Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: September 27, 2026

R. Sharif
CyberSecAI Ltd

March 26, 2026

OpenID Connect Agent Identity Claims for Autonomous AI Agents

draft-sharif-openid-agent-identity-00

Abstract

This specification defines a profile of OpenID Connect Core 1.0 that enables Identity Providers (IdPs) to issue identity tokens for autonomous software agents. It introduces a set of standard claims for representing agent identity, ownership, trust posture, authorised capabilities, and compliance screening status within OpenID Connect ID Tokens.

The profile is designed to operate within existing OpenID Connect infrastructure without requiring modifications to the core protocol. It defines how Relying Parties (RPs) validate agent tokens and enforce graduated access controls based on agent trust levels and sanctions screening results.

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 September 27, 2026.

Copyright Notice

Table of Contents

1. Introduction

OpenID Connect Core 1.0 [OpenID.Core] defines an identity layer on top of OAuth 2.0 [RFC6749] that enables Clients to verify the identity of End-Users. The core specification models identity around human subjects: natural persons who authenticate via user agents (typically web browsers) and consent to the release of identity claims to Relying Parties.

The emergence of autonomous software agents -- programs that act on behalf of users or organisations to perform tasks, execute transactions, and interact with services without continuous human supervision -- creates a requirement for machine-readable identity that existing OpenID Connect claims do not address.

Specifically, current limitations include:

- The "sub" claim identifies an End-User, not the autonomous agent acting on their behalf. When multiple agents operate under the same user account, there is no standard mechanism to distinguish between them.

- There is no standard claim to express the trust posture of an agent. Relying Parties cannot make graduated access control decisions based on an agent’s behavioural history

or verification level.

- There is no standard claim to express what actions an agent is authorised to perform, distinct from OAuth scopes which express what resources the client application may access.

- There is no mechanism for Relying Parties to determine whether an agent or its controlling entity has been screened against sanctions lists, a requirement in regulated financial services.

This specification addresses these gaps by defining a profile of OpenID Connect that introduces agent-specific claims carried within standard ID Tokens. The profile is designed with the following principles:

1. Backward compatibility: All defined claims are additional. Existing OpenID Connect deployments that do not recognise agent claims will ignore them per Section 5.1 of [OpenID.Core], which states that Claims not understood MUST be ignored.

2. Separation of concerns: The "sub" claim continues to identify the End-User or organisational entity. The "agent_id" claim identifies the specific agent instance. The "agent_owner" claim links the agent to its controlling subject.

3. Graduated trust: The trust model supports five levels (L0 through L4) that map to progressively wider sets of permitted actions, enabling Relying Parties to implement proportionate access controls.

4. Regulatory alignment: Claims for sanctions screening status and spend limits enable compliance with Anti-Money Laundering (AML) regulations in jurisdictions that require entity screening before financial transactions.

This specification is informed by work on agent payment trust frameworks [I-D.sharif-agent-payment-trust], security guidance for Model Context Protocol deployments [I-D.sharif-mcps-secure-mcp], and emerging requirements for autonomous agent identity in financial services.

2. Requirements Notation and 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.

In the following text, all references to "Section" without a document identifier refer to sections within this specification.

3. Terminology

This specification uses the terms "Access Token", "Authorization Code", "Authorization Endpoint", "Authorization Server", "Client", "Client Authentication", "Client Identifier", "Grant Type", "ID Token", "Refresh Token", "Resource Server", "Response Type", and "Token Endpoint" defined by OAuth 2.0 [RFC6749], the terms "Claim", "Claim Type", "Claims Provider", "ID Token", "Issuer", and "Relying Party (RP)" defined by OpenID Connect Core 1.0 [OpenID.Core], and the terms "JSON Web Token (JWT)", "Claim Name", "Claim Value", "Header", "Payload", and

"Signature" defined by JSON Web Token (JWT) [RFC7519].

This specification also defines the following terms:

Agent
: An autonomous software program that acts on behalf of a user or organisation to perform tasks, execute transactions, or interact with services. An Agent operates with a degree of autonomy and MAY make decisions within the boundaries of its authorised capabilities.

Agent Identity Provider (Agent IdP)
: An OpenID Provider (OP) that issues ID Tokens containing the agent claims defined in this specification. An Agent IdP MAY be the same entity as the user's OpenID Provider or MAY be a separate service specialising in agent identity.

