

Transaction Tokens For Agents
draft-araut-oauth-transaction-tokens-for-agents-01

Abstract

This document specifies an extension to the OAUTH-TXN-TOKENS (<https://drafts.oauth.net/oauth-transaction-tokens/draft-ietf-oauth-transaction-tokens.html>) to support agent context propagation within Transaction Tokens for agent-based workloads. The extension defines the use of the act field to identify the agent performing the action, and leverages the existing sub field (as defined in the base Transaction Tokens specification) to represent the principal. The sub field is populated according to the rules specified in OAUTH-TXN-TOKENS (<https://drafts.oauth.net/oauth-transaction-tokens/draft-ietf-oauth-transaction-tokens.html>), based on the 'subject_token' provided in the token request. For autonomous agents operating independently, the sub field represents the agent itself. These mechanisms enable services within the call graph to make more granular access control decisions, thereby enhancing security.

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://ashayraut.github.io/oauth-transactiontokens-for-agents/draft-oauth-transaction-tokens-for-agents.html>. Status information for this document may be found at <https://datatracker.ietf.org/doc/draft-araut-oauth-transaction-tokens-for-agents/>.

Source for this draft and an issue tracker can be found at <https://github.com/ashayraut/oauth-transactiontokens-for-agents>.

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 13 November 2026.

Copyright Notice

Copyright (c) 2026 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	3
1.1. Conventions and Terminology	4
2. Terminology	4
3. Protocol overview	5
3.1. Transaction Flow	5
3.2. Agent Application Transaction Flows	5
3.2.1. Principal-Initiated Flow	5
3.2.2. Autonomous Flow	6
3.3. Flow Diagrams	7
3.3.1. Principal-Initiated Flow	7
3.3.2. Autonomous Flow	8
3.4. Replacement tokens	9
3.5. Txn-Token Format	10
3.5.1. JWT Header	10
3.5.2. JWT Body Claims	10
3.6. Agentic Context	11
3.6.1. Field definitions and population	11
3.6.2. Example of <code>agentic_ctx</code> with additional context	12
3.6.3. Implementation Note: Integrity and Resolution	13
4. Multi-agent flows	13
4.1. The Bifurcated Trust Model	13
4.2. Monotonic Attenuation of Trust	14
4.2.1. Delegation via Replacement Flow	14
4.2.2. Multi-agent example JWT body claims	14
4.2.3. Loop prevention	15

5. Security Considerations	16
6. References	19
6.1. Normative References	19
Appendix A. Acknowledgments	20
Appendix B. Contributors	20
Author's Address	20

1. Introduction

Traditional zero trust authorization systems face new challenges when applied to AI agent workloads. Unlike conventional web services, AI agents possess capabilities for autonomous operation, behavioral adaptation, and dynamic integration with various data sources. These characteristics may lead to decisions that extend beyond their initial operational boundaries.

Existing zero trust models, which effectively manage permissions and access scopes for traditional web services, require enhancement to address the unique properties of AI agents. Authorization systems must evaluate each AI agent interaction independently, considering both the immediate context and intended action. This necessitates more sophisticated approaches to policy enforcement, behavioral monitoring, and audit tracking to maintain security governance.

Transaction Tokens (Txn-Tokens) are short-lived, signed JSON Web Tokens RFC7519 (<https://tools.ietf.org/html/rfc7519>) that convey identity and authorization context. However, the current Txn-Token format lacks sufficient context for services within the call chain to implement fine-grained access control policies for agent-based workflows. Specifically, it does not provide adequate information about the AI agent's identity or its initiating entity, limiting transaction traceability. With this extension, Transaction Tokens will carry agent identity information which will help in better traceability for AI Agent's actions deep down the web service graph connecting multiple web services involved in completing a transaction in distributed systems.

