

Web Authorization Protocol
Internet-Draft
Intended status: Standards Track
Expires: 20 April 2026

S. C. Mora
P. Dingle
Microsoft Corporation
17 October 2025

OAuth 2.0 Entity Profiles
draft-mora-oauth-entity-profiles-00

Abstract

This specification introduces Entity Profiles as a mechanism to categorize OAuth 2.0 entities—clients and subjects—based on their operational context. Entity Profiles provide structured descriptors for the client initiating the OAuth flow and the subject represented in tokens. This document defines new JWT Claim names and metadata parameters for use in access tokens, ID tokens, token introspection responses, dynamic client registration, and Authorization Server metadata.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at <https://drafts.srey.io/draft-mora-oauth-entity-profiles.html>. Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-mora-oauth-entity-profiles/>.

Discussion of this document takes place on the Web Authorization Protocol Working Group mailing list (<mailto:oauth@ietf.org>), which is archived at <https://mailarchive.ietf.org/arch/browse/oauth/>. Subscribe at <https://www.ietf.org/mailman/listinfo/oauth/>.

Source for this draft and an issue tracker can be found at <https://github.com/Sreyanth/draft-mora-oauth-entity-profiles>.

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 April 2026.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	4
1.1. Conventions and Definitions	4
1.2. Terminology	4
2. Motivation and Use Cases	5
3. Entity Profiles	5
3.1. Standardized Entity Profiles	6
3.1.1. user	6
3.1.2. device	6
3.1.3. native_app	6
3.1.4. web_app	6
3.1.5. browser_app	7
3.1.6. service	7
3.1.7. ai_agent	7
3.2. Private or Custom Entity Profiles	8
3.3. Representation in Token Claims and Metadata	8
4. Entity Profile JWT Claims	9
4.1. client_profile Claim	9
4.2. sub_profile Claim	9
5. Authorization Server Metadata	10
6. Dynamic Client Registration Metadata	10
7. Token Introspection Response Parameters	11
8. Client Behavior	12
8.1. Registration	12
9. Authorization Server Behavior	12
9.1. Client Registration	12

9.2.	Subject Profile Assignment	13
9.3.	JWT Token Issuance	13
9.4.	Token Introspection Responses	14
9.5.	Validation Requirements	14
10.	Resource Server Behavior	15
10.1.	Handling Multiple Entity Profiles in Authorization Policies	16
11.	Delegation Considerations	17
12.	Security Considerations	17
12.1.	Trust and Verification	17
12.2.	Entity Profile Spoofing	17
12.3.	Consistency and Semantics	17
12.4.	Separation from Authentication and Assurance	18
12.5.	Layered Policy Enforcement	18
12.6.	Audit and Monitoring	18
12.7.	Default Handling	18
12.8.	Misclassification Risks	19
12.9.	Token Bloating	19
13.	Privacy Considerations	19
13.1.	Fingerprinting	19
13.2.	Profiling and Behavioral Inference	19
13.3.	Minimization	20
13.4.	Data Exposure	20
14.	IANA Considerations	20
14.1.	OAuth Entity Profiles Registry	20
14.1.1.	Registration Template	20
14.1.2.	Initial Registry Contents	21
14.2.	JWT Claims Registration	23
14.2.1.	client_profile Claim	23
14.2.2.	sub_profile Claim	23
14.3.	Authorization Server Metadata Registration	23
14.3.1.	entity_profiles_supported	23
14.4.	Dynamic Client Registration Metadata Registration	23
14.4.1.	client_profile	24
14.5.	Token Introspection Response Registration	24
14.5.1.	client_profile	24
14.5.2.	sub_profile	24
15.	References	24
15.1.	Normative References	24
15.2.	Informative References	25
Appendix A.	Example Usage in Various Flows and Use Cases	26
Appendix B.	Document History	27
B.1.	draft-00	27
Acknowledgments	27
Authors' Addresses	27

1. Introduction

This specification introduces a mechanism for classifying entities participating in OAuth 2.0 and OpenID Connect flows using standard or custom-defined "Entity Profiles." These profiles offer a structured way to describe the nature or operational context of clients and token subjects, enhancing authorization decisions, policy enforcement, risk assessment, and audit capabilities.

This specification introduces two new Claims:

- * `client_profile`: Describes the nature of the client software or application initiating the OAuth flow (e.g., web app, native app, AI agent).
- * `sub_profile`: Describes the entity represented by the subject (sub) Claim in an issued token (e.g., user, service, AI agent).