Agent Owner
: The End-User or organisation that deployed and controls the Agent. The Agent Owner is identified by the "agent_owner" claim and is responsible for the Agent's actions.

Trust Score
: A numerical representation of an Agent's behavioural trustworthiness, expressed as an integer in the range [0, 100]. Trust Scores are computed by the Agent IdP based on criteria including transaction history, attestation strength, and compliance screening results.

Trust Level
: A graduated classification of an Agent's trust posture, expressed as one of five levels (L0 through L4). Trust Levels determine the maximum scope of actions an Agent is permitted to perform at a given Relying Party.

4. Agent Identity Claims

This section defines the claims that represent agent identity within an OpenID Connect ID Token. These claims are carried in the ID Token payload alongside standard OpenID Connect claims.

All claims defined in this section use the "agent_" prefix to avoid collision with existing OpenID Connect claims and to enable Relying Parties to identify agent-specific assertions unambiguously.

The following table summarises the agent claims:

Claim	Type	Required
agent_id	string	REQUIRED
agent_name	string	OPTIONAL
agent_owner	string	REQUIRED
agent_trust_score	integer	OPTIONAL
agent_trust_level	string	OPTIONAL
agent_capabilities	array	OPTIONAL
agent_sanctions_status	string	OPTIONAL
agent_spend_limit	integer	OPTIONAL
agent_attestation_method	string	OPTIONAL
agent_created_at	integer	OPTIONAL

Table 1: Agent Identity Claims

4.1. agent_id

Claim Name: agent_id

Claim Description: A globally unique identifier for the Agent.

Claim Type: String

Requirement Level: REQUIRED

The value MUST be a non-empty string of no more than 255 characters. The value MUST be unique within the namespace of the issuing Agent IdP. To achieve global uniqueness across IdPs, implementations SHOULD use one of the following formats:

- A UUID v4 as defined in [RFC4122], e.g., "550e8400-e29b-41d4-a716-446655440000"
- A URN using a domain-scoped identifier, e.g., "urn:example.com:agent:payment-bot-001"
- A domain-qualified identifier using the dot-separated convention, e.g., "payment-bot.example.com"

The Agent IdP MUST NOT reuse an "agent_id" value for a different agent instance. Once assigned, an "agent_id" MUST remain stable for the lifetime of the agent.

Example:

```
"agent_id": "payment-bot.cybersecai.co.uk"
```

4.2. agent_name

Claim Name: agent_name

Claim Description: A human-readable display name for the Agent.

Claim Type: String

Requirement Level: OPTIONAL

The value MUST be a non-empty string of no more than 128 characters when present. The value is intended for display purposes in administrative interfaces and audit logs. Relying Parties MUST NOT use the "agent_name" for identification or access control decisions; the "agent_id" claim MUST be used for that purpose.

Example:

```
"agent_name": "Payment Processing Agent"
```

4.3. agent_owner

Claim Name: agent_owner

Claim Description: The subject identifier of the entity that deployed and controls the Agent.

Claim Type: String

Requirement Level: REQUIRED

The value MUST be the "sub" claim value of the Agent Owner as issued by the same Agent IdP, or a URI that resolves to the owner’s identity record at the Agent IdP. This establishes a

verifiable chain of accountability from the Agent to its controlling entity.

When the Agent IdP is the same OpenID Provider that issued the owner's identity, the value MUST be identical to the "sub" claim in the owner's own ID Token.

When the Agent IdP is a separate service, the value SHOULD be a URI of the form:

```
https://{owner-idp}/.well-known/openid-configuration
  #sub={value}
```

Relying Parties MUST verify that the "agent_owner" value corresponds to a valid, active subject at the referenced identity provider before granting access to protected resources.

Example:

```
"agent_owner": "user_2fK9xLm3nP7qR8s"
```

4.4. agent_trust_score

Claim Name: agent_trust_score

Claim Description: A numerical trust score representing the Agent's behavioural trustworthiness.

Claim Type: Integer

Requirement Level: OPTIONAL

The value MUST be an integer in the range [0, 100] inclusive. A value of 0 indicates no established trust. A value of 100 indicates the highest level of trust.

The methodology for computing trust scores is not defined by this specification and is at the discretion of the Agent IdP. However, the Agent IdP SHOULD document its trust score computation methodology in its OpenID Provider metadata or in a referenced policy document.