This document defines three new contexts within the Transaction Token to address these limitations:

1. The act claim, which identifies the AI agent performing the action, aligning with OAuth 2.0 Token Exchange RFC8693 (<https://tools.ietf.org/html/rfc8693>) terminology for actor tokens
2. The sub claim, as defined in OAUTH-TXN-TOKENS (<https://drafts.oauth.net/oauth-transaction-tokens/draft-ietf-oauth-transaction-tokens.html>), which represents the principal on

whose behalf the transaction is being performed. The population of this field follows the rules specified in the base Transaction Tokens specification, based on the 'subject_token' provided in the token request.

3. An optional `agentic_ctx` claim. The value of this claim, if present, MUST be a JSON object. The `agentic_ctx` claim conveys attributes about the agent and its operational constraints that are relevant to authorization, auditing, and policy evaluation.

This extension leverages the existing Txn-Token infrastructure to enable secure propagation of AI agent context throughout the service graph.

1.1. Conventions and Terminology

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 (<https://datatracker.ietf.org/doc/html/rfc2119>) RFC8174 (<https://datatracker.ietf.org/doc/html/rfc8174>) when, and only when, they appear in all capitals, as shown here.

2. Terminology

Agentic-AI: AI Agentic applications are software applications that utilize Large Language Models (LLM)s and plans, reasons, and takes actions independently to achieve complex, multi-step goals with minimal human oversight.

Workload: An independent computational unit that can autonomously receive and process invocations, and can generate invocations of other workloads. Examples of workloads include containerized microservices, monolithic services and infrastructure services such as managed databases.

Trust Domain: A collection of systems, applications, or workloads that share a common security policy. In practice this may include a virtually or physically separated network, which contains two or more workloads. The workloads within a Trust Domain may be invoked only through published interfaces.

Call Chain: A sequence of synchronous invocations that results from the invocation of an external endpoint.

External Endpoint: A published interface to a Trust Domain that results in the invocation of a workload within the Trust Domain. This is the first service in the call chain where request starts.

Transaction Token (Txn-Token): A signed JWT with a short lifetime, providing immutable information about the user or workload, certain parameters of the call, and specific contextual attributes of the call. The Txn-Token is used to authorize subsequent calls in the call chain.

Transaction Token Service (Txn-Token Service): A special service within the Trust Domain that issues Txn-Tokens to requesting workloads. Each Trust Domain using Txn-Tokens MUST have exactly one logical Txn-Token Service.

3. Protocol overview

3.1. Transaction Flow

This section describes the process by which an agent application obtains a Transaction Token, either acting autonomously or on behalf of a principal. The external endpoint requests a Txn-Token following the procedures defined in OAUTH-TXN-TOKENS (<https://drafts.oauth.net/oauth-transaction-tokens/draft-ietf-oauth-transaction-tokens.html>), augmented with additional context for agent identity and, when applicable, principal identity.

3.2. Agent Application Transaction Flows

The Transaction Token creation process varies depending on the presence of a principal.

3.2.1. Principal-Initiated Flow

When a principal initiates the workflow, the following steps occur:

1. The principal invokes the agent application to perform a task.
2. The agent application calls an external endpoint. External endpoint throws back OAuth challenges.
3. The agent application authenticates using an OAuth 2.0 Auth code flow RFC6749 (<https://tools.ietf.org/html/rfc6749>) access token. The access token contains subject and clientId claims as per RFC9068 (<https://datatracker.ietf.org/doc/rfc9068>).
4. The external endpoint submits the received access token along with its Subject token to the Txn-Token Service. Subject token requirements are specified in OAUTH-TXN-TOKENS (<https://drafts.oauth.net/oauth-transaction-tokens/draft-ietf-oauth-transaction-tokens.html>).