This specification establishes a registry for OAuth Entity Profiles and defines an initial set of Entity Profile values. This document also defines how the Entity Profiles can be used in access tokens, ID tokens, token introspection responses, dynamic client registration, and Authorization Server metadata.

1.1. Conventions and Definitions

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.

This specification uses the Augmented Backus-Naur Form (ABNF) notation of [RFC5234].

1.2. Terminology

This document uses the terms "User", "Resource Owner", "Client", "Authorization Server", "Resource Server", and "Access Token" defined in [RFC6749] and the terms "JSON Web Token (JWT)", "Claim", "Claim Name" defined in [RFC7519].

- * ***Entity Profile***: A string-based descriptor identifying the category of an OAuth entity (client or subject) based on its operational context.
- * ***Client Profile***: A descriptor for the client software or

application initiating the OAuth flow (e.g., web app, native app, AI agent etc.).

- * ***Subject Profile***: A descriptor for the entity represented by the token subject (e.g., user, AI agent, service account etc.).

2. Motivation and Use Cases

As OAuth 2.0 continues to be used in increasingly diverse environments—including cloud-native architectures, zero-trust systems, IoT ecosystems, and AI-driven platforms—the nature of entities participating in OAuth flows has become more heterogeneous.

Current OAuth deployments lack a standardized way to represent and reason about these differing entity types. The introduction of Entity Profiles helps address several practical needs:

- * ***Access Control and Policy Enforcement***: Resource Servers can apply different access policies depending on whether the caller is a user, service, device, or AI agent.
- * ***Risk Scoring and Security Posture***: Authorization Servers can assess and adjust risk and their security measures based on client type (e.g., public native app vs. backend service).
- * ***Auditing and Forensics***: Security logs become more useful and interpretable when entity types are made explicit.
- * ***Future-Proofing OAuth Flows***: As AI agents and autonomous systems take on greater roles, profiles offer a way to signal and manage their participation explicitly.
- * ***User Experience***: Customized interfaces and consent flows, such as enhanced disclosures and controls for human users and granular permissions for AI agents.

By introducing a consistent way to describe the operational context of clients and subjects, this specification enhances the overall authorization decisions, policy enforcement, risk assessment, and audit capabilities in OAuth 2.0 and OpenID Connect deployments.

3. Entity Profiles

3.1. Standardized Entity Profiles

The following is a list of Entity Profile values defined by this specification. These identifiers are not intended to be exhaustive, and additional profiles can be defined by implementers or through other specifications. The values defined herein are an intentionally small set that covers the most common scenarios.

3.1.1. user

A user is a human resource owner who interacts with the system through a client such as a web application, native application, user-agent-based application, or device interface. Users provide consent, make authorization decisions, and are typically assumed to have access to their own credentials, session tokens, and authorization settings on their devices. While users often act directly through a client, they may also delegate access to applications or AI agents acting on their behalf. It is assumed that users can make informed consent decisions, understand the implications of delegated access, and manage their own permissions within the system.

3.1.2. device

A device is a hardware-based computing entity (e.g., smart TVs, IoT sensors, mobile phones, kiosk terminals) that can host client applications. Devices may have limited user interfaces and may act autonomously or under user control. They are often constrained in security capabilities (e.g., lack of secure storage, no local display, indirect input methods) and typically require indirect authorization mechanisms such as the Device Authorization Grant [RFC8628]. Devices are generally considered public clients unless they can securely protect credentials.

Devices are distinct from backend services or virtualized compute instances (e.g., VMs, containers) based on their physical deployment and user-facing interaction model.

3.1.3. native_app

Same as "native application" defined in Section 2.1 of [RFC6749].

3.1.4. web_app

Same as "web application" defined in Section 2.1 of [RFC6749].

3.1.5. browser_app

Same as "user-agent-based application" defined in Section 2.1 of [RFC6749].

3.1.6. service

A service is a non-human actor that represents a logical service, backend process, or microservice within a distributed system. When acting as a client in OAuth flows, services are classified as confidential applications, meaning they can securely store credentials and access tokens. Unlike user-facing applications, services do not involve direct user interaction and typically operate within system-defined scopes and trust boundaries, often performing automated tasks or facilitating communication between components in the system.

3.1.7. ai_agent

An AI agent is an autonomous or semi-autonomous system capable of initiating actions and making decisions independently or on behalf of a user or organization. These agents may participate in OAuth flows as clients, subjects, or both. They might perform delegated tasks based on user instructions, operate persistently across sessions and services, possess their own identity, credentials, and entitlements. They are typically powered by machine learning or reasoning systems, exhibiting adaptive, context-aware, non-deterministic, and proactive behavior over extended sessions and across multiple systems.

