Individual Submission
Internet-Draft
Intended status: Informational
Expires: 20 October 2026

G. Valverde
Zentity

18 April 2026

VEIL: Verified Ephemeral Identity Layer for OAuth 2.1
draft-valverde-oauth-veil-00

Abstract

VEIL (Verified Ephemeral Identity Layer) is a security profile of OAuth 2.1 for privacy-preserving identity verification. It separates claims into two tracks: proof claims (boolean verification results, compliance flags, assurance levels) travel through standard token claims, while identity claims (name, date of birth, address, nationality) travel only through an ephemeral, single-consume channel that keeps personally identifiable information off long-lived tokens. Subject identifiers are pairwise by default. Consent records are HMAC-protected. Step-up authentication scales with operation sensitivity. VEIL is the base profile for domain-specific extensions.

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 20 October 2026.

Copyright Notice

Table of Contents

1. Introduction

Identity verification produces two kinds of outputs: attestation (boolean results, compliance scores, assurance tiers) and identity data (name, date of birth, address, document details). OpenID Connect Core 1.0 [OIDC-Core] delivers both through the same channel, with the same lifetime and the same correlation properties. VEIL separates them: attestation is carried in standard token claims, while identity data is delivered through an ephemeral, single-consume channel bound to a user-initiated disclosure intent.

The profile composes and constrains the following specifications:

Concern	Specifications
Authorization Framework	OAuth 2.1 [I-D.ietf-oauth-v2-1]
Proof Key	PKCE [RFC7636], mandatory
Pushed Authorization	PAR [RFC9126], mandatory
Sender Constraining	DPoP [RFC9449]
Identity Layer	OpenID Connect Core 1.0 [OIDC-Core]
Structured Intent	Rich Authorization Requests [RFC9396]
Token Exchange	[RFC8693]
Token Introspection	[RFC7662]
Back-Channel Logout	[OIDC-BCL]

Table 1

This document specifies: a two-track claim model (Section 5), an ephemeral identity delivery channel (Section 6), pairwise subject identifiers as the default (Section 4), an abstract interface for assurance level derivation (Section 7), HMAC-verified consent integrity (Section 9), and an extension profile architecture for domain-specific profiles (Section 12).

2. Conventions and Terminology

2.1. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.2. Versioning

VEIL uses MAJOR.MINOR versioning, with -draft appended for pre-release revisions. Implementations MUST reject configurations or discovery documents with an unrecognized major version. Implementations SHOULD accept documents with a higher minor version than expected and ignore unrecognized fields, preserving forward compatibility.

2.3. Definitions

Assurance Level: A tiered classification of how thoroughly a user's identity has been verified. The profile defines the interface (inputs and outputs) but not the specific checks or tier definitions.

Ephemeral Store: An in-memory, TTL-bounded storage mechanism for identity claims with single-consume semantics. PII staged in the ephemeral store is consumed on first retrieval and automatically expires if unclaimed.

Identity Claim: A claim that contains personally identifiable information (name, date of birth, address, nationality, document details). Identity claims require explicit user action to release and flow only through the ephemeral store.

Proof Claim: A claim that conveys a boolean verification result, assurance level, or compliance flag without revealing the underlying PII. Proof claims flow through standard token claims and the userinfo endpoint.

Proof Scope: An OAuth scope that authorizes the release of proof claims. Proof scopes MUST NOT trigger identity claim delivery.

Identity Scope: An OAuth scope that authorizes the release of identity claims. Identity scopes require user action to unlock PII and flow through the ephemeral delivery channel.

Sector Identifier: A stable client-bound identifier used as input to pairwise subject derivation. In OIDC this is the registered sector identifier for the client; absent an explicit sector identifier, the chosen value SHOULD remain stable across redirect URI reordering.

Step-Up Authentication: A flow where the authorization server enforces a higher assurance requirement (acr_values) or session freshness (max_age) than the user's current session provides, triggering re-verification or re-authentication.

3. Transport Constraints

The following OAuth 2.1 options are mandatory under this profile. Requirements that OAuth 2.1 states as SHOULD or RECOMMENDED are restated here as MUST.

3.1. Authorization Server Requirements

The authorization server MUST:

* Support OAuth 2.1 [I-D.ietf-oauth-v2-1] authorization code flow with PKCE.