Factors that MAY influence the trust score include:

- Duration and consistency of agent operation
- Transaction completion rate and dispute history
- Strength of the attestation method used
- Sanctions screening result
- Compliance with rate limits and usage policies

Example:

```
"agent_trust_score": 72
```

4.5. agent_trust_level

Claim Name: agent_trust_level

Claim Description: A graduated trust classification for the Agent.

Claim Type: String

Requirement Level: OPTIONAL

The value MUST be one of the following enumerated strings:

Level	Label	Description
"L0"	Unverified	No identity verification performed. Agent MAY only access public resources.
"L1"	Basic	API key or shared secret attestation. Agent MAY perform read operations.
"L2"	Verified	Certificate or JWT-based attestation. Agent MAY perform write operations within defined limits.
"L3"	Trusted	Challenge-response attestation with cryptographic proof. Agent MAY perform financial transactions within spend limits.
"L4"	High Assurance	Multi-factor attestation including hardware-bound keys. Agent MAY perform high-value transactions.

Table 2: Agent Trust Levels

When both "agent_trust_score" and "agent_trust_level" are present, the "agent_trust_level" MUST be consistent with the "agent_trust_score". The following mapping defines the minimum trust score for each level:

```
L0: score < 20
L1: 20 <= score < 40
L2: 40 <= score < 60
L3: 60 <= score < 80
L4: score >= 80
```

If the values are inconsistent, the Relying Party MUST reject the token.

Example:

```
"agent_trust_level": "L3"
```

4.6. agent_capabilities

Claim Name: agent_capabilities

Claim Description: The set of actions the Agent is authorised to perform.

Claim Type: JSON array of strings

Requirement Level: OPTIONAL

Each element of the array MUST be a non-empty string that identifies a specific capability. Capability identifiers SHOULD use the dot-separated hierarchical naming convention:

```
{domain}.{resource}.{action}
```

Relying Parties MUST treat the "agent_capabilities" claim as the maximum set of actions the Agent is authorised to perform.

A Relying Party MAY further restrict the effective capabilities based on its own policies, the Agent's trust level, or other contextual factors.

The "agent_capabilities" claim is distinct from OAuth 2.0 scopes. Scopes define what resources the client application may access at the authorisation server level. Agent capabilities define what actions the specific Agent instance is permitted to perform at the application level.

Example:

```
  "agent_capabilities": [
    "payments.transfer.initiate",
    "payments.balance.read",
    "reporting.transactions.export"
  ]
```

4.7. agent_sanctions_status

Claim Name: agent_sanctions_status

Claim Description: The result of sanctions screening performed on the Agent and/or its owner.

Claim Type: String

Requirement Level: OPTIONAL

The value MUST be one of the following enumerated strings:

Value	Description
"CLEAR"	The Agent and its owner have been screened against applicable sanctions lists and no matches were found.
"HIT"	The screening process identified a potential match against one or more sanctions lists. The Agent MUST NOT be permitted to perform financial transactions until the match is resolved.
"NOT_SCREENED"	No sanctions screening has been performed.

Table 3: Sanctions Status Values

When the value is "HIT", the Agent IdP SHOULD include additional context in a separate claims source or via a UserInfo endpoint response. This specification does not define the format of additional sanctions screening details.

The Agent IdP MUST record the date and time of the most recent screening. Relying Parties operating in regulated environments SHOULD verify that the screening was performed within a timeframe acceptable to their compliance policies (see Section 8.4).

Example:

```
  "agent_sanctions_status": "CLEAR"
```

4.8. agent_spend_limit

Claim Name: agent_spend_limit

Claim Description: The maximum transaction amount the Agent is authorised to initiate in a single transaction.

Claim Type: Integer

Requirement Level: OPTIONAL

The value MUST be a non-negative integer representing the amount in minor currency units (e.g., pence for GBP, cents for USD). A value of 0 indicates that the Agent is not authorised to initiate financial transactions.

The currency is not encoded in this claim. The applicable currency MUST be determined from context, such as the Relying Party's configuration or an associated claim in the ID Token. If the currency is ambiguous, the Relying Party MUST reject financial transaction requests.

Example (representing GBP 250.00):

```
"agent_spend_limit": 25000
```

4.9. agent_attestation_method

Claim Name: agent_attestation_method

Claim Description: The method by which the Agent's identity was verified by the Agent IdP.