AI agents can be deployed in various environments, such as backend services, cloud-based systems, edge devices, or within user-controlled platforms. Depending on their operational context and deployment, AI agents can be clients -- making outbound API calls and initiating OAuth flows; or, subjects -- acting as the resource owner in systems that authorize autonomous services. Depending on their ability to securely store credentials and maintain integrity, they may be classified as either confidential or public clients. In some contexts, even well-secured AI agents may be treated as public clients due to the inherent complexity and unpredictability of their behavior. While it can be difficult to objectively verify that an entity truly qualifies as an AI agent, trusted publishers or registries may attest to the nature of the entity.

3.2. Private or Custom Entity Profiles

Entity Profiles can be defined at will by those using this specification. However, in order to prevent collisions, any new Entity Profile should either be registered in the IANA OAuth Entity Profiles registry established by (Section 14.1) or be a collision-resistant value. The definer of the value needs to take reasonable precautions to ensure they are in control of the part of the namespace they use to define the Entity Profile.

3.3. Representation in Token Claims and Metadata

The value of an Entity Profile, whether appearing as a Claim in JWT tokens or as a parameter in metadata or introspection responses, is expressed as a space-delimited, case-insensitive list of strings. Each string represents a classification of the entity (either a client or a subject) and MUST adhere to the syntax defined below.

Multiple values MAY be included if the entity fits into more than one category. The order of values is insignificant, and there is no implicit hierarchy or relationship between the values. When multiple Entity Profiles are present, each value is interpreted independently. When including multiple values, the Entity Profile combinations SHOULD be semantically meaningful and validated against expected usage patterns. Conflicting profiles (e.g., user service) SHOULD be avoided unless a clear operational justification exists (e.g., user ai_agent to indicate an AI agent acting as a digital employee).

```
entity-profile = profile-token * ( SP profile-token )
profile-token = 1*( ALPHA / DIGIT / "-" / "_" / "." )
```

Examples:

```
* "user"
* "ai_agent"
* "service ai_agent"
```

All profile values MUST either:

- * Be registered in the OAuth Entity Profiles IANA registry (see Section 14.1), or
- * Be privately defined (see Section 3.2).

When processing these values, Authorization Servers and Resource Servers MUST NOT assume any implicit relationships or hierarchies between the Entity Profiles unless explicitly agreed upon by the Authorization Server and the Resource Server or if defined in a different specification.

For example, a token might include the Entity Profiles `ai_agent` and `acme_verified_robot`, where `acme_verified_robot` is a vendor-defined profile used to identify industrial automation clients from ACME Corp. These two profiles are independent — the presence of both does not imply that one is a subtype of the other, nor that they share privilege or trust levels. Treating `acme_verified_robot` as a subclass of `ai_agent`, or vice versa, could result in incorrect access decisions.

4. Entity Profile JWT Claims

This specification defines two new Claim Names: `client_profile` and `sub_profile`. These Claims may appear in access tokens (e.g., JWT access tokens) and OpenID Connect ID tokens.

4.1. `client_profile` Claim

The `client_profile` (Client Profile) Claim indicates the Entity Profile(s) of the Client identified by the `client_id` (Client ID) Claim in a JWT access token or ID token. If included, the value of this Claim MUST conform to the rules defined in Section 3.3. Use of this Claim is OPTIONAL.

4.2. `sub_profile` Claim

The `sub_profile` (Subject Profile) Claim indicates the Entity Profile(s) of the Subject represented by the `sub` (Subject) Claim in a JWT access token or ID token. If included, the value of this Claim MUST conform to the rules defined in Section 3.3. Use of this Claim is OPTIONAL.

Below is a non-normative example illustrating how the new Entity Profile Claims can appear within a JWT access token. Other standard Claims are omitted for brevity:

```
{
  "sub": "user123",
  "sub_profile": "user",
  "client_id": "client456",
  "client_profile": "service ai_agent"
}
```

5. Authorization Server Metadata

If an Authorization Server supports publishing metadata as defined in [RFC8414] and also implements the mechanisms defined in this specification, it SHOULD advertise its support for the mechanisms defined in this specification using the following metadata parameter in its Authorization Server Metadata document:

"entity_profiles_supported": OPTIONAL. JSON object containing two JSON arrays: client and subject. Each array lists the Entity Profiles supported by the Authorization Server. The client array SHOULD include all Entity Profiles that the server can issue in client_profile Claims, and the subject array SHOULD include all Entity Profiles that the server can issue in sub_profile Claims. Authorization Servers MAY omit some supported profiles from this metadata if desired.