* Require Pushed Authorization Requests (PAR, [RFC9126]) for all authorization requests.

* Support DPoP [RFC9449] for sender-constrained tokens.

* Issue access tokens signed with EdDSA (Ed25519) for compact signatures.

* Support id_token_signed_response_alg client metadata for per-client signing algorithm selection.

The authorization server SHOULD:

* Support multiple signing algorithms for id_tokens (RS256, ES256, EdDSA, ML-DSA-65).

* Support Rich Authorization Requests [RFC9396] for structured intent.

* Support Token Exchange [RFC8693] for audience rebinding and scope attenuation.

* Support Back-Channel Logout [OIDC-BCL] for federated session termination.

3.2. Client Requirements

Clients MUST:

* Use PKCE with S256 challenge method.

* Submit authorization parameters via PAR before initiating the authorization flow.

* Include the resource parameter [RFC8707] on PAR requests.

* Support DPoP for sender-constrained token usage.

Clients MUST NOT:

* Use the implicit grant.

* Use the resource owner password credentials grant.

4. Subject Unlinkability

Under this profile, pairwise subject identifiers are the default. Unless a client explicitly registers subject_type: "public", each relying party receives a distinct pseudonym for the same user, preventing cross-RP correlation through the subject identifier.

4.1. Default Posture

All client registrations MUST default to subject_type: "pairwise". Clients that require globally stable subject identifiers explicitly opt in via subject_type: "public" at registration.

4.2. Derivation

Pairwise subject identifiers are derived as:

sub = HMAC-SHA-256(PAIRWISE_SECRET, sector + "." + userId)

Where:

* PAIRWISE_SECRET is a server-side secret of at least 32 bytes.

* sector is the client's stable sector identifier. If the authorization server supports OIDC sector_identifier_uri, it MUST use that registered sector identifier. Otherwise it MUST derive a stable client-bound value that does not change when redirect URI order changes.

* userId is the internal user identifier.

Two clients with different sector identifiers receiving tokens for the same user see different sub values. Neither client can derive the other's sub or determine that both values refer to the same user.

4.3. Token Exchange

When performing Token Exchange [RFC8693] with an audience parameter, the authorization server MUST re-derive the pairwise sub for the target audience's sector identifier. The exchanged token's sub MUST differ from the subject token's sub unless both clients share the same sector.

4.4. Extension Point

Extension profiles MAY apply pairwise derivation to additional token claims. PACT [I-D.valverde-oauth-pact] extends pairwise derivation to the act.sub claim for agent session identifiers.

5. Claim Separation

VEIL divides claims into two tracks based on whether they contain personally identifiable information. The separation is enforced at three independent layers: scope definitions (Section 5.2), consent decisions (Section 5.3), and delivery mechanisms (Section 6).

5.1. The Two Tracks

Proof claims convey verification results without PII. Examples: age_verification: true, nationality_verified: true, verification_level: "full", sybil_resistant: true. Proof claims are derived from cryptographic artifacts (ZK proofs, signed claims, encrypted attributes) rather than raw identity data.

Identity claims contain PII. Examples: name: "Jane Doe", date_of_birth: "1990-05-15", address: {...}, nationality: "FR". Identity claims require explicit user action to unlock and flow through the ephemeral delivery channel (Section 6).

The distinction between tracks is not what the claim describes but what it reveals. An age verification proof and a date of birth both describe the user's age, but the proof reveals only a boolean while the date reveals the value itself.

5.2. Scope Families

The authorization server MUST define two disjoint scope families:

Proof scopes (e.g., proof:age, proof:nationality, proof:verification, proof:compliance):

* Authorize the release of proof claims only.

* MUST NOT trigger identity claim delivery.

* Do not require vault unlock or any user action beyond initial consent.

* May be auto-approved on subsequent requests if consent already exists.

Identity scopes (e.g., identity.name, identity.address, identity.nationality, identity.dob):

* Authorize the release of identity claims.

* MUST trigger the ephemeral delivery channel (Section 6).

* Require explicit user action to unlock PII for each authorization.

* MUST NOT be persisted in durable consent records (ensuring the unlock prompt reappears on each request).

