFC 7515:

```
base64url(header) "." base64url(payload) "." base64url(signature)
```

The HTTP response body is the JWS compact string. Content-Type is `application/jose`.

JWS Header:

```
{
  "alg": "EdDSA",
  "kid": "abcdef0123456789abcdef0123456789"
}
```

- `alg` MUST be `EdDSA`.
- `kid` MUST be a 32-character lowercase hex string derived from the provider's signing key: the first 16 bytes of `SHA-256(DER(public_key))`, hex-encoded. It is an opaque key lookup identifier, not a cryptographic proof. This allows the platform to look up the correct public key from the registry.

Payload (decoded):

```
{
  "subject_id":              "xK7mN2pR9sT4vW6yBQw",
  "status":                  "active",
  "score":                   82,
  "score_state":             "stable",
  "score_components": {
    "verification_age_days": 180,
    "recent_events":         ["phone_changed_45d_ago"],
```

```
    "active_flags":          []
  },
  "certificate_fingerprint": "sha256:abcdef...",
  "issued_at":               "2025-09-01T00:00:00Z",
  "expires_at":              "2025-09-01T00:05:00Z",
  "nonce":                   "echoed-from-request"
}
```

Field	Type	Description
`subject_id`	string	MUST match the request `subject_id`.
`status`	string	One of the user status values defined in Section 8.
`score`	integer	The user's current confidence score (0–100).
`score_state`	string	One of `stable`, `recently_dropped`, `recovering`. See Section 7.4.
`score_components.verification_age_days`	integer	Days since the user's last successful verification.
`score_components.recent_events`	array of string	Recent events affecting the score. Each entry is formatted as `{event_type}_{N}d_ago` where `event_type` is one of: `phone_changed`, `email_changed`, `new_device`, `inactivity`, `failed_mfa`, `platform_report`, and `N` is the number of days since the event.
`score_components.active_flags`	array of string	Active flags on the account. Valid values: `account_suspended`, `under_review`, `inactive_suspended`, `account_deceased`.
`certificate_fingerprint`	string	SHA-256 fingerprint of the user's certificate public key, formatted as `sha256:{hex}`.
`issued_at`	string	ISO 8601 timestamp when this attestation was created.
`expires_at`	string	ISO 8601 timestamp when this attestation expires. MUST NOT exceed 5 minutes from `issued_at`.
`nonce`	string	MUST match the request `nonce` exactly.

Providers SHOULD include an `HIP-Version: 1.0` HTTP response header alongside the JWS body.

6.4 Platform Verification Flow

A platform MUST verify an attestation using the following steps:

1. Receive the JWS compact string from the provider.
2. Split on `"."` to obtain `header`, `payload`, and `signature` (all base64url-encoded).
3. Decode the header. Extract the `kid` value.
4. Look up the provider's public key from the registry using the `kid`.
5. Verify the Ed25519 signature over `base64url(header) "." base64url(payload)`.
6. Decode the payload.
7. Verify `nonce` matches the original request.

8. Verify `expires_at` has not passed.
9. Apply platform-specific policy based on `status`, `score`, and `score_state`.

A platform MUST reject the response if any of steps 5–8 fail.

6.5 Error Responses

When a provider cannot fulfill a verification request, it MUST return a JSON error response (not a JWS):

```
{
  "error": {
    "code": 404,
    "message": "subject not found"
  }
}
```

Code	Meaning
400	Bad request (malformed JSON, missing required fields)
401	Unauthorized (invalid, missing, expired, or revoked API key)
403	Forbidden (platform suspended or revoked)
404	Subject not found (no user with this subject ID)
409	Nonce already used (replay detected)
429	Rate limited
500	Internal server error

6.6 Nonce Replay Prevention

A provider MUST maintain a nonce store. Upon receiving a verification request, the provider MUST check whether the nonce (scoped to the requesting platform) has been seen before. If it has, the provider MUST reject the request with a 409 error.

The nonce store MUST retain entries for a minimum of 24 hours.

6.7 Platform Authentication (API Keys)

Platforms authenticate to the provider's SDK-facing endpoints (`/.well-known/hip/verify`, `/.well-known/hip/exchange`, `/oauth/token`, and `/v1/platforms/{id}/*`) using opaque API keys.

Property	Value
Format	`hip_sk_{64-character hex}` (≥256 bits of entropy)
Transport	`Authorization: Bearer {api_key}` (MUST be sent over TLS)
Storage (provider)	SHA-256 hash only; plaintext is shown once at creation
Scope	Bound to a single platform. The provider identifies the platform from the key.
Lifetime	No expiry by default; platforms MAY set an expiry at creation
Rotation	Platforms MAY have multiple active keys simultaneously to enable rolling rotations
Revocation	Immediate. Revoked keys MUST NOT be accepted.

A provider MUST reject requests with a missing, malformed, unknown, expired, or revoked API key. A provider MUST enforce per-key rate limits to contain abuse from a leaked key.