5. The Txn-Token Service validates the access token.
6. The Txn-Token Service populates the Txn-Token's sub claim following the rules specified in OAUTH-TXN-TOKENS (<https://drafts.oauth.net/oauth-transaction-tokens/draft-ietf-oauth-transaction-tokens.html>). The sub claim is determined based on the subject_token provided in the request, according to the conditions and rules defined in the base Transaction Tokens specification. This ensures that the principal is properly represented in the Txn-Token.
7. The Txn-Token Service copies the access token's clientId claim to the Txn-Token's act field. Any nested structure within the clientId claim is preserved. If the access token contains an act claim, that value MAY be used instead of clientId.
8. The Txn-Token Service issues the Txn-Token to the requesting workload.

3.2.2. Autonomous Flow

When the agent application operates autonomously, the following steps occur:

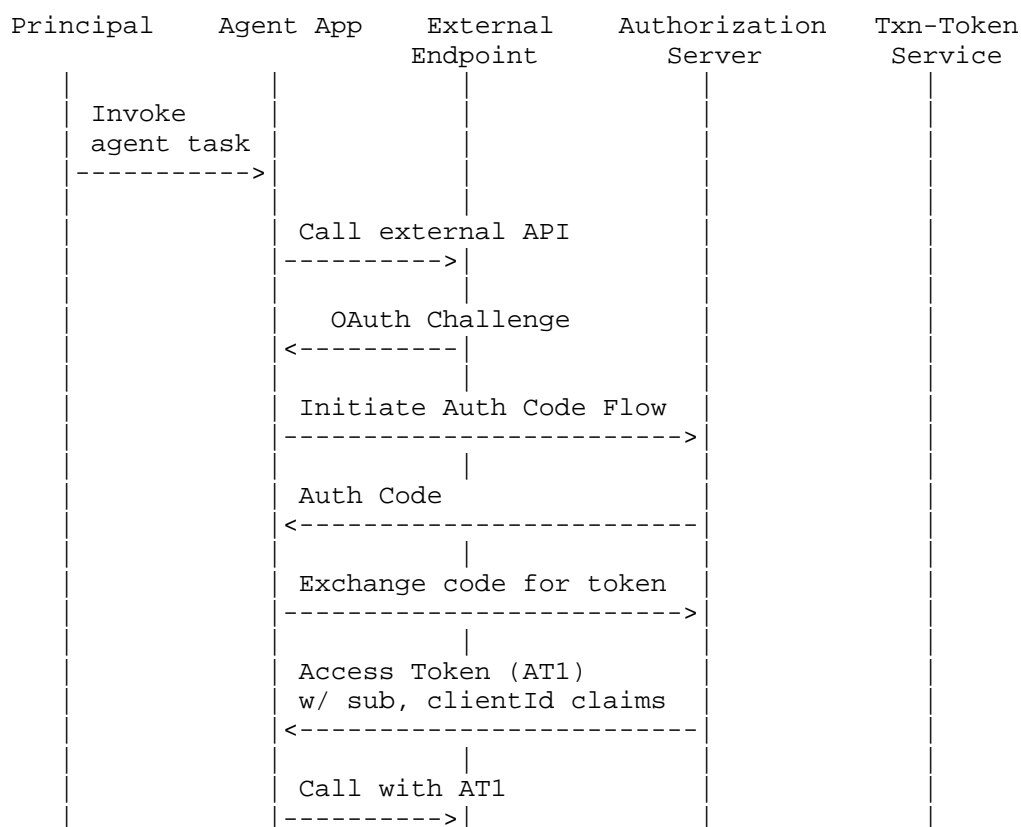
1. The agent application initiates a task based on an event or scheduled assignment.
2. The agent application calls an external endpoint. OAuth challenge flow starts.
3. The agent application authenticates using an OAuth 2.0 RFC6749 (<https://tools.ietf.org/html/rfc6749>). When an autonomous agent (no human resource owner) needs to call another resource server using OAuth, it follows the Client Credentials Grant defined explicitly in RFC6749 (<https://tools.ietf.org/html/rfc6749>).
4. The agent application uses the access token to call the external endpoint.
5. The external endpoint submits the received access token along with its Subject token to the Txn-Token Service. Subject token requirements are specified in OAUTH-TXN-TOKENS (<https://drafts.oauth.net/oauth-transaction-tokens/draft-ietf-oauth-transaction-tokens.html>).
6. The Txn-Token Service validates the access token.