Clients can use this metadata to determine which Entity Profiles the Authorization Server recognizes and to understand how their own Entity Profiles might be interpreted or classified.

Below is an example of how this metadata parameter might be included in the Authorization Server Metadata document. Other standard parameters are omitted for brevity:

```
{
  "entity_profiles_supported":
  {
    "client": ["native_app", "web_app", "browser_app", "service", "ai_agent"],
    "subject": ["user", "device", "service", "ai_agent"]
  }
}
```

6. Dynamic Client Registration Metadata

If an Authorization Server supports Dynamic Client Registration as defined in [RFC7591] and also implements the mechanisms defined in this specification, it SHOULD allow clients to declare their Entity Profile using the following metadata parameter in their registration requests:

"client_profile": OPTIONAL. The Entity Profile of the client. This value MAY be included in the client's registration request and, if present, SHOULD match the client's actual Entity Profile. If omitted, the Authorization Server MAY assign a default Entity Profile based on its policies or the client's other metadata.

If the provided `client_profile` does not match the server's policy or fails the server's validation and verification checks, the Authorization Server MUST reject the registration request with an `invalid_client_metadata` error. The verification mechanisms for the `client_profile` value are out of scope for this specification, but it is recommended that Authorization Servers implement appropriate checks based on their security policies and operational context. They may also employ verification strategies like checking against known registries, validating against trusted sources, admin approval, metadata inference, out-of-band attestation by an accountable party etc.

Authorization Servers MAY choose to ignore all or some of the provided Entity Profile values during registration, depending on their policies and other client metadata. Authorization Servers MAY also choose to assign additional Entity Profiles that are not requested by the client based on their policies.

Below is an example of how this metadata parameter might be included in a Dynamic Client Registration request. Other standard parameters are omitted for brevity:

```
{
  "client_name": "Example Client",
  "redirect_uris": ["https://example.com/callback"],
  "client_profile": "service ai_agent"
}
```

7. Token Introspection Response Parameters

If an Authorization Server supports Token Introspection as defined in [RFC7662] and also implements the mechanisms defined in this specification, it SHOULD include the following parameters in its introspection responses:

`"sub_profile"`: OPTIONAL. The Entity Profile of the resource owner identified by the `sub` Claim in the access token.

`"client_profile"`: OPTIONAL. The Entity Profile of the client identified by the `client_id` Claim in the access token.

The following is a non-normative example of how these parameters might appear in a token introspection response. Other standard parameters are omitted for brevity:

```
{
  "active": true,
  "sub": "user123",
  "sub_profile": "user",
  "client_id": "client456",
  "client_profile": "native_app"
}
```

8. Client Behavior

8.1. Registration

Clients implementing this specification:

- * MAY declare `client_profile` at registration when using OAuth Dynamic Client Registration
- * MUST use registered or private Entity Profiles
- * MUST NOT attempt to register with Entity Profiles that do not match their actual nature

In environments where clients are manually registered or configured (e.g., enterprise deployments), `client_profile` provided MUST accurately represent the nature of the client. The value MUST be a registered Entity Profile or follow the naming conventions for private Entity Profiles.

Clients are incentivized to declare accurate `client_profile` values because authorization and access policies rely on them. Misrepresenting profiles may cause stricter enforcement, failed authorizations, or rejection due to incompatible policy constraints. Moreover, Authorization Servers and Resource Servers should monitor client behavior to ensure consistency with the declared profile, with deviations potentially resulting in penalties or revocation of access.

9. Authorization Server Behavior

9.1. Client Registration

During client registration (dynamically or otherwise), Authorization Servers implementing this specification:

- * MUST verify the `client_profile` value during client registration, if present in the registration request.

- * MUST ensure that the `client_profile` value conforms to the rules defined in Section 3.3.
- * MUST reject registration requests with invalid or unrecognized `client_profile` values.
- * SHOULD provide clear error messages when rejecting registration requests due to invalid or unrecognized `client_profile` values.
- * MAY log Client Profile information for auditing and monitoring purposes.
- * MAY restrict clients from updating their `client_profile` after registration or limit the frequency of such updates, depending on their policies.

9.2. Subject Profile Assignment

The Authorization Server MAY determine the `sub_profile` value based on:

- * Attributes of the subject (e.g., user, service account, AI agent) as provided during registration.
- * Subject attributes returned during authentication.
- * The grant type used (e.g., `client_credentials` implies service account).
- * Preconfigured mappings.