Providers MAY return `402 Payment Required` when provider-enforced billing policy blocks the platform (e.g. subscription past-due). Platforms SHOULD treat a `402` response as a temporary block and surface the billing state to their operators. This response is optional; providers that do not charge per verification will never return `402`.

Rationale: API keys are chosen over signed JWTs because the protocol already relies on the provider's Ed25519 signature on the response for integrity. The platform-to-provider direction does not need a second signature layer and benefits from simpler SDK implementation. The provider DB stores only hashes, so a DB leak cannot be used to impersonate a platform.

---

7. Confidence Score

7.1 Range and Floor

Confidence scores are integers in the range [0, 100].

- **Initial score**: 100 (set upon successful identity verification or re-verification).
- **Floor**: 20. A user's score MUST NOT drop below 20 through decay or event drops. For active users, the effective score range is [20, 100]. A score of 0 indicates the account is non-functional (suspended, deceased, or suspended_inactive) and MUST NOT be returned for active users.

The score represents the provider's confidence that the account is controlled by the person who originally verified.

7.2 Time-Based Decay (Protocol Constant)

The decay schedule is a **protocol constant**. All compliant providers MUST implement the identical decay curve. Changing the decay curve would break cross-provider score comparability.

Period	Days	Score Range
Post-verification	0	100
Year 1	1–365	100 → 90
Years 2–3	366–1095	90 → 70
Years 4–5	1096–1825	70 → 50
Beyond year 5	1826+	50 → 20 (floor)

The decay function is continuous and linear within each period. Providers MUST compute the current time-based score using the following formula:

```
Given d = days since last successful verification:

if d <= 0:
    time_score = 100
elif d <= 365:
    time_score = 100 - (10 * d / 365)
elif d <= 1095:
    time_score = 90 - (20 * (d - 365) / 730)
elif d <= 1825:
    time_score = 70 - (20 * (d - 1095) / 730)
else:
    time_score = max(20, 50 - (30 * (d - 1825) / 1825))
```

The result MUST be rounded to the nearest integer.

7.3 Event-Based Drops (Policy)

Event-based score drops are **policy decisions**. Providers MAY adjust the magnitudes. The following are the recommended values:

Event	Default Drop	Recovery
Phone number changed	−30	+5 per month if stable
Email changed	−10	Permanent (no auto-recovery)
New device (first 30 days)	−15	Auto-recovers after 30 days
Inactivity > 12 months	−20	Requires re-verification
3+ failed 2FA attempts	−10	Auto-recovers after successful 2FA
Platform-reported suspicious activity	−25	Appealable via re-verification

A provider MUST document its event drop policy and make it available to platforms.

7.4 Score States

State	Definition
`stable`	No event drops in the last 90 days. Score follows the decay curve normally.
`recently_dropped`	One or more event drops occurred within the last 30 days.
`recovering`	The most recent event drop occurred between 30 and 90 days ago, and no new drops have occurred since.

7.5 Score Composition

Providers maintain each user's score by applying the decay curve (Section 7.2) and event drops (Section 7.3) as they occur. The verification response returns the provider's current score for the user.

The effective score at any point in time is:

```
effective_score = clamp(time_score + sum(active_event_drops), floor=20, ceiling=100)
```

Where `active_event_drops` is the sum of all currently active event drop values (negative numbers). Event drops that have fully recovered are no longer active.

The decay schedule in Section 7.2 is the normative reference. Providers MAY implement it via periodic batch updates rather than real-time computation, provided the returned score is within 1 point of the reference value.

7.6 Suggested Platform Thresholds (Informative)

Score Range	Suggested Use Case
80–100	Banking, finance, age verification
60–79	Social platforms, marketplaces
40–59	Comments, low-stakes access
< 40	Require re-verification before granting access

These are suggestions. Platforms set their own thresholds based on risk tolerance.