An umbrella proof scope (e.g., proof:identity) MAY expand to all proof sub-scopes at consent time, allowing the user to opt in per sub-scope.

5.3. Delivery Channels

The two tracks produce different delivery behavior at each OAuth endpoint:

In id_tokens: Proof claims MAY be embedded directly. Identity claims MUST NOT be embedded in id_tokens. Identity scopes unlock PII only through the ephemeral delivery channel and userinfo consumption path described in Section 6.

In access tokens: Access tokens ordinarily carry only structural claims (sub, scope, aud, cnf, and extension-defined claims like act). Proof claims are not embedded there except where a profile explicitly defines an access-token-only proof artifact. VEIL defines one such exception in Section 5.4: proof:sybil yields a per-RP sybil_nullifier in access tokens only. Identity claims MUST NOT be embedded in access tokens.

Via userinfo: Both proof claims and identity claims are delivered. Proof claims are resolved from the user’s verification state. Identity claims are consumed from the ephemeral store (single-consume; the entry is deleted after retrieval).

5.4. Sybil Nullifiers

The profile defines an optional proof:sybil scope that produces a per-RP unlinkable nullifier:

```
sybil_nullifier = HMAC-SHA-256(
    DEDUP_HMAC_SECRET,
    dedupKey + ":" + clientId
)
```

The same person receives the same nullifier at one RP but different nullifiers at different RPs. This enables per-human rate limiting and duplicate detection without cross-RP identity linkage.

6. Transient Disclosure

Identity claims (PII) reach the relying party through an in-memory store with single-consume semantics, bound to a user-initiated intent flow. The profile mandates the delivery semantics; the unlock mechanism (passkey PRF, password-derived key, wallet signature) is implementation-specific.

6.1. Single-Consume Properties

When identity scopes are approved, the authorization server stages PII in an in-memory store with the following properties:

Property	Requirement
Storage	In-memory only; MUST NOT be written to disk or database.
Key	Implementation-defined. It MUST either include a request-specific handle or otherwise guarantee that only one live staged entry exists per userId:clientId.
TTL	Configurable; SHOULD default to 5 minutes for interactive flows.
Consumption	Single-consume; the entry MUST be deleted on first retrieval.
Concurrency	If the key does not include a request-specific handle, concurrent staging for the same userId:clientId MUST be rejected until the earlier entry is consumed, cleared, or expires.
Replay protection	Intent JTIs MUST be persisted in durable storage (not memory-only) and checked to prevent replay of unlock events. Persistence MUST use insert-or-ignore semantics to handle concurrent first-use races.
Rate limiting	Intent and stage endpoints MUST be rate-limited per user identity to prevent brute-force unlock attempts and ephemeral store flooding.
Ambiguity safety	If multiple entries exist for the same userId when the consuming endpoint lacks clientId context, the server MUST return no claims rather than risk cross-client PII misdelivery.

Table 2

6.2. Intent Flow

The staging process follows an intent-then-stage pattern to ensure that PII staging is bound to a specific user action and cannot be replayed or triggered by the relying party alone:

1. *User action.* The user performs a credential-specific unlock action (the mechanism is implementation-specific).

2. *Intent token.* The authorization server issues a signed intent token (HMAC-SHA-256, short TTL) carrying a scope hash and, for backchannel flows, the authorization request identifier that binds the unlock to a specific consent request.

3. *Stage.* The client presents the intent token. The server validates it, filters the user's identity data by the approved scopes, and stages the filtered claims in the ephemeral store.

4. *Consume.* The relying party calls the userinfo endpoint. The server retrieves and deletes the staged entry in a single atomic operation.

6.3. Exclusions

The ephemeral store MUST NOT contain:

* Raw biometric data (images, embeddings, liveness scores).

* Document images.

* Cryptographic key material.

* Internal identifiers that could be used to correlate across RPs.

Only the identity claims authorized by the approved identity scopes are staged.

7. Tiered Verification

VEIL defines the interface for assurance level derivation (inputs, outputs, properties) but leaves the verification pipeline, check definitions, and tier labels to each implementation.

7.1. Abstract Interface

Input: A set of verification artifacts. The profile does not mandate the artifact types, but common examples include zero-knowledge proof verification results, signed claims from verification processes, encrypted attribute presence flags, biometric match results, and sybil resistance indicators.