The client's declared profile or request parameters MUST NOT directly influence or determine the value of `sub_profile` to prevent manipulation or unauthorized elevation of subject classification.

9.3. JWT Token Issuance

Authorization Servers implementing this specification:

- * MAY include `sub_profile` and `client_profile` Claims in access tokens and ID tokens, according to their policies and client registration metadata. When included, these Claims MUST conform to the rules defined in Section 3.3.
- * MAY automatically include Entity Profile Claims for certain categories of clients, such as always providing `client_profile` for autonomous AI agent clients.

- * SHOULD include these Claims when explicitly requested by clients through supported mechanisms (e.g., OpenID Connect Claims parameter [OIDC], specific scope requests).
- * SHOULD populate these Claims consistently using verified Entity Profile information obtained during client registration or other trusted validation methods.
- * When issuing refreshed or exchanged tokens, Authorization Servers SHOULD re-evaluate the Entity Profiles and update them if context or identity has changed. Entity Profiles MUST NOT be assumed to persist across sessions without validation.

This specification does not prescribe how clients request Entity Profile Claims. In practice, Authorization Servers may enforce the inclusion of these Claims, especially for high-risk or privileged profiles such as AI agents, to ensure consistent and secure policy enforcement. For other entity profiles, to reduce token size and privacy exposure, Claims may be omitted by default and included only when explicitly requested through standard mechanisms like the OpenID Connect claims parameter [OIDC] or OAuth scopes. This approach balances the need for security and efficient token management, while preventing clients from arbitrarily adding or omitting Entity Profile information to manipulate access control decisions.

9.4. Token Introspection Responses

Authorization Servers implementing this specification:

- * SHOULD include `sub_profile` and `client_profile` parameters in token introspection responses. If included, Authorization Servers MUST ensure that the values of these parameters conform to the rules defined in Section 3.3.

9.5. Validation Requirements

Authorization Servers:

1. MUST verify that Entity Profile values are either registered in the OAuth Entity Profiles registry or follow proper namespaced private conventions per the rules defined in Section 3.3.
2. SHOULD ensure that Entity Profile assignments are trustworthy and not based solely on unverified self-assertion. The mechanisms for these verifications are out of scope for this specification, but it is recommended that Authorization Servers implement appropriate checks based on their security policies and operational context.

3. MUST enforce authentication assurance and policy requirements appropriate to the Entity Profile.

If validation fails during:

- * ***Client registration***: Authorization Servers SHOULD return `invalid_client_metadata` (as defined in [RFC7591]).
- * ***Token issuance***: Servers MAY refuse to issue tokens or omit the invalid profile Claims.
- * ***Token introspection***: Servers SHOULD ensure introspection results match stored profile metadata and MUST NOT fabricate or guess unknown profiles.

Clear, actionable error responses MUST be returned in accordance with OAuth and OpenID Connect error handling frameworks.

Authorization Servers MAY:

1. Log Entity Profile assignments to support auditing and forensic analysis.
2. Incorporate Entity Profiles into rate limiting and other risk-based controls.
3. Return clear, descriptive error messages if Entity Profile validation fails.

10. Resource Server Behavior

Resource Servers handling tokens with Entity Profile Claims:

- * SHOULD use `sub_profile` and `client_profile` Claims to inform access control decisions and apply appropriate policies.
- * SHOULD apply conservative or default-deny policies when processing unrecognized Entity Profile Claims, when Entity Profile values are missing, or when the semantics of the Entity Profile are not known.
- * MAY log Entity Profile information for auditing, monitoring, and anomaly detection purposes.
- * MUST NOT interpret or infer additional meaning beyond the profile's definition.

This specification does not prescribe specific behaviors or policies for Resource Servers based on Entity Profiles. However, it encourages Resource Servers to use these Claims to strengthen security, enforce fine-grained policies, and improve user experience.

When rejecting a request due to invalid, missing, or unsupported Entity Profile Claims, Resource Servers SHOULD provide informative error responses to assist with diagnostics and troubleshooting. These responses SHOULD use existing OAuth 2.0 and HTTP mechanisms, such as the WWW-Authenticate [RFC6750] header or the error_description field [RFC6749] in the response body, to convey additional context.

These messages are intended for human end-users or developers and are not standardized by this specification. Clients MUST NOT rely on specific error formats for automated decision-making.

10.1. Handling Multiple Entity Profiles in Authorization Policies

When multiple Entity Profiles are present, authorization policies associated with each should be evaluated in combination, with the most restrictive outcome prevailing. Implementers should avoid writing Entity Profile checks that override previous decisions imperatively, as this can lead to inconsistent behavior.