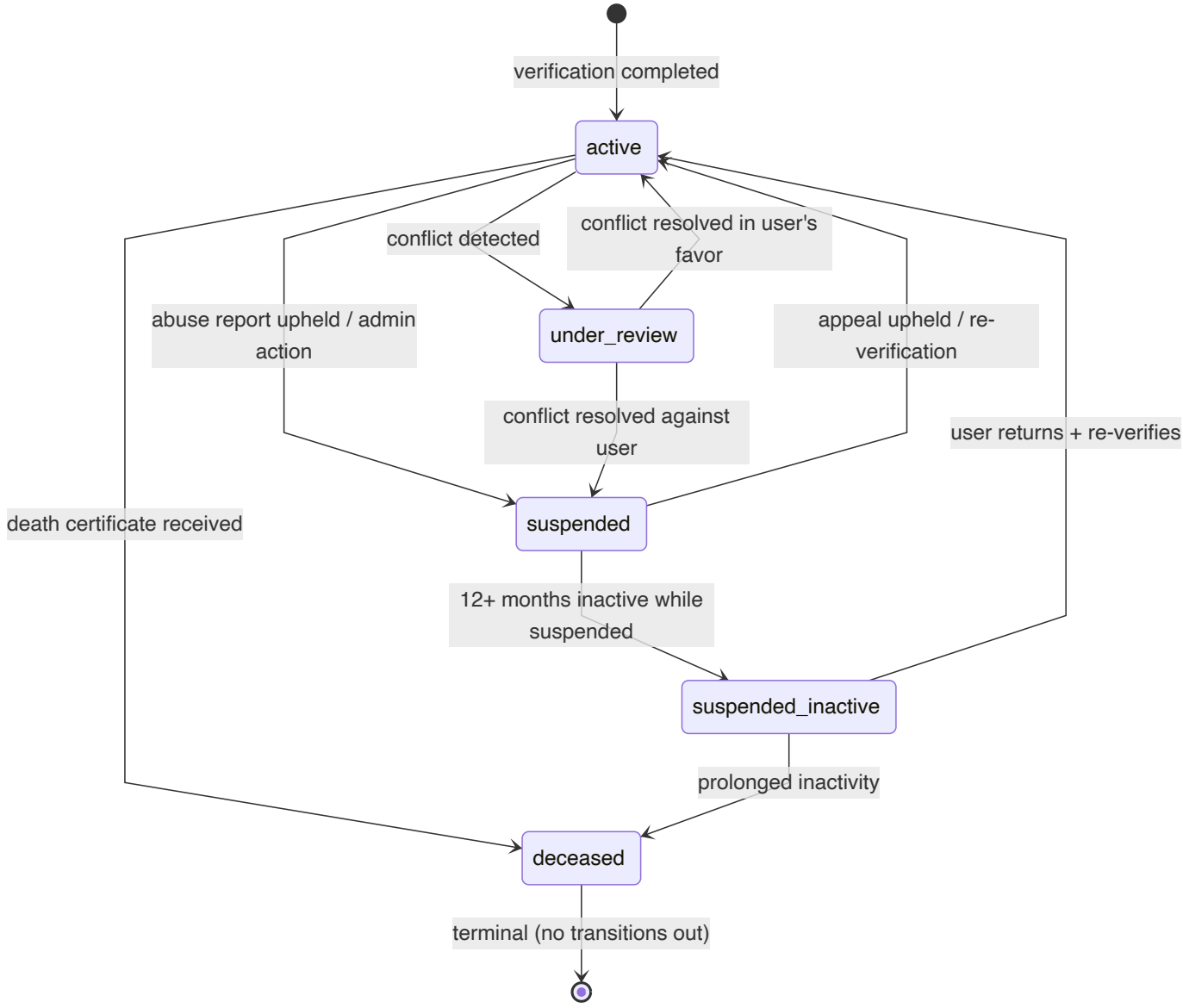
---

8. User Status

8.1 Status Values

Status	Meaning	Verification Response Behavior
`active`	User is in good standing.	Return score and attestation normally.
`suspended`	Account flagged for abuse or policy violation.	Return `status: "suspended"`, `score: 0`.
`under_review`	Conflict investigation in progress. Score frozen.	Return `status: "under_review"`, score frozen at pre-conflict value.
`deceased`	Certificate permanently revoked.	Return `status: "deceased"`, `score: 0`.
`suspended_inactive`	Suspended with prolonged inactivity. Likely deceased or permanently abandoned.	Return `status: "suspended_inactive"`, `score: 0`.

8.2 Status Transitions



The `deceased` status is terminal. No transitions out of `deceased` are permitted.

---

9. Certificates

9.1 Structure

A provider MUST issue an Ed25519 keypair for each user upon successful identity verification.

Field	Type	Description
`fingerprint`	string	SHA-256 hash of the DER-encoded public key, formatted as `sha256:{lowercase_hex}`.
`public_key`	bytes	The Ed25519 public key (32 bytes).
`issued_at`	string	ISO 8601 timestamp of issuance.
`expires_at`	string	ISO 8601 timestamp of expiry.

9.2 Validity

Certificate validity SHOULD be between 1 and 5 years. The provider determines the validity period based on its re-verification cadence and risk posture. The RECOMMENDED default is 3 years.

9.3 Revocation

A provider MUST revoke a certificate when:

- The user's status transitions to `deceased`.
- The user requests account deletion.
- A conflict is resolved against the user (`verify_new` resolution — see Section 12).

Revocation is immediate. A provider MUST NOT return attestations referencing a revoked certificate.

9.4 Reissuance

On re-verification, the provider MUST issue a new certificate (new Ed25519 keypair). The previous certificate is revoked. The subject ID does not change — it is derived from the master secret, not the certificate.

---

10. Registry

10.1 Purpose

The registry is a public, append-only, cryptographically signed log of providers and platforms. It is the only shared infrastructure in the protocol. Providers and platforms never communicate directly except through the verification endpoint.