Claim Type: String

Requirement Level: OPTIONAL

The value MUST be one of the following enumerated strings:

Value	Description
"challenge_response"	Identity verified via a cryptographic challenge-response protocol (see Section 6.2).
"certificate"	Identity verified via a valid X.509 certificate presented during TLS client authentication or at the application layer.
"jwt"	Identity verified via a signed JWT presented as a client assertion per [RFC7523].
"api_key"	Identity verified via a pre-shared API key.

Table 4: Attestation Method Values

The attestation method informs Relying Parties of the strength of the identity verification. Relying Parties SHOULD require "challenge_response" or "certificate" attestation for agents performing high-value or sensitive operations.

Example:

```
  "agent_attestation_method": "challenge_response"
```

4.10. agent_created_at

Claim Name: agent_created_at

Claim Description: The time at which the Agent was created or registered with the Agent IdP.

Claim Type: Integer

Requirement Level: OPTIONAL

The value MUST be a NumericDate as defined in Section 2 of [RFC7519], representing the number of seconds from 1970-01-01T00:00:00Z UTC to the agent creation time.

Relying Parties MAY use this claim in conjunction with the trust score to assess whether a newly created agent with an unexpectedly high trust score warrants additional scrutiny.

Example (representing 2026-01-15T09:30:00Z):

```
  "agent_created_at": 1768561800
```

5. Agent ID Token

The Agent ID Token is a standard OpenID Connect ID Token [OpenID.Core] Section 2 that includes the agent claims defined in Section 4 of this specification.

The Agent ID Token MUST conform to all requirements of an ID Token as defined in Section 2 of [OpenID.Core]. In particular:

- The token MUST be a JWT [RFC7519] signed using JWS [RFC7515].
- The "iss" claim MUST contain the Issuer Identifier of the Agent IdP.
- The "sub" claim MUST contain a subject identifier. When the Agent acts on behalf of a user, the "sub" claim SHOULD contain the user’s subject identifier. When the Agent acts on behalf of an organisation without a specific user context, the "sub" claim SHOULD contain an organisational identifier.
- The "aud" claim MUST contain the client_id of the Relying Party.
- The "iat" and "exp" claims MUST be present.
- The token MUST contain at minimum the "agent_id" and "agent_owner" claims defined in this specification.

The following is a non-normative example of an Agent ID Token payload:

```
  {
    "iss": "https://idp.example.com",
    "sub": "org_8kP2mN5xQ9",
    "aud": "client_rp_payments_001",
    "exp": 1768565400,
    "iat": 1768561800,
    "nonce": "n-0S6_WzA2Mj",
    "agent_id": "payment-bot.example.com",
    "agent_name": "Payment Processing Agent",
    "agent_owner": "org_8kP2mN5xQ9",
```

```
    "agent_trust_score": 72,
    "agent_trust_level": "L3",
    "agent_capabilities": [
      "payments.transfer.initiate",
      "payments.balance.read",
      "reporting.transactions.export"
    ],
    "agent_sanctions_status": "CLEAR",
    "agent_spend_limit": 25000,
    "agent_attestation_method": "challenge_response",
    "agent_created_at": 1768561800
  }
```

The following is the corresponding JOSE Header:

```
  {
    "alg": "RS256",
    "typ": "JWT",
    "kid": "2026-03-key-01"
  }
```

Agent claims MAY also be returned from the UserInfo Endpoint as defined in Section 5.3 of [OpenID.Core]. When agent claims are requested via the UserInfo Endpoint, the same claim names and semantics defined in Section 4 apply.

When an Agent IdP supports agent claims, it SHOULD advertise this capability in its OpenID Provider Metadata [OpenID.Discovery] by including the following metadata parameter:

```
  "agent_claims_supported": true
```

The Agent IdP SHOULD also list the supported agent claims in the "claims_supported" metadata parameter:

```
  "claims_supported": [
    "sub",
    "name",
    "email",
    "agent_id",
    "agent_name",
    "agent_owner",
    "agent_trust_score",
    "agent_trust_level",
    "agent_capabilities",
    "agent_sanctions_status",
    "agent_spend_limit",
    "agent_attestation_method",
    "agent_created_at"
  ]
```

6. Agent Authentication Flow

This section defines how an Agent IdP issues tokens for Agents and how Agents authenticate to Relying Parties.

6.1. Token Issuance via Client Credentials

Agent authentication uses the OAuth 2.0 Client Credentials grant type [RFC6749] Section 4.4, extended with agent claim parameters.