Output: An assurance result containing:

Field	Type	Description
verified	boolean	Whether the user meets the minimum verification threshold.
level	string	Tiered assurance level (implementation-defined labels).
numericLevel	integer	Numeric ordering of tiers for comparison.
checks	object	Individual boolean check results (implementation-defined).

Table 3

7.2. Derivation Requirements

The derivation function MUST be:

* Pure. No database access, no side effects. All inputs are passed explicitly.

* Deterministic. The same inputs always produce the same output.

* Monotonic in tier ordering. A higher numeric level MUST satisfy any requirement for a lower level.

The authorization server MUST expose the assurance level through:

* The acr claim in id_tokens (mapping implementation tiers to URN identifiers).

* Proof claims filtered by granted proof scopes.

7.3. Step-Up Enforcement

Step-up authentication is _spatial_ (trust depends on which operation is attempted) rather than _temporal_ (trust deepening over time). Two enforcement points apply:

When a client includes acr_values in an authorization request, the authorization server MUST verify that the user's current assurance level satisfies the requested tier. If it does not, the server MUST return an explicit step-up error rather than silently downgrading. In browser-based authorization flows, the error SHOULD be surfaced as

interaction_required; extension profiles MAY define an equivalent continuation-oriented error if they also provide a resumable step-up path.

When a client includes max_age, the authorization server MUST verify that the user's session is not older than the specified value. If it is, the server MUST require re-authentication. In PAR-based flows, the PAR record SHOULD be preserved with an extended TTL to cover the re-authentication round-trip, so the authorization flow can resume after login. When acr_values cannot be satisfied in a browser-based PAR flow, the PAR record MUST be deleted and the server MUST redirect with the same step-up error contract used for other authorization flows (for example, interaction_required).

ACR identifiers are deployment-local. Consistent with OIDC Core 1.0 Section 2 and established IETF practice (e.g., urn:mace:incommon:iap:silver), a conforming authorization server MUST document the ACR URNs it issues and their tier semantics in its discovery metadata (acr_values_supported). VEIL does not register specific ACR values and does not reserve a VEIL-specific ACR namespace; interoperability is achieved by publishing the local URNs rather than by centralized registration.

Extension profiles MAY define additional step-up enforcement points.

8. Signing Agility

8.1. Algorithm Selection

The authorization server MUST support at least RS256 for id_tokens (OIDC Discovery 1.0 Section 3 mandatory). The authorization server SHOULD support ES256, EdDSA, and ML-DSA-65.

Access tokens MUST be signed with EdDSA (compact 64-byte signatures for Bearer headers).

Clients MAY declare a preferred id_token signing algorithm via id_token_signed_response_alg in their registration metadata. The authorization server MUST honor the declared preference if it supports the algorithm.

8.2. Key Management

Signing keys MUST be encrypted at rest (SHOULD use AES-256-GCM with a dedicated key encryption key).

Key rotation MUST support an overlap window during which both the retiring and replacement keys are served by the JWKS endpoint. The retiring key carries an expiresAt timestamp; after the overlap window, it is removed.

9. Tamper-Evident Consent

An attacker who can modify stored consent scopes can bypass both the two-track model and ephemeral delivery. Consent records are protected by an HMAC (Section 9.1), and identity scopes are excluded from durable consent storage (Section 9.3).

9.1. Scope Verification

The authorization server MUST verify the integrity of stored consent records on every authorization request. Consent records MUST be protected by an HMAC computed over the consent context:

```
scope_hmac = HMAC-SHA-256(SECRET, length_prefix(context) ||
  length_prefix(userId) || length_prefix(clientId) ||
  length_prefix(referenceId) || length_prefix(sorted_scopes))
```

Length-prefixed encoding prevents concatenation collisions. The HMAC key MUST be derived from the server's base secret via a KDF (e.g., HKDF [RFC5869]) with a domain-separating info parameter, not used as a raw key. This separation ensures that the consent integrity key and the session authentication key are cryptographically independent even when derived from the same root secret.

9.2. Tamper Response

Before processing an authorization request against stored consent, the server MUST recompute the HMAC and compare it to the stored value. If they differ, the server MUST delete the consent record and require re-consent. This prevents three attack vectors: direct database scope escalation, consent replay across clients, and consent replay across users.