10.2 Provider Entry

```
{
  "provider_id":          "provider.example.com",
  "display_name":         "Example Provider",
  "well_known_url":       "https://provider.example.com/.well-known/hip",
  "public_key":           "base64-ed25519-public-key",
  "public_key_id":        "abcdef0123456789abcdef0123456789",
  "certificate_tier":     "provisional",
  "certificate_expires": "2027-06-01T00:00:00Z",
```

```
  "supported_documents": ["passport:US", "national_id:NG"],
  "supported_countries": ["US", "NG"],
  "coverage_hours_utc":  "00:00-24:00",
  "status":              "active",
  "registered_at":       "2025-01-15T00:00:00Z",
  "issued_by":           "root_ca_id",
  "verification_count":  1500000
}
```

The `well_known_url` is the base path for protocol endpoints. Platforms MUST append `/verify` (Section 6.2) or `/exchange` (Section 20.3) to construct the full endpoint URL. The `public_key_id` matches the `kid` value used in JWS headers (see Section 6.3).

10.3 Platform Entry

```
{
  "platform_id":           "uuid",
  "canonical_platform_id": "platform.example.com",
  "legal_entity":          "Platform Inc.",
  "parent_entity":         "Parent Corp.",
  "domain":                "platform.example.com",
  "redirect_uris":         ["https://platform.example.com/oauth/callback"],
  "status":                "active",
  "registered_at":         "2025-01-15T00:00:00Z"
}
```

Field	Type	Description
`platform_id`	string	Registry-assigned UUID.
`canonical_platform_id`	string	Stable identifier used in subject ID derivation (Section 4.2). Survives domain changes.
`legal_entity`	string	The registered legal name of the platform operator.
`parent_entity`	string	The ultimate beneficial owner, for report grouping (Section 13.4). OPTIONAL.
`domain`	string	The platform's primary domain.
`redirect_uris`	array of string	Registered OAuth redirect URIs (Section 21.5). REQUIRED if the platform uses the OAuth flow; otherwise MAY be empty.
`status`	string	One of `active` or `disabled`.
`registered_at`	string	ISO 8601 timestamp of registration.

10.4 Trust Hierarchy

```
Root CA
  +-- Sovereign (government-operated, 5-year certificate)
  +-- Established (vetted provider, 1–2 year certificate, can sponsor Provisional providers)
        +-- Provisional (new provider, 90-day certificate, < 10k verifications/month)
```

Promotion criteria:

- **Provisional to Established**: Volume threshold met, human vetting process passed, no sanctions on record.
- **Established to Sovereign**: Government-operated, extensive vetting completed.

Providers never communicate directly with each other. The registry is the only trust anchor.

10.5 Endpoints

```
GET https://registry.humanidentity.io/providers
GET https://registry.humanidentity.io/providers/{provider_id}
GET https://registry.humanidentity.io/providers/{provider_id}/certificate
GET https://registry.humanidentity.io/platforms/{canonical_platform_id}
```

10.6 Caching

Data	Recommended Cache TTL
Provider public keys	24 hours
Provider status	1 hour

SDKs and platforms SHOULD implement a last-known-good fallback when the registry is unavailable. If the cache has expired and the registry cannot be reached, the platform MAY use the last known value and SHOULD flag the verification result as `registry_stale`.

10.7 Key Rotation

A provider MUST announce key rotation at least 30 days in advance via the registry. During the rotation window, the registry entry MUST include a `pending_public_key` and `pending_public_key_id` alongside the active key. Platforms resolve which key to use via the `kid` in the JWS header. After the rotation window closes, the pending key becomes the active key and the old key is removed. The old key MUST be treated as invalid after the window closes.

---

11. Cryptographic Operations

11.1 Algorithms

Operation	Algorithm	Reference
Protocol signatures (attestations)	Ed25519 (EdDSA)	RFC 8032
Subject ID derivation	HMAC-SHA256	RFC 2104
Document field hashing (dedup)	SHA-256	FIPS 180-4
Attestation format	JWS compact serialization	RFC 7515
Platform API key hashing	SHA-256	FIPS 180-4

No alternative algorithms are permitted for these operations in HIP/1.0.

11.2 JWS Details

- **Header**: `{"alg":"EdDSA","kid":"{key_id}"}` — both fields REQUIRED.
- **`kid` derivation**: `Hex(SHA-256(DER(public_key)))[0:32]` — the first 16 bytes (32 hex characters) of the SHA-256 hash of the DER-encoded (PKIX) public key. This is an opaque lookup identifier; it is NOT the same as the certificate

fingerprint (which hashes the raw key bytes).

- **Payload**: The verification response JSON, serialized without unnecessary whitespace.
- **Signature**: Ed25519 signature over `base64url(header) "." base64url(payload)`.
- **Output**: The three segments joined by `"."` — `header.payload.signature`.