The Agent or its controlling software sends a token request to the Agent IdP’s Token Endpoint. The request MUST include the following parameters:

grant_type
   REQUIRED. Value MUST be "client_credentials".

client_id
   REQUIRED. The client identifier issued to the Agent or its
   controlling application by the Agent IdP.

client_secret or client_assertion
   REQUIRED. Client authentication as defined in Section 2.3
   of [RFC6749] or Section 2.2 of [RFC7523].

scope
   OPTIONAL. The requested scope. To request agent claims in
   the resulting token, the scope MUST include "openid" and
   SHOULD include "agent_identity".

agent_id
   REQUIRED. The identifier of the Agent for which the token
   is being requested.

The following is a non-normative example of a token request:

```
POST /token HTTP/1.1
Host: idp.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials
&client_id=agent_controller_001
&client_secret=s3cr3t
&scope=openid%20agent_identity%20payments.transfer
&agent_id=payment-bot.example.com
```

If the request is valid and the client is authorised to request tokens for the specified agent, the Agent IdP responds with a token response as defined in Section 4.4.3 of [RFC6749], with the addition of an "id_token" parameter containing the Agent ID Token:

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store

{
  "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6ImF0K...",
  "token_type": "Bearer",
  "expires_in": 3600,
  "id_token": "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIs..."
}
```

The Agent IdP MUST verify the following before issuing the token:

1. The client is authenticated and authorised to request tokens for the specified "agent_id".

2. The specified Agent exists and is in an active state.

3. The Agent’s trust score, trust level, and sanctions status are current.

4. The requested scopes are within the Agent’s authorised capabilities.

6.2. Challenge-Response Extension

For high-assurance agent verification (trust level L3 and above), this specification defines a challenge-response extension that provides cryptographic proof of agent identity.

The flow consists of two exchanges:

Step 1: Challenge Request

The Agent requests a challenge from the Agent IdP:

```
POST /agent/challenge HTTP/1.1
Host: idp.example.com
Content-Type: application/json

{
  "agent_id": "payment-bot.example.com",
  "client_id": "agent_controller_001"
}
```

The Agent IdP responds with a challenge:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "challenge": "dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk",
  "challenge_id": "ch_9xK2mN5pQ7rS",
  "expires_in": 300
}
```

The "challenge" value MUST be a cryptographically random string of at least 32 bytes, Base64url-encoded without padding. The Agent IdP MUST store the challenge and associate it with the specified "agent_id" and "client_id". The challenge MUST expire after the number of seconds indicated by "expires_in", which MUST NOT exceed 600 seconds.

Step 2: Challenge Response

The Agent signs the challenge using its private key and submits the response as part of the token request:

```
POST /token HTTP/1.1
Host: idp.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials
&client_id=agent_controller_001
&client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3A
  client-assertion-type%3Ajwt-bearer
&client_assertion=eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCJ9...
&scope=openid%20agent_identity
&agent_id=payment-bot.example.com
&challenge_id=ch_9xK2mN5pQ7rS
&challenge_response=MEUCIQDf1...
```

The "challenge_response" value MUST be the digital signature of the "challenge" value, computed using the Agent’s private key and encoded as Base64url without padding.

The Agent IdP MUST verify:

1. The "challenge_id" corresponds to a valid, unexpired challenge issued to the specified "agent_id" and "client_id".

2. The "challenge_response" is a valid signature of the original challenge value, verified using the Agent's registered public key.

3. The challenge has not been previously used (one-time use).

Upon successful verification, the Agent IdP issues a token with "agent_attestation_method" set to "challenge_response" and "agent_trust_level" of at least "L3".

The following is a non-normative example of the signed challenge JWT payload (used in client_assertion):

```
{
  "iss": "agent_controller_001",
  "sub": "payment-bot.example.com",
  "aud": "https://idp.example.com/token",
  "jti": "ch_9xK2mN5pQ7rS",
  "challenge":
    "dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk",
  "iat": 1768561800,
  "exp": 1768562100
}
```

6.3. Trust Level Requirements per Scope

Relying Parties SHOULD define minimum trust level requirements for each action scope they support. The following table provides a non-normative example of scope-to-trust-level mappings:

Scope	Min Trust Level	Min Attestation Method
data.public.read	L0	any
data.private.read	L1	api_key
data.private.write	L2	jwt
payments.transfer.initiate	L3	challenge_response
payments.high_value.initiate	L4	certificate