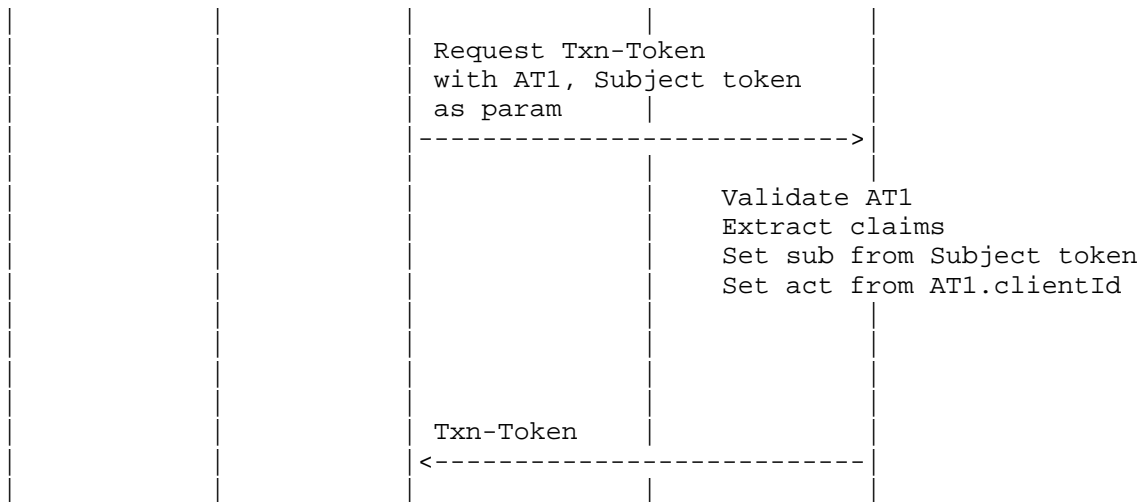
7. The Txn-Token Service populates the Txn-Token's sub claim following the rules specified in OAUTH-TXN-TOKENS (<https://drafts.oauth.net/oauth-transaction-tokens/draft-ietf-oauth-transaction-tokens.html>). The sub claim is determined based on the subject_token provided in the request. For autonomous agents, this typically represents the agent's own identity.
8. The Txn-Token Service copies the access token's sub or clientId claim to the Txn-Token's act field. Any nested structure is preserved. The act field identifies the agent performing the autonomous action.

3.3. Flow Diagrams

3.3.1. Principal-Initiated Flow

Based on the updated flow, here's a more detailed RFC-style flow diagram:





Legend:

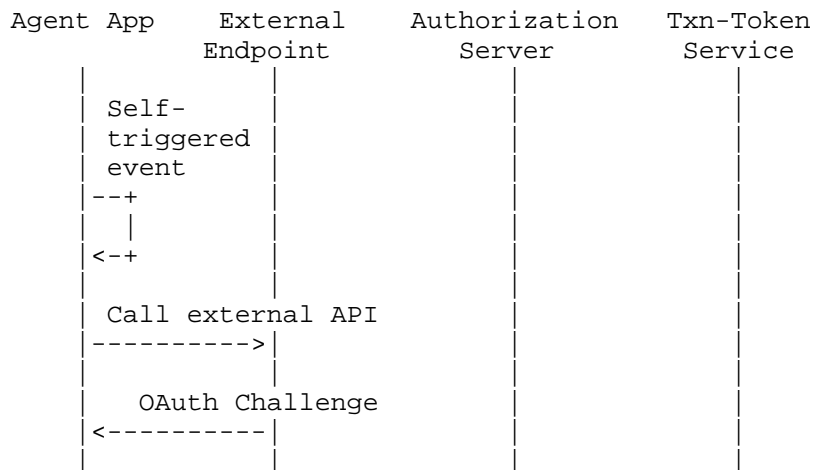
----> : Request flow

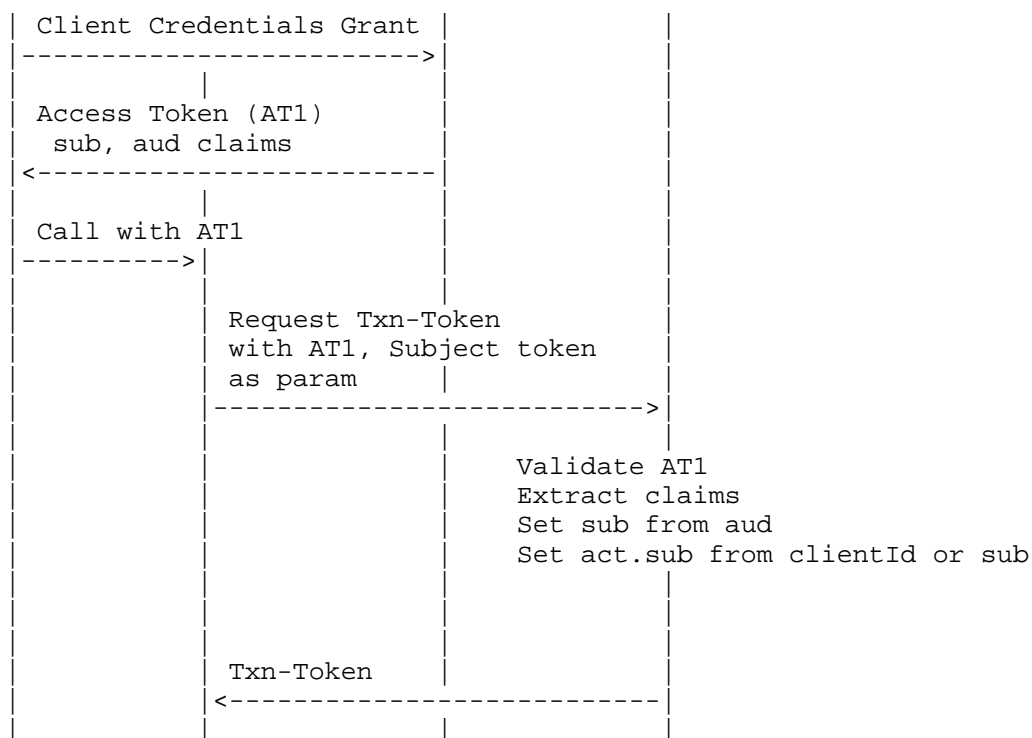
<---- : Response flow

| : Component boundary

Notes: 1. AT1 refers to the access token obtained by Agent App 2. The External Endpoint uses its own access token to call Txn-Token Service 3. AT1 is passed as a parameter in the Txn-Token request 4. The flow shows detailed OAuth 2.0 Authorization Code flow steps 5. Token validation and claim extraction steps are shown in the Txn-Token Service

3.3.2. Autonomous Flow





Legend:

----> : Request flow
 <---- : Response flow
 | : Component boundary
 + : Internal process
 --+ : Self-triggered event

Notes:

- * AT1: Access token obtained via Client Credentials Grant
- * External Endpoint uses subject token for authenticating itself to Txn-Token Service
- * AT1 is included as parameter in Txn-Token request
- * Self-triggered events can be scheduled tasks or external triggers
- * Token validation includes signature and claims verification

3.4. Replacement tokens

Txn-Token Service provides capability to get a replacement Txn-Token as defined in the OAUTH-TXN-TOKENS.replacement flow (<https://drafts.oauth.net/oauth-transaction-tokens/draft-ietf-oauth-transaction-tokens.html#name-creating-replacement-txn-to>). If the original Txn-Token used to get replacement token contains 'actor' and 'principal' claims then in the replaced Txn-Token, the values of the

'actor' and 'principal' MUST remain unchanged similar to 'txn', sub and 'aud' claims.