11.3 Field Normalization Before Hashing

Before computing content hashes for deduplication and conflict detection, fields MUST be normalized according to their type:

Field Type	Normalization Steps
Name	NFC unicode normalization → lowercase → trim leading/trailing whitespace → collapse internal whitespace to single spaces → replace hyphens with spaces → strip apostrophes (both straight `'` and curly `'` `'`)
Date	Providers MUST receive dates from identity verification vendors in ISO 8601 format (YYYY-MM-DD). Non-ISO formats (DD.MM.YYYY, MM/DD/YYYY) MUST be converted to ISO 8601 before normalization. Normalization: strip all non-digit characters to produce YYYYMMDD. `1990-01-15` normalizes to `19900115`.
Document ID	Lowercase → strip spaces, hyphens, and dots. `AB-123.456` normalizes to `ab123456`.

After normalization, the content hash is: `SHA-256(normalized_value)`, stored as a lowercase hex string.

11.4 Key Generation

Providers MUST generate Ed25519 keys from a cryptographically secure random source (e.g., `/dev/urandom`, `crypto/rand`). Private keys MUST be stored encrypted at rest. The master secret (for ID derivation) MUST be generated independently from the signing key.

---

12. Conflict Detection and Resolution

12.1 Detection Triggers

A provider MUST check for conflicts during the identity verification process when any of the following match an existing account:

- **Document number hash**: Exact SHA-256 match on the normalized document number.
- **Name + DOB hash**: Exact SHA-256 match on the normalized composite of full name and date of birth.
- **Face similarity**: Liveness vendor reports a face match above a demographic-calibrated threshold (not a fixed percentage).

12.2 Conflict States

```
pending → under_review → resolved
```

When a conflict is detected, both the new verification attempt and the matched existing user account enter `under_review` status.

12.3 Resolution Options

A human verifier resolves conflicts using one of four options:

Resolution	Effect on New Verification	Effect on Existing Account
`verify_new`	Accepted. Certificate issued.	Suspended. Certificate revoked.
`reject_new`	Rejected.	Unaffected (returns to previous status).
`hold_both`	Held. Escalated to senior review.	Held. Escalated to senior review.
`verify_both`	Accepted. Certificate issued.	Unaffected.

`verify_both` is used for rare cases where two distinct individuals legitimately share identity signals (e.g., twins, document numbering errors).

12.4 Resolution Principles

- **Evidence quality takes precedence over verification seniority.** The first person to verify does not automatically win a conflict.
- The protocol does not adjudicate legal identity disputes. Unresolvable conflicts MUST be escalated per the provider's jurisdiction-specific legal process.
- A human verifier MAY request supplementary documents (utility bills, tax records) from both parties during a live video session. Supplementary documents are reviewed visually — not processed through automated pipelines.

Note: Throughout this section, "new verification" and "existing account" refer respectively to the user attempting verification and the previously verified user who triggered the conflict.

12.5 Resolution SLA

Providers SHOULD resolve conflicts within 14 days. During resolution, both accounts remain in `under_review` status, and platforms querying either account receive `status: "under_review"`.

---

13. Platform Reporting

13.1 Report Structure

Platforms MAY submit abuse reports about users. Reports are scoped to **identity concerns only** — the provider verifies humanness, not platform-specific policy. General fraud, TOS violations, and content moderation are the platform's responsibility.

A report MUST include:

- **Report type**: One of `suspicious_activity`, `impersonation`, `automated_behavior`.
  - `suspicious_activity` — patterns suggesting the account is not controlled by a genuine human.
  - `impersonation` — suspected stolen or fabricated identity (triggers conflict investigation).
  - `automated_behavior` — suspected bot or AI agent operating without human oversight.
- **Evidence hashes**: Timestamps and log hashes (hashed, not transmitted in full to the provider). Platforms MUST retain the original evidence for a minimum of 365 days from submission so the hash can be re-verified during review or appeal.
- **Multi-strike history**: Pattern description, not a single incident.

13.2 Fee Model

Platforms pay a fee ($5–$10 USD equivalent) per report on submission. The fee is refunded if the report is upheld after review. Platforms with a greater than 80% uphold rate over a quarter MAY have fees waived.

13.3 Score Impact

Score impact is **delayed**. The provider reviews the report before any score change.

- **If upheld**: −25 score drop (appealable by the user via re-verification).
- **If rejected**: No score impact. The platform is notified.
- **Review SLA**: 72 hours for initial assessment.

A single platform report alone MUST NOT trigger account suspension. Suspension requires multiple independent reports or a provider-initiated investigation.

13.4 Parent Entity Grouping

Reports are grouped by ultimate beneficial owner (parent entity). One parent entity counts as one reporting source, regardless of how many platforms it operates.

Example: If Google, YouTube, and Gmail all report the same user, that counts as one report from Alphabet Inc.

13.5 Platform Accountability

Patterns of frivolous reports (consistently rejected by provider review) result in reporting privilege suspension. Reinstatement requires review and a waiting period.