Here is a non-normative pseudo-code example of how these values can be used for enforcing authorization policies at a Resource Server:

```
client_profiles = set(token.get("client_profile", "").split(" "))
sub_profiles = set(token.get("sub_profile", "").split(" "))

# Define applicable policies
policies_to_apply = []

if "service" in client_profiles:
    # Add service-specific policies
    policies_to_apply.append(apply_service_policy)

if "ai_agent" in sub_profiles:
    # Add AI agent-specific policies
    policies_to_apply.append(apply_ai_agent_policy)

if "user" in sub_profiles:
    # Add user-specific policies
    policies_to_apply.append(apply_user_policy)

# Evaluate combined policies (e.g., intersection of permissions or most restrictive)
final_decision = evaluate_combined_policies(policies_to_apply, mode="most_restrictive")
)
```

11. Delegation Considerations

Entity Profile Claims do not define or imply delegation relationships. They are intended solely to classify entities participating in an OAuth flow and to provide additional information that can assist relying parties in enforcing policies or making authorization decisions.

Implementations MAY include Entity Profile Claims in delegated contexts to convey additional information about the delegator or delegate. For example, in a token exchange flow [RFC8693], the act Claim could include a `sub_profile` value to indicate the profile of the acting entity within the delegation chain.

12. Security Considerations

12.1. Trust and Verification

Entity Profile Claims should only be trusted when issued by verified and trusted authorities. Without proper validation, self-asserted or spoofed Claims may result in misclassification and undermine security policies. Authorization Servers should enforce strict validation during client registration and token issuance.

12.2. Entity Profile Spoofing

Malicious clients may attempt to register or use false Entity Profiles to bypass access controls. To mitigate this, Authorization Servers should require verification for all profiles, especially privileged types like `"ai_agent"`, and restrict dynamic registration of sensitive profiles unless additional validation is performed.

Implementations are encouraged to monitor client behavior for consistency with the declared profile, applying penalties or revoking access when discrepancies indicate potential abuse or security risks.

12.3. Consistency and Semantics

Authorization Servers, Resource Servers, and Clients must interpret Entity Profile Claims consistently. To avoid misclassification or policy errors they should follow the definitions and semantics outlined in the specification, and not infer additional meanings or relationships beyond the defined Entity Profiles. This consistency is crucial for interoperability and security.

12.4. Separation from Authentication and Assurance

Entity Profile Claims are intended solely for classification and do not convey authentication strength or assurance. It is advised not to use these Entity Profile Claims as replacements for Claims such as acr (Authentication Context Class Reference [OIDC]). Authorization decisions should incorporate authentication context Claims alongside Entity Profiles to ensure robust access control.

Also, different Entity Profiles often require different levels of authentication assurance. It is recommended that Authorization Servers define minimum assurance requirements per Entity Profile, express these through the acr Claim, and reject authorization requests that do not meet the required assurance level. This alignment helps prevent unauthorized access by weakly authenticated entities.

12.5. Layered Policy Enforcement

It is recommended that Entity Profile Claims be used as one element within a multi-layered authorization policy. Relying exclusively on Entity Profile Claims can create brittle or exploitable policies. Implementers should combine Entity Profile Claims with scopes, roles, authentication assurance, and other contextual information.

12.6. Audit and Monitoring

It is recommended that Entity Profile Claims be incorporated thoughtfully into logging, telemetry, and audit trails to improve visibility into system behavior and support forensic investigations. However, implementers should balance these benefits against potential privacy concerns, ensuring that sensitive classification information is not overexposed or retained longer than necessary. Careful access controls and data minimization practices are advised when handling logs containing Entity Profile details.

12.7. Default Handling

While it is recommended to apply conservative or restrictive policies when Entity Profile Claims are missing, Resource Servers should also consider backward compatibility. Many existing tokens and systems may not include these Claims, so resources might need to accept requests without them. In such cases, applying default or fallback policies that balance security and usability is advised. Monitoring and logging requests missing Entity Profile Claims can help identify when stronger enforcement or client registration updates are needed.

12.8. Misclassification Risks

Entity Profile Claims can be misclassified or misinterpreted, leading to incorrect access control decisions. It is recommended that implementers validate Entity Profile Claims against known registries or trusted sources to ensure accurate classification. Implementers should also consider the implications of misclassification and implement fallback mechanisms or default policies to mitigate risks.