9.3. Identity Scope Stripping

Identity scopes MUST NOT be persisted in durable consent records. The full scope set (including identity scopes) is written for authorization code derivation, then identity scopes are stripped immediately so that the consent page reappears on subsequent requests. This ensures that PII delivery requires a fresh user decision each time.

10. Federated Session Termination

10.1. Back-Channel Logout

The authorization server MUST support OIDC Back-Channel Logout [OIDC-BCL] for federated session termination. On user sign-out:

1. Query all registered clients with a backchannel_logout_uri.
2. For each: build a logout token JWT with the client's pairwise sub and sid (if backchannel_logout_session_required).
3. POST the logout token to the client's endpoint.
4. Retry with exponential backoff on transient failures.

10.2. Extension Coordination

Extension profiles MAY define additional cleanup actions triggered by logout.

11. Machine-Readable Surfaces

11.1. OIDC Discovery Extensions

The authorization server's OpenID Provider Configuration (/.well-known/openid-configuration) MUST include:

* subject_types_supported: ["pairwise", "public"] with pairwise listed first.
* acr_values_supported listing all supported assurance tier URNs.
* scopes_supported listing all proof and identity scopes.
* backchannel_logout_supported: true.
* backchannel_logout_session_supported: true.
* require_pushed_authorization_requests: true.
* dpop_signing_alg_values_supported.
* id_token_signing_alg_values_supported (at minimum ["RS256"]).

11.2. Extension Discovery

Extension profiles define their own discovery documents at dedicated well-known paths. The OIDC Discovery document SHOULD NOT be overloaded with extension-specific metadata.

12. Compositional Extension

Domain-specific profiles (agent delegation, verifiable credentials, regulatory compliance) extend VEIL through the mechanism defined below. Extensions MUST NOT modify VEIL itself.

12.1. Architecture

Each extension profile:

* References VEIL as its base and inherits all VEIL requirements.
* Adds domain-specific capabilities without modifying VEIL's core requirements.
* Defines its own discovery document at a dedicated well-known path.
* Specifies additional conformance requirements for its domain.

12.2. Extension Boundaries

Extension profiles MUST NOT:

* Weaken pairwise subject requirements (e.g., defaulting to public subjects).
* Bypass the ephemeral delivery channel for identity claims.
* Embed identity claims in access tokens or extension-specific token types.
* Modify the consent integrity mechanism.

Extension profiles MAY:

* Define additional pairwise derivations.
* Define additional step-up enforcement points.
* Define additional token types via Token Exchange [RFC8693].
* Define additional consent routing logic.

* Extend the ephemeral store TTL for their specific flow patterns.

13. Security Considerations

Implementations MUST satisfy the security considerations of each composed specification: OAuth 2.1 [I-D.ietf-oauth-v2-1], PKCE [RFC7636], PAR [RFC9126], DPoP [RFC9449], RAR [RFC9396], Token Exchange [RFC8693], OIDC Core 1.0 [OIDC-Core], and OIDC Back-Channel Logout 1.0 [OIDC-BCL].

13.1. PII Exposure Window

Identity claims exist in cleartext only during the ephemeral delivery window (between staging and consumption). Outside that window, identity data is either encrypted (credential-wrapped secrets, FHE ciphertexts) or absent.

Authorization servers MUST NOT log identity claim values. Authorization servers MUST NOT cache identity claims beyond the ephemeral store TTL.

13.2. Correlation Resistance

Pairwise subject identifiers prevent cross-RP user correlation under the standard OIDC threat model. However, timing correlation (two RPs comparing token issuance timestamps) and metadata correlation (user agent, IP address) remain possible. The authorization server SHOULD minimize metadata leakage in token responses and SHOULD NOT include client-identifying information in error responses visible to other clients.

13.3. Sender Constraining

DPoP sender-constraining prevents token theft and replay. When combined with pairwise subjects, a stolen token is useless to a different RP (wrong aud) and useless to a different client (wrong cnf.jkt).