---

14. Authentication

14.1 Email as Primary Anchor

Email is the primary identity anchor. Phone numbers are recycled by carriers (sometimes within 90 days), are vulnerable to SIM swap attacks, and are lost when users change countries. Email provides greater stability and deliverability.

14.2 Authentication Methods

Priority	Method	Description
1	TOTP (RFC 6238)	Time-based one-time password via authenticator app. RECOMMENDED as strongest factor.
2	Email magic link + code	Provider sends a one-time link and/or 6-digit code to the verified email.
3	SMS OTP	One-time code sent to verified phone number. Convenience factor only.

14.3 Recovery

Scenario	Recovery Path
Phone lost	Email verification + re-verification
Email lost	Full re-verification through the identity verification pipeline
Both lost	Full re-verification (by design — no backdoors)

There is no special recovery path that bypasses the identity verification pipeline. Recovery is re-verification.

14.4 Sensitive Operations

The following operations MUST require full re-authentication (not just a session token):

- Change phone number
- Change email address
- Close account
- Appeal a score drop

---

15. Data Minimization and Privacy

15.1 Data Classification

Data	Storage Form	Purpose	Retention
Full name	Encrypted (envelope encryption)	Account identity	Account lifetime
Date of birth	Encrypted (envelope encryption)	Re-verification cadence, age signals	Account lifetime
Email	Plaintext	Authentication anchor	Account lifetime
Phone	Plaintext	Convenience 2FA	Account lifetime
Document number	SHA-256 hash only	Dedup / fraud detection	Indefinite (irreversible)
Name + DOB composite	SHA-256 hash only	Conflict detection	Indefinite (irreversible)
Vendor check ID	Plaintext	Audit trail	Account lifetime
Facial templates	**Never stored**	N/A	Deleted by vendor after check
Raw document images	**Never stored**	N/A	Deleted immediately after verification
Score, events, certificates	Provider-managed	Protocol operation	Account lifetime

15.2 What Platforms Receive

A platform receives ONLY the following from a verification response:

- User status
- Confidence score
- Score state
- Score components (verification age, recent events, active flags)
- Certificate fingerprint
- Attestation timestamps

No name, no date of birth, no document details, no biometric data, no email, no phone number.

15.3 Government Surveillance Limitations

Per-platform identifiers make cross-platform correlation difficult without the provider's cooperation. Decentralization (multiple providers across jurisdictions) limits the scope of any single government demand.

However: if a government legally compels a Tier 1 provider within its jurisdiction, the provider can reconstruct per-platform ID mappings. The only mitigation is Tier 3 key storage (v2), jurisdictional diversity, and legal resistance.

Honest position: the protocol makes mass surveillance expensive. It does not make it impossible.

---

16. Security Considerations

16.1 Threat Model

Threat	Mitigation
Deepfake injection during liveness	Active liveness (PAD Level 2) + human verifier with physical-world challenges + device integrity checks
Document forgery	Vendor document validation + content hash dedup + human verifier review
Identity theft (stolen documents)	Conflict detection on document number, name+DOB, and face similarity. Human resolution.
SIM swap	Phone is not the primary anchor. Phone changes require full re-verification and trigger −30 score drop.
Provider compromise	Ed25519 key rotation via registry. Certificate revocation. Affected users re-verify with new keys.
Provider DB compromise	API keys stored as SHA-256 hashes only; leaked DB cannot be used to impersonate platforms. Attacker must steal plaintext keys from each platform's own infrastructure.
Registry compromise	Append-only log with cryptographic signatures. Key pinning in SDKs.
Government coercion	Per-platform IDs. Jurisdictional diversity. Transparency log. User can report provider to registry.
Replay attacks	Nonce in every verification request. 24-hour nonce TTL. Attestation expires in 5 minutes.
Leaked platform API key	Immediate revocation via platform dashboard. Per-key rate limits contain damage. Platform may rotate by creating a new key and revoking the old once deployed.
Industrialized verification fraud	Device fingerprint clustering, IP velocity limits, face embedding clustering, document perceptual hashing

16.2 Liveness Defense Layers

1. **Vendor liveness**: Active challenge-response, iBeta PAD Level 2 certified.
2. **Device integrity**: Virtual camera detection, root/jailbreak detection, emulator detection. Mobile-first verification RECOMMENDED.

3. **Human verifier**: Physical-world challenges (writing, hand-face interaction, environment changes).
4. **Server-side detection**: Device fingerprint clustering, temporal pattern analysis, face embedding clustering, document perceptual hashing.
5. **Post-verification retrospective**: Re-analyze stored signals when new attack techniques are discovered.

16.3 Identity Verification Velocity Limits

Providers MUST implement rate limits on identity verification attempts:

- Per device fingerprint: maximum N verification attempts per day.
- Per IP address: maximum N verification attempts per day.
- Per phone number: maximum N verification attempts per day.

After 3 failed attempts from the same signal cluster, a cooldown period applies: 24 hours, then 7 days, then 30 days (escalating).