12.9. Token Bloating

Entity Profile Claims can increase the size of JWT access tokens or ID tokens, especially when multiple profiles are included. This can lead to performance issues, particularly in environments with limited bandwidth or storage. Implementers should consider the impact of token size on their systems and apply data minimization principles, only including Entity Profile Claims when necessary.

13. Privacy Considerations

Entity Profile Claims can reveal sensitive information about clients, users, or system architecture. When exposed improperly, they may increase risks related to fingerprinting, profiling, or data leakage. Implementers should evaluate the privacy impact of using these Claims and apply protective measures accordingly.

13.1. Fingerprinting

Entity Profile Claims can potentially be used to fingerprint clients or subjects based on their operational context. It is recommended that implementers consider the privacy implications of exposing Entity Profile information in tokens, especially when these Claims are accessible to untrusted parties. It is recommended to only include Entity Profile Claims when necessary, and assess whether their inclusion could enable cross-context tracking or re-identification.

13.2. Profiling and Behavioral Inference

Entity Profile Claims may unintentionally support profiling of users or clients, enabling assumptions about behavior, preferences, or roles. It is recommended that implementers carefully consider the potential for behavioral inference and ensure that Entity Profile Claims are not used to make unwarranted assumptions about user behavior or preferences. Implementers should also be cautious about exposing sensitive Entity Profile information in tokens that could be accessible to untrusted parties.

13.3. Minimization

Implementers should apply data minimization principles when including Entity Profile Claims in tokens. This means only including Entity Profile information that is necessary for the intended purpose and avoiding overexposure of sensitive classification details.

Implementers should also consider whether Entity Profile Claims are needed in all contexts or if they can be omitted in certain scenarios to reduce privacy risks.

13.4. Data Exposure

Entity Profile Claims may reveal sensitive information about system architecture or user categories. It is advised to carefully consider the exposure of these Claims in tokens accessible to untrusted parties. Limiting or anonymizing Entity Profile information can reduce the risk of unintended data disclosure.

14. IANA Considerations

14.1. OAuth Entity Profiles Registry

This specification requests the creation of a new IANA registry titled "OAuth Entity Profiles Registry" within the "OAuth Parameters" group. This registry will contain the string identifiers used to represent the Entity Profiles described in this specification.

NOTE: Private or vendor-specific Entity Profiles may use namespaced values and do not require IANA registration.

14.1.1. Registration Template

Each registry entry MUST include:

- * Entity Profile Name: A case-insensitive ASCII string representing the entity type (e.g., "user").
- * Entity Profile Description: Brief human-readable description of the entity type.
- * Usage location: The location(s) where the Entity Profile can be used. The possible locations are "Subject Profile" and "Client Profile".
- * Change Controller: The party responsible for the definition (e.g., IESG).

- * Specification Document: A stable URL or RFC that defines the semantics and use of the value.

14.1.2. Initial Registry Contents

14.1.2.1. user

- * Entity Profile Name: "user"
- * Entity Profile Description: Human resource owner interacting through a client
- * Usage Location: "Subject Profile"
- * Change Controller: IESG
- * Specification Document: Section 3.1 of this document.

14.1.2.2. device

- * Entity Profile Name: "device"
- * Entity Profile Description: Hardware device or IoT endpoint
- * Usage Location: "Client Profile", "Subject Profile"
- * Change Controller: IESG
- * Specification Document: Section 3.1 of this document.

14.1.2.3. native_app

- * Entity Profile Name: "native_app"
- * Entity Profile Description: Native application running on a device (e.g., mobile app, desktop app)
- * Usage Location: "Client Profile"
- * Change Controller: IESG
- * Specification Document: Section 3.1 of this document.

14.1.2.4. web_app

- * Entity Profile Name: "web_app"

- * Entity Profile Description: Web application as defined in [RFC6749].
- * Usage Location: "Client Profile"
- * Change Controller: IESG
- * Specification Document: Section 3.1 of this document.

14.1.2.5. browser_app

- * Entity Profile Name: "browser_app"
- * Entity Profile Description: User-agent-based application as defined in [RFC6749].
- * Usage Location: "Client Profile"
- * Change Controller: IESG
- * Specification Document: Section 3.1 of this document.

14.1.2.6. service

- * Entity Profile Name: "service"
- * Entity Profile Description: Non-human backend service or microservice.
- * Usage Location: "Client Profile", "Subject Profile"
- * Change Controller: IESG
- * Specification Document: Section 3.1 of this document.