14. Privacy Considerations

Implementations MUST satisfy the privacy considerations of each composed specification. Participants are the authorization server (AS), the relying party (RP), and the end user. Data categories are proof claims (Section 5), pairwise subjects (Section 4), sybil nullifiers (Section 5.4), and identity claims (PII), which flow only through the ephemeral channel (Section 6). Cross-RP correlation through subject identifiers requires explicit opt-in at registration (subject_type: "public"); identity disclosure requires explicit opt-

in at authorization (identity scopes, per-scope user consent).

14.1. Double Anonymity

The profile supports a double-anonymity mode where both sides of the identity verification relationship are protected. The relying party cannot identify the user or recognize returning users (pairwise subjects, no email scope), and the authorization server cannot reconstruct which services the user visits (consent records do not accumulate when identity scopes are stripped).

14.2. Sybil Nullifier Privacy

The sybil_nullifier (Section 5.4) is derived per-RP. Two RPs receiving nullifiers for the same user cannot determine that fact. The nullifier enables per-human uniqueness enforcement without cross-RP identity linkage.

14.3. Session Metadata Nullification

The authorization server MUST NOT persist the user’s IP address or user agent in session records. If the session layer collects these values (as most frameworks do by default), they MUST be nullified before storage. Without this, two RPs that obtain session metadata (via breach, legal compulsion, or insider access) can correlate users by matching IP + user agent + timestamp patterns.

14.4. Provider Fingerprint Removal

Four concrete signals can fingerprint the authorization server and enable cross-RP provider correlation:

1. Provider identifiers in proof claims. Proof claim payloads MUST NOT include provider-identifying fields (e.g., issuer_id). Provider identity is an internal audit concern.

2. Trust framework naming. The trust framework identifier in identity assurance claims SHOULD identify the regulatory framework (e.g., eidas), not the provider name.

3. Credential type identifiers. Verifiable credential type identifiers (VCT) MUST use provider-neutral URNs (e.g., urn:credential:identity-verification:v1) rather than provider-domain URLs.

4. Structural fingerprinting. SD-JWT credentials SHOULD include decoy disclosures to prevent fingerprinting via disclosed claim set analysis.

14.5. Claim Specificity

Proof claims are derived from cryptographic artifacts, not from raw PII. The authorization server SHOULD minimize the specificity of proof claims. For example, nationality_group: "EU" is preferable to nationality: "FR" when the relying party's requirement is jurisdiction membership rather than specific nationality.

15. IANA Considerations

This document has no immediate IANA actions that require new registry creation. It relies on identifiers and registries that are either already allocated or that are delegated to extension profiles.

15.1. OAuth Parameter Registries

VEIL does not register new OAuth parameters, grant types, token type URIs, client metadata fields, or authorization server metadata fields. It narrows existing OAuth 2.1 and OIDC Core 1.0 parameters to a mandatory subset (see Section 3). Extension profiles that build on VEIL are responsible for registering any new OAuth identifiers they introduce.

15.2. Well-Known URI Registrations

VEIL relies on the existing well-known URI registrations openid-configuration (OIDC Discovery 1.0) and oauth-authorization-server [RFC8414]. It does not register a new well-known URI. Extension profiles MAY register their own well-known URI suffixes per [RFC8615].

15.3. Assurance Certification Values (acr)

ACR URNs listed in acr_values_supported are deployment-local, consistent with OIDC Core 1.0 Section 2 and established IETF practice. VEIL does not register a VEIL-specific ACR namespace and does not request a registry; each authorization server documents its own ACR URNs and their tier mapping in its discovery metadata (see Section 7.3).

15.4. Verifiable Credential Type Identifiers

Where a deployment emits verifiable credentials, credential type identifiers (VCT) MUST be provider-neutral URNs (see Section 14.4). VEIL does not define or reserve specific VCT URNs; deployments choose values consistent with the credential ecosystem they participate in.

16. Conformance

16.1. Authorization Server Requirements

A conforming authorization server MUST:

* Implement the OAuth 2.1 baseline (Section 3).
* Default to pairwise subject identifiers (Section 4).
* Implement the two-track claim model (Section 5).
* Implement ephemeral identity delivery with single-consume semantics (Section 6).
* Implement an assurance level derivation function that satisfies the abstract interface (Section 7).
* Enforce step-up authentication for acr_values and max_age (Section 7.3).
* Protect consent records with HMAC integrity verification (Section 9).
* Strip identity scopes from persisted consent (Section 9.3).
* Support back-channel logout ([OIDC-BCL]).
* Publish OIDC Discovery metadata.