3.5. Txn-Token Format

3.5.1. JWT Header

No changes to the JWT header from the base specification: typ MUST be txntoken+jwt, with a signing key identifier such as kid.

3.5.2. JWT Body Claims

The Txn-Token body augments the base claim set with the act field for agent context. Existing claims like txn, sub, aud, iss, iat, exp, scope, tctx, and req_wl retain identical semantics, population rules, and immutability guarantees as defined in OAUTH-TXN-TOKENS (<https://drafts.oauth.net/oauth-transaction-tokens/draft-ietf-oauth-transaction-tokens.html>).

In this example, the agent is 3rd party and not part of trust domain. It hits API Gateway in trust domain and API Gateway requests Txn-Token from Txn-token Service using access token received from 3P agent and its own subject token (to authenticate with Txn-Token Service). Requesting workload is API Gateway. Agent is agent-identity-1 (clientId in the access token issued to 3P agent to act on behalf of user:alice)

```
{
  "txn": "c2dc3992-2d65-483a-93b5-2dd9f02c276e",
  "sub": "user:alice@example.com", // if its human initiated
  "aud": "https://trading.trust-domain.example/stocks",
  "iss": "https://txn-svc.trust-domain.example",
  "iat": 1697059200,
  "exp": 1697059500,
  "purp": "trade.stocks",
  "tctx": {
    "action": "BUY",
    "ticker": "MSFT",
    "quantity": "100"
  },
  "req_wl": "apigateway.trust-domain.example", // API gateway requests Txn-token
  "act": {
    "sub": "agent-identity-1" // 3P agent hitting API gateway owned by trust domain
  }
}
```

3.6. Agentic Context

The Txn-Token MAY contain an `agentic_ctx` claim. Txn-Tokens are increasingly used in environments where transactions are executed by or with the assistance of autonomous or semi-autonomous agents (for example, Large Language Model (LLM) 纂澥ased agents, workflow orchestrators, and policy-driven automation components). In such deployments, relying exclusively on subject identity and generic transaction parameters is insufficient to make robust authorization decisions. Additional information about the agent that is interpreting and acting on the transaction is often required.

3.6.1. Field definitions and population

All fields are OPTIONAL as some of these fields may not be available for 3P (third party agents) connecting to your external service (edge) within trust domain. However, for internal agents within trust domain, you MAY get following information. For 3P agents outside your trust domain, as called out above, you will be able to get act claim information using access token and rest of the below fields in `agentic_context` may or may not be available.

- * `*prov (Provenance)*`: Defines the "Behavioral DNA" of the agent. The `manifest_hash` is a cryptographic digest of the agent 纂洳 system instructions and core logic, ensuring the agent 纂洳 "guardrails" have not been modified. The `manifest_hash` is an opaque key. Resource Servers are expected to resolve this hash against a local or remote policy store to determine the specific behavioral guardrails applied to the agent at that version.
- * `*posture (Environmental Integrity)*`: Details the security tier of the runtime. This includes hardware-backed proof (e.g., TEE) that the agent is isolated from the host OS or cloud provider.
- * `*identity (Workload Origin)*`: Captures the specific machine-actor instance. The `workload_id` distinguishes the instrument (the agent software) from the subject (the end-user).