Table 5: Scope to Trust Level Mapping

When an Agent presents a token with a trust level below the minimum required for the requested action, the Relying Party MUST reject the request with an HTTP 403 response and SHOULD include an error response body indicating the required trust level:

```
HTTP/1.1 403 Forbidden
Content-Type: application/json

{
  "error": "insufficient_trust_level",
  "error_description": "This action requires
    agent_trust_level L3 or above. The presented
    token contains trust level L1.",
  "required_trust_level": "L3",
  "current_trust_level": "L1"
```

```
  }
```

7. Agent Token Validation

This section defines how Relying Parties validate Agent ID Tokens.

7.1. Standard Validation

Relying Parties MUST perform all validation steps defined in Section 3.1.3.7 of [OpenID.Core] for ID Token validation. In addition, the following agent-specific validation steps MUST be performed:

1. Verify that the "agent_id" claim is present and is a non-empty string of no more than 255 characters.

2. Verify that the "agent_owner" claim is present and is a non-empty string.

3. If the "agent_trust_score" claim is present, verify that it is an integer in the range [0, 100].

4. If the "agent_trust_level" claim is present, verify that it is one of the values defined in Table 2.

5. If both "agent_trust_score" and "agent_trust_level" are present, verify that they are consistent per the mapping defined in Section 4.5. If they are inconsistent, the token MUST be rejected.

6. If the "agent_capabilities" claim is present, verify that it is a JSON array of non-empty strings.

7. If the "agent_sanctions_status" claim is present, verify that it is one of the values defined in Table 3.

8. If the "agent_spend_limit" claim is present, verify that it is a non-negative integer.

9. If the "agent_attestation_method" claim is present, verify that it is one of the values defined in Table 4.

10. If the "agent_created_at" claim is present, verify that it is a valid NumericDate and that it is not in the future.

If any of the above validation steps fail, the Relying Party MUST reject the token and MUST NOT grant access to protected resources.

7.2. Trust Score Thresholds

Relying Parties that implement graduated access controls based on trust scores SHOULD define threshold values for each category of protected resource. The following is a non-normative example of threshold-based access control logic:

```
  function evaluateAccess(token, requestedAction) {
    // Reject if sanctions status is HIT
    if (token.agent_sanctions_status === "HIT") {
      return { allowed: false, reason: "sanctions_hit" };
    }

    // Determine required trust score for the action
    const thresholds = {
```

```
        "read":      20,
        "write":     40,
        "transfer":  60,
        "admin":     80
      };

      const requiredScore =
        thresholds[requestedAction.category];
      if (requiredScore === undefined) {
        return {
          allowed: false,
          reason: "unknown_action"
        };
      }

      if ((token.agent_trust_score || 0) < requiredScore) {
        return {
          allowed: false,
          reason: "insufficient_trust_score",
          required: requiredScore,
          actual: token.agent_trust_score || 0
        };
      }

      return { allowed: true };
    }
```

Relying Parties SHOULD log all access decisions, including the agent_id, requested action, trust score, and decision outcome, for audit purposes.

7.3. Sanctions Status Enforcement

Relying Parties operating in regulated financial services jurisdictions MUST enforce sanctions status as follows:

1. If the "agent_sanctions_status" claim is "HIT", the Relying Party MUST deny all financial transaction requests from the Agent. The Relying Party MAY permit non-financial read-only access at its discretion.

2. If the "agent_sanctions_status" claim is "NOT_SCREENED" and the Relying Party requires sanctions screening, the Relying Party MUST deny financial transaction requests and SHOULD return an error indicating that screening is required:

```
        HTTP/1.1 403 Forbidden
        Content-Type: application/json

        {
          "error": "sanctions_screening_required",
          "error_description": "Agent sanctions screening
            has not been performed.  Financial transactions
            require a CLEAR sanctions status."
        }
```

3. If the "agent_sanctions_status" claim is absent, the Relying Party MUST treat it as equivalent to "NOT_SCREENED" when sanctions screening is required by regulation or policy.

8. Security Considerations

8.1. Agent Impersonation

An attacker who obtains an Agent's client credentials can request tokens on behalf of the Agent. To mitigate this risk:

- Agent IdPs SHOULD require challenge-response attestation (Section 6.2) for Agents operating at trust level L3 or above.
- Agent private keys used for challenge-response SHOULD be stored in hardware security modules (HSMs) or trusted execution environments (TEEs).
- Agent IdPs SHOULD implement anomaly detection to identify token requests from unexpected network locations or at unusual times.
- Relying Parties SHOULD implement request-level anomaly detection in addition to token validation.