16.4 SIM Swap Protection

Phone is a convenience factor, not the primary anchor. A phone number change:

- Requires full identity re-verification (not just SMS confirmation of the new number).
- Triggers a −30 score drop with slow recovery (+5/month).
- The new phone number is not trusted for recovery for 30–90 days.

The protocol does not attempt to detect SIM swaps. The score penalty handles the risk window.

---

17. Known Limitations

The following are honest limitations of the protocol:

- **Account rental**: A verified human can let someone else use their account. The protocol raises the cost of abuse (one human per account) but cannot prevent sharing.
- **Humans-for-hire**: Real humans can be paid to complete verification and then hand over accounts. The protocol raises cost but does not eliminate this.
- **AI agents on verified accounts**: Acceptable by design. The protocol verifies the human, not their tools. Platforms enforce their own policies on agent usage.
- **Nation-state compulsion**: If a government controls the provider and the registry in their jurisdiction, the protocol cannot protect users. The defense is jurisdictional diversity.
- **Identity disputes**: The protocol cannot adjudicate who is the "real" owner of a contested identity. Courts are the final arbiter, same as real-world passport fraud.
- **Excluded populations**: People without government-issued IDs (~1.1 billion globally) cannot complete the identity verification process.
- **One person, one account (v1)**: No multi-identity support. Domestic abuse survivors, whistleblowers, and others who need separate identities must wait for the v2 exception process.

---

18. Versioning and Compatibility

18.1 Version Scheme

Protocol versions follow the format `HIP/{major}.{minor}`.

- **Major version**: Incremented on breaking changes.
- **Minor version**: Incremented on backwards-compatible additions.

18.2 Breaking Changes

The following constitute breaking changes (require major version increment):

- Changes to the HMAC subject ID derivation formula.
- Changes to the time-based score decay curve.
- Changes to the JWS format or signing algorithm.
- Removal of required fields from the verification request or response.
- Changes to the well-known endpoint path.

18.3 Backwards Compatibility

A provider supporting HIP/1.x MUST accept HIP/1.0 requests. New optional fields MAY be added to the verification response without incrementing the major version. Platforms MUST ignore unknown fields in the response payload.

---

19. IANA Considerations

This protocol uses the well-known URI `/.well-known/hip/` as defined in this specification. Registration with IANA per RFC 8615 will be pursued prior to the protocol reaching stable status.

---

20. Signup Code Exchange

Signup codes provide a user-initiated mechanism for platforms to verify users without requiring the user to share their full HIP identifier in advance. This flow is useful for manual signup processes where users enter a code on a platform's website.

20.1 Signup Code Format

A signup code has the form:

```
{code}@id.{provider_domain}
```

- `code` is a 9-character lowercase string from the 31-character alphabet `a–z` + `2–9` (excluding ambiguous characters `0`, `1`, `i`, `l`, `o`).
- `id.` is the same namespace prefix used in subject IDs.
- `provider_domain` is the provider's registry-assigned domain.

Example: `vgd847m6p@id.humanidentity.io`

The code format mimics the subject ID format so users can easily understand where to enter it.

20.2 Code Lifecycle

Property	Value
Generation	User-initiated via provider portal
Validity	Configurable, default 1 hour
Usage	Single-use (consumed on exchange)
Max Active	Configurable per user, default 5

A provider MUST:

- Require user authentication to generate codes.
- Require the user to be in `active` status with a valid certificate.
- Allow users to revoke active codes before expiry.
- Delete codes immediately upon successful exchange.

20.3 Exchange Endpoint

```
POST https://{provider_domain}/.well-known/hip/exchange
```

Request Format:

```
POST /.well-known/hip/exchange
Content-Type: application/json
Authorization: Bearer {platform_api_key}
{
  "signup_code": "vgd847m6p",
  "nonce":       "string"
}
```

Field	Type	Required	Description
`signup_code`	string	REQUIRED	The 9-character signup code (without the `@id.{provider}` suffix).
`nonce`	string	REQUIRED	A cryptographically random string (minimum 16 characters, maximum 128 characters) for replay prevention.

Response Format:

The response is identical to the standard verification response (Section 6.3): a JWS compact serialization containing the user's status, score, and derived subject ID for the requesting platform.

Error Responses:

HTTP Status	Error Code	Description
400	`invalid_code`	Code format is invalid, does not exist, has expired, or has already been consumed. Consumed and expired codes MUST NOT be distinguishable to the caller.
401	`unauthorized`	Platform authentication failed
409	`nonce_reused`	Nonce was already used (replay detected)

20.4 Security Considerations

- Codes MUST be generated with sufficient entropy (at least 40 bits).
- Codes MUST NOT be logged or stored in plaintext after generation.
- Rate limiting MUST be applied to prevent enumeration attacks.
- Expired codes MUST be cleaned up promptly to reduce storage.

---

21. OAuth Authorization Flow

The OAuth authorization flow provides a browser-based mechanism for platforms to verify users. This is similar to "Login with Google" but returns a verification attestation instead of user profile data.