14.1.2.7. ai_agent

- * Entity Profile Name: "ai_agent"
- * Entity Profile Description: Autonomous or semi-autonomous AI-based entity.
- * Usage Location: "Client Profile", "Subject Profile"
- * Change Controller: IESG
- * Specification Document: Section 3.1 of this document.

14.2. JWT Claims Registration

This document requests registration of the following Claims in the "JSON Web Token Claims" registry:

14.2.1. client_profile Claim

- * Claim Name: "client_profile"
- * Claim Description: Client Entity Profile information
- * Change Controller: IESG
- * Specification Document: Section 4 of this document.

14.2.2. sub_profile Claim

- * Claim Name: "sub_profile"
- * Claim Description: Subject or resource owner Entity Profile information
- * Change Controller: IESG
- * Specification Document: Section 4 of this document.

14.3. Authorization Server Metadata Registration

IANA is requested to register the following fields in the "OAuth Authorization Server Metadata" [RFC8414] registry:

14.3.1. entity_profiles_supported

- * Metadata Name: entity_profiles_supported
- * Metadata Description: JSON object containing two JSON arrays: client and subject, each listing the Entity Profiles supported.
- * Change Controller: IESG
- * Specification Document: Section 5 of this document.

14.4. Dynamic Client Registration Metadata Registration

IANA is requested to register the following field in the "OAuth Dynamic Client Registration Metadata" registry:

14.4.1. client_profile

- * Client Metadata Name: "client_profile"
- * Client Metadata Description: Entity Profile information of the registering client
- * Change Controller: IESG
- * Specification Document: Section 6 of this document.

14.5. Token Introspection Response Registration

IANA is requested to register the following fields in the "OAuth Token Introspection Response" [RFC7662] registry:

14.5.1. client_profile

- * Name: "client_profile"
- * Description: Entity Profile information of the client
- * Change Controller: IESG
- * Specification Document: Section 7 of this document.

14.5.2. sub_profile

- * Name: "sub_profile"
- * Description: Entity Profile information of the subject or resource owner
- * Change Controller: IESG
- * Specification Document: Section 7 of this document.

15. References

15.1. Normative References

- [OIDC] "OpenID Connect Core 1.0", 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>>.

- [RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <<https://www.rfc-editor.org/rfc/rfc5234>>.
- [RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <<https://www.rfc-editor.org/rfc/rfc6749>>.
- [RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, DOI 10.17487/RFC6750, October 2012, <<https://www.rfc-editor.org/rfc/rfc6750>>.
- [RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <<https://www.rfc-editor.org/rfc/rfc7519>>.
- [RFC7591] Richer, J., Ed., Jones, M., Bradley, J., Machulak, M., and P. Hunt, "OAuth 2.0 Dynamic Client Registration Protocol", RFC 7591, DOI 10.17487/RFC7591, July 2015, <<https://www.rfc-editor.org/rfc/rfc7591>>.
- [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>>.
- [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>>.
- [RFC8628] Denniss, W., Bradley, J., Jones, M., and H. Tschofenig, "OAuth 2.0 Device Authorization Grant", RFC 8628, DOI 10.17487/RFC8628, August 2019, <<https://www.rfc-editor.org/rfc/rfc8628>>.

15.2. Informative References

- [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>>.

Appendix A. Example Usage in Various Flows and Use Cases

The following non-normative examples illustrate how Entity Profiles might appear in various OAuth flows.

Flow / Use Case	Client Profile	Subject Profile
Authorization Code Flow (Web App)	web_app	user
Authorization Code Flow (SPA)	browser_app	user
Authorization Code Flow (Mobile App)	native_app	user
Client Credentials Flow (Backend Service)	service	service
Device Authorization Flow (Smart TV app)	native_app	user
IoT sensors reporting telemetry	device	device
A web app talking to a downstream API on behalf-of a user	service web_app	user
Resource Owner Password Credentials Flow (legacy)	web_app	user
S2S OBO Flows (e.g., Service Mesh)	service	service
An AI acting as itself (e.g., Workspace bots)	service ai_agent	ai_agent service
AI agents acting on behalf of a user (e.g., personal assistant)	ai_agent	user
An AI agent running in a desktop app	native_app ai_agent	user

Table 1

Appendix B. Document History

B.1. draft-00

- * Initial draft

Acknowledgments

The authors would like to thank the following people for their contributions and reviews of this specification: Adrian Frei, Anna Barhudarian, Diana Smetters, Emily Lauber.

Authors' Addresses

Sreyantha Chary Mora
Microsoft Corporation
Email: sreyanthmora@microsoft.com

Pamela Dingle
Microsoft Corporation
Email: pamela.dingle@microsoft.com