8.2. Trust Score Manipulation

Trust scores are asserted by the Agent IdP and are only as reliable as the IdP's computation methodology and integrity.

- Relying Parties MUST NOT accept trust scores from Agent IdPs that they have not explicitly trusted.
- Agent IdPs MUST protect trust score computation systems from tampering. Trust score updates SHOULD require multi-party authorisation for changes exceeding a defined threshold (e.g., an increase of more than 20 points in a single update).
- Agent IdPs SHOULD implement rate limiting on trust score increases to prevent rapid escalation attacks.
- Relying Parties MAY implement their own supplementary trust scoring that combines the Agent IdP's score with locally observed agent behaviour.

8.3. Token Replay

Agent ID Tokens are bearer tokens and are susceptible to replay attacks if intercepted.

- Agent ID Tokens SHOULD have short expiration times. An expiration time ("exp") of no more than 3600 seconds after issuance ("iat") is RECOMMENDED for general use. For financial transactions, an expiration time of no more than 300 seconds is RECOMMENDED.
- Relying Parties SHOULD implement token binding or sender-constrained tokens as defined in [RFC9449] (DPoP) to prevent token replay by parties other than the intended agent.
- When the nonce claim is present, Relying Parties MUST verify that the nonce has not been previously used within the token's validity period.

8.4. Sanctions Screening Freshness

Sanctions lists are updated periodically. A sanctions screening result that was valid at the time of screening may become stale.

- Agent IdPs SHOULD re-screen Agents and their owners at

regular intervals. A re-screening interval of no more than 24 hours is RECOMMENDED for Agents performing financial transactions.

- Agent IdPs SHOULD include a "screened_at" claim (NumericDate) in the ID Token or make it available via the UserInfo Endpoint, indicating when the most recent screening was performed.

- Relying Parties in regulated environments SHOULD reject tokens where the sanctions screening is older than their compliance policy permits. A maximum staleness of 24 hours is RECOMMENDED for PSD2-regulated services.

9. Privacy Considerations

9.1. Agent Identity as PII

When an Agent acts on behalf of a specific End-User, the combination of "agent_id" and "agent_owner" can constitute Personally Identifiable Information (PII) as defined in applicable data protection regulations, including the UK GDPR and EU GDPR.

- Agent IdPs MUST process agent identity data in accordance with applicable data protection regulations.

- When an Agent acts on behalf of a natural person, the Agent Owner MUST have provided informed consent for the Agent IdP to disclose their subject identifier in the "agent_owner" claim.

- Relying Parties MUST NOT retain agent identity claims beyond the period necessary for the purpose for which they were collected, subject to regulatory retention requirements.

- Agent IdPs SHOULD support pairwise subject identifiers as defined in Section 8 of [OpenID.Core] for the "agent_owner" claim to limit cross-Relying Party correlation of Agent Owners.

9.2. Minimal Disclosure

Agent IdPs and Relying Parties SHOULD implement the principle of minimal disclosure:

- Agent IdPs SHOULD issue tokens containing only the agent claims requested by the Relying Party and necessary for the intended transaction.

- Relying Parties SHOULD request only the agent claims necessary for their access control decisions.

- The "agent_trust_score" claim provides finer-grained information than "agent_trust_level". When the Relying Party’s access control decisions are based solely on trust levels, the "agent_trust_score" SHOULD be omitted to reduce information disclosure.

- The "agent_capabilities" claim SHOULD be filtered to include only capabilities relevant to the Relying Party, rather than the complete set of the Agent’s capabilities.