21.1 Flow Overview

```
Platform                    User Browser                    Provider
   |                             |                             |
   |----(1) Redirect------------>|                             |
   |                             |----(2) GET /oauth/authorize->|
   |                             |<---(3) Consent Page----------|
   |                             |----(4) POST (allow/deny)---->|
   |<---(5) Redirect with code---|                             |
   |----(6) POST /oauth/token---------------------------------->|
   |<---(7) Verification Response----------------------------|
```

21.2 Authorization Endpoint

```
GET https://{provider_domain}/oauth/authorize
```

Query Parameters:

Parameter	Required	Description
`client_id`	REQUIRED	The platform's canonical platform ID
`redirect_uri`	REQUIRED	Must match one of the platform's registered redirect URIs
`state`	RECOMMENDED	Opaque value for CSRF protection
`response_type`	OPTIONAL	Must be `code` if provided (default)

User Consent:

The provider MUST display a consent page showing:

- The platform's name (from registry)
- What information will be shared: verification status and score
- What will NOT be shared: personal information, email, documents
- Clear "Allow" and "Deny" options

Redirect on Allow:

```
{redirect_uri}?code={authorization_code}&state={state}
```

Redirect on Deny:

```
{redirect_uri}?error=access_denied&state={state}
```

21.3 Token Endpoint

```
POST https://{provider_domain}/oauth/token
Content-Type: application/json
Authorization: Bearer {platform_api_key}
{
  "grant_type": "authorization_code",
  "code":       "{authorization_code}"
}
```

Field	Type	Required	Description
`grant_type`	string	REQUIRED	Must be `authorization_code`
`code`	string	REQUIRED	The authorization code received from the redirect

Response:

The response is a verification attestation containing:

```
{
  "subject_id":   "xK7mN2pR9sT4vW6yBQw",
  "status":       "active",
  "score":        82,
  "score_state":  "stable",
  "attestation":  "{JWS compact serialization}",
  "issued_at":    "2026-01-15T12:00:00Z",
  "expires_at":   "2026-01-15T12:10:00Z"
}
```

The `attestation` field contains the standard JWS-signed verification response as defined in Section 6.3.

21.4 Authorization Code Properties

Property	Value
Format	Cryptographically random, minimum 32 bytes
Validity	5 minutes (OAUTH_CODE_TTL)
Usage	Single-use
Binding	Bound to specific platform + redirect_uri

21.5 Redirect URI Validation

Providers MUST validate that the `redirect_uri` exactly matches one of the platform's registered URIs. Wildcard subdomain matching MAY be supported for development purposes only and MUST require explicit opt-in.

21.6 Consent Persistence

A provider MAY remember user consent and skip the consent page on subsequent authorizations to the same platform. Users MUST be able to revoke consent through their account settings.

Appendix A: Score Decay Reference Table

Reference values for the time-based decay function at key intervals. All values rounded to the nearest integer.

Days Since Verification	Score
0	100
30	99
90	98
180	95
365	90
548 (1.5 years)	85
730 (2 years)	80
1095 (3 years)	70
1460 (4 years)	60
1825 (5 years)	50
2190 (6 years)	44
2555 (7 years)	38
2920 (8 years)	32
3285 (9 years)	26
3650 (10 years)	20 (floor)

Appendix B: Normalization Test Vectors

Name Normalization

Input	Normalized	SHA-256 (hex)
`" Jean-Pierre O'Brien "`	`"jean pierre obrien"`	`616ae47fe12dd44c71061240bf7257ac9397d71927f52c0a04c6a01cbd1180c8`
`"María García-López"`	`"maria garcia lopez"`	`7864ab7f883671f6ea34b918d967c5818e2e28992433514883b8b76bf0c5c1fa`
`"John Smith"`	`"john smith"`	`32ddaf65cc3aa8d3e6eda3ca2da7c18b71e169e9aa444cccb479c9ca759dd095`

Date Normalization

Input	Normalized	SHA-256 (hex)
`"1990-01-15"`	`"19900115"`	`4747c382bedef489a190a6797e6f4451907b86511bdd49cfa8f9d4c1a78d8bac`
`"1990/01/15"`	`"19900115"`	(same as above)

Note: Non-ISO date formats (e.g., `"15.01.1990"`) MUST be converted to ISO 8601 before normalization. Providers MUST NOT normalize dates by simply stripping non-digit characters from non-ISO formats, as this produces incorrect digit ordering.

Document ID Normalization

Input	Normalized	SHA-256 (hex)
`"AB-123.456"`	`"ab123456"`	`595a92a9ef887d8f780cb5d77f1a863c3cadad1e1bad06e77adeb3dad8b8e809`
`"ab 123 456"`	`"ab123456"`	(same as above)

All SHA-256 values are computed over the normalized UTF-8 byte sequence. Implementers MUST reproduce these exact digests to confirm their normalization logic is correct.

Appendix C: Platform Verification Sequence Diagram