16.2. Client Requirements

A conforming client MUST:

* Use PKCE with S256 and PAR for all authorization requests.
* Include the resource parameter on PAR requests.
* Support DPoP for token binding.
* Consume identity claims from the userinfo endpoint, not from id_token claims alone.
* Treat userinfo identity claim delivery as single-consume (do not retry expecting the same data).

17. References

17.1. Normative References

[I-D.ietf-oauth-v2-1]
Hardt, D., Parecki, A., and T. Lodderstedt, "The OAuth 2.1 Authorization Framework", Work in Progress, Internet-Draft, draft-ietf-oauth-v2-1-15, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-v2-1-15>.

[OIDC-BCL] OpenID Foundation, "OpenID Connect Back-Channel Logout 1.0 incorporating errata set 1", 14 September 2022, <https://openid.net/specs/openid-connect-backchannel-1_0.html>.

[OIDC-Core]
OpenID Foundation, "OpenID Connect Core 1.0 incorporating errata set 2", 15 December 2023, <https://openid.net/specs/openid-connect-core-1_0.html>.

[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>.

[RFC7636] Sakimura, N., Ed., Bradley, J., and N. Agarwal, "Proof Key for Code Exchange by OAuth Public Clients", RFC 7636, DOI 10.17487/RFC7636, September 2015, <https://www.rfc-editor.org/rfc/rfc7636>.

[RFC7662] Richer, J., Ed., "OAuth 2.0 Token Introspection", RFC 7662, DOI 10.17487/RFC7662, October 2015, <https://www.rfc-editor.org/rfc/rfc7662>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8693] Jones, M., Nadalin, A., Campbell, B., Ed., Bradley, J., and C. Mortimore, "OAuth 2.0 Token Exchange", RFC 8693, DOI 10.17487/RFC8693, January 2020, <https://www.rfc-editor.org/rfc/rfc8693>.

[RFC9126] Lodderstedt, T., Campbell, B., Sakimura, N., Tonge, D., and F. Skokan, "OAuth 2.0 Pushed Authorization Requests", RFC 9126, DOI 10.17487/RFC9126, September 2021, <https://www.rfc-editor.org/rfc/rfc9126>.

[RFC9396] Lodderstedt, T., Richer, J., and B. Campbell, "OAuth 2.0 Rich Authorization Requests", RFC 9396, DOI 10.17487/RFC9396, May 2023, <https://www.rfc-editor.org/rfc/rfc9396>.

[RFC9449] Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., and D. Waite, "OAuth 2.0 Demonstrating Proof of Possession (DPoP)", RFC 9449, DOI 10.17487/RFC9449, September 2023, <https://www.rfc-editor.org/rfc/rfc9449>.

17.2. Informative References

[FAPI-2] OpenID Foundation, "FAPI 2.0 Security Profile", n.d., <https://openid.net/specs/fapi-2_0-security-profile.html>.

[HAIP] OpenID Foundation, "OpenID4VC High Assurance Interoperability Profile 1.0", n.d., <https://openid.net/specs/openid4vc-high-assurance-interoperability-profile-1_0.html>.

[I-D.valverde-oauth-pact] Valverde, G., "PACT: Private Agent Consent and Trust Profile", 2026.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/rfc/rfc5869>.

[RFC8414] Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Authorization Server Metadata", RFC 8414, DOI 10.17487/RFC8414, June 2018, <https://www.rfc-editor.org/rfc/rfc8414>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/rfc/rfc8615>.

[RFC8707] Campbell, B., Bradley, J., and H. Tschofenig, "Resource Indicators for OAuth 2.0", RFC 8707, DOI 10.17487/RFC8707, February 2020, <https://www.rfc-editor.org/rfc/rfc8707>.

Acknowledgments

The author thanks the OAuth working group and the OpenID Foundation for the foundational specifications on which this profile composes, and the FAPI 2.0 [FAPI-2] and HAIP [HAIP] working groups whose security-profile patterns informed this document’s structure.

Author’s Address

Gustavo Valverde
Zentity
Portugal
Email: g.valverde02@gmail.com