10. IANA Considerations

10.1. Claims Registration

This specification registers the following claims in the IANA "JSON Web Token Claims" registry established by [RFC7519]:

Claim Name: agent_id
Claim Description: Unique identifier for an autonomous software agent
Change Controller: IESG
Specification Document(s): Section 4.1 of this specification

Claim Name: agent_name
Claim Description: Human-readable display name for an agent
Change Controller: IESG
Specification Document(s): Section 4.2 of this specification

Claim Name: agent_owner
Claim Description: Subject identifier of the entity controlling the agent
Change Controller: IESG
Specification Document(s): Section 4.3 of this specification

Claim Name: agent_trust_score
Claim Description: Numerical trust score for agent behavioural trustworthiness
Change Controller: IESG
Specification Document(s): Section 4.4 of this specification

Claim Name: agent_trust_level
Claim Description: Graduated trust classification for an agent
Change Controller: IESG
Specification Document(s): Section 4.5 of this specification

Claim Name: agent_capabilities
Claim Description: Set of actions an agent is authorised to perform
Change Controller: IESG
Specification Document(s): Section 4.6 of this specification

Claim Name: agent_sanctions_status
Claim Description: Result of sanctions screening on an agent or its owner
Change Controller: IESG
Specification Document(s): Section 4.7 of this specification

Claim Name: agent_spend_limit
Claim Description: Maximum single transaction amount for an agent in minor currency units
Change Controller: IESG
Specification Document(s): Section 4.8 of this specification

Claim Name: agent_attestation_method
Claim Description: Method used to verify agent identity
Change Controller: IESG
Specification Document(s): Section 4.9 of this specification

Claim Name: agent_created_at
Claim Description: Unix timestamp of agent creation or registration
Change Controller: IESG
Specification Document(s): Section 4.10 of this specification

10.2. OAuth Dynamic Client Registration Metadata Registration

This specification registers the following client metadata parameter in the IANA "OAuth Dynamic Client Registration

Metadata" registry:

Client Metadata Name: agent_claims_supported
Client Metadata Description: Boolean indicating whether the OpenID Provider supports issuing agent identity claims
Change Controller: IESG
Specification Document(s): Section 5 of this specification

11. References

11.1. Normative References

[OpenID.Core]
Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0 incorporating errata set 2", December 2023, <https://openid.net/specs/openid-connect-core-1_0.html>.

[OpenID.Discovery]
Sakimura, N., Bradley, J., Jones, M., and E. Jay, "OpenID Connect Discovery 1.0 incorporating errata set 2", December 2023, <https://openid.net/specs/openid-connect-discovery-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/info/rfc2119>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[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/info/rfc7519>.

[RFC7523] Jones, M., Campbell, B., and C. Mortimore, "JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants", RFC 7523, DOI 10.17487/RFC7523, May 2015, <https://www.rfc-editor.org/info/rfc7523>.

[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/info/rfc8174>.

[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/info/rfc9449>.

11.2. Informative References

[I-D.sharif-agent-payment-trust]
Sharif, R., "Agent Payment Trust Framework",
Internet-Draft
draft-sharif-agent-payment-trust-00, March 2026,
<https://datatracker.ietf.org/doc/
draft-sharif-agent-payment-trust/>.

[I-D.sharif-mcps-secure-mcp]
Sharif, R., "MCPS: Secure Model Context Protocol",
Internet-Draft
draft-sharif-mcps-secure-mcp-00, March 2026,
<https://datatracker.ietf.org/doc/
draft-sharif-mcps-secure-mcp/>.

[OWASP.MCP]
OWASP Foundation, "MCP Security Cheat Sheet", 2026,
<https://cheatsheetseries.owasp.org/cheatsheets/
MCP_Security_Cheat_Sheet.html>.

[NIST.AI.Agent]
National Institute of Standards and Technology,
"NCCoE AI Agent Identity Concept Paper", 2026,
<https://www.nist.gov/artificial-intelligence>.

Appendix A. Acknowledgements

The author gratefully acknowledges the contributions of the OpenID Foundation AI Identity Management Community Group members whose review and feedback shaped this specification.

This work builds on the agent payment trust framework described in [I-D.sharif-agent-payment-trust] and incorporates security considerations from the OWASP MCP Security Cheat Sheet [OWASP.MCP] Section 7, and security guidance for MCP deployments from [I-D.sharif-mcps-secure-mcp].

Appendix B. Document History

draft-sharif-openid-agent-identity-00 -- March 2026

- Initial draft.
- Defined ten agent identity claims with full type definitions and validation rules.
- Specified Agent ID Token structure and Agent IdP metadata.
- Defined client credentials flow with challenge-response extension.
- Defined trust level requirements per action scope.
- Defined token validation procedures including trust score thresholds and sanctions enforcement.

Author’s Address

Raza Sharif
CyberSecAI Ltd
London, United Kingdom

Email: contact@agentsign.dev
URI: https://cybersecai.co.uk