The `agentic_ctx` claim is populated during the token exchange process by the Transaction Token Service (TTS), which serves as the authoritative source for the agent 纂洳 operational identity. This context MAY be derived from various but not limited to sources such as : (1) Static Manifests, which include cryptographic hashes of the agent 纂洳 system prompt, model configuration, and registered source code identifiers; (2) Environmental Posture, consisting of telemetry injected by the execution environment or API gateway, such as the hardware security level (e.g., TEE measurements) or network origin; and (3) Workload Mapping, where attributes are mirrored from the

agent's primary workload identity (e.g., SPIFFE or OIDC claims). By centralizing this population at the TTS, the `agentic_ctx` provides a consistent and tamper-resistant representation of the agent's "persona." This allows downstream resource servers to evaluate the agent's integrity and origin against local safety policies independently of the specific actions or permissions requested in the transaction.

To ensure the integrity of the `agentic_ctx`, the Transaction Token Service (TTS) MUST not rely on self-reported data from the agent. Instead, it populates these fields through a Verified Exchange model. Hardware-backed fields like posture and tee are derived from cryptographic Attestation Documents generated by the agent's execution environment (e.g., a Trusted Execution Environment) and verified by the TTS against cloud provider roots of trust. Software-related fields, such as the `manifest_hash`, are retrieved via a Registry-First approach: the TTS performs an out-of-band lookup in a secure Agent Registry using the agent's authenticated `client_id`, ensuring that the "behavioral fingerprint" in the token matches the developer's registered configuration rather than a potentially tampered runtime state. Finally, the identity claims are mirrored from the transport layer (e.g., mTLS certificates or SPIFFE SVIDs), binding the token to the specific verified workload instance.

3.6.2. Example of `agentic_ctx` with additional context

```
{
  "agentic_ctx": {
    "prov": {
      "manifest_hash": "sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991
b7852b855",
      "model_id": "llama-3.1-70b-v1",
      "version": "2.4.1"
    },
    "posture": {
      "tee": "aws-nitro-enclave",
      "assurance": "high",
      "boot_gold": true
    },
    "identity": {
      "workload_id": "spiffe://prod.acme.com/billing-agent",
      "origin_node": "node-77-east-1"
    }
  }
}
```

3.6.3. Implementation Note: Integrity and Resolution

The fields within `agentic_ctx` represent a "Statement of Posture" rather than a set of permissions. To avoid authorization failure, implementations should ensure that:

- * **Registry Synchronization**: The `manifest_hash` is treated as a lookup key. Resource Servers should maintain or have access to a known-good database of hashes to map cryptographic signatures to behavioral guardrails.
- * **Hardware Roots of Trust**: When posture claims are present, the Transaction Token Service MUST verify the underlying attestation document against the hardware manufacturer's public keys.
- * **Non-Collusion**: The `agentic_ctx` is distinct from the sub claim. While the sub identifies the authorizing principal, the `agentic_ctx` identifies the machine-actor. Authorization logic SHOULD evaluate the intersection of both identities.

4. Multi-agent flows

In complex agentic workflows, a transaction often originates from a 3rd-party (3P) agent and propagates through one or more 1st-party (1P) agents within the local trust domain. To maintain Zero Trust integrity, this specification uses a structured Postured Lineage within the `agentic_ctx`. This ensures that downstream Resource Servers can evaluate the security posture of the entire chain, rather than relying solely on the identity of the immediate caller.

4.1. The Bifurcated Trust Model

The `agentic_ctx` differentiates between two types of actors in a chain:

- * **The Originator (External/3P)**: The entry point of the request into the trust domain. Because the hardware and software of 3P agents are outside local control, their context is Asserted via identity federation. Posture and provenance fields for these actors are typically marked as unverified or none.
- * **The Current Actor (Internal/1P)**: The agent currently executing the request. For internal agents, the context is Verified by the Transaction Token Service (TTS) using hardware attestation and registry-based manifest lookups.

4.2. Monotonic Attenuation of Trust

A chain's security posture is only as strong as its weakest link. The TTS MUST calculate a `min_assurance_level` during every token replacement flow. If a "High-Trust" internal agent is triggered by a "Low-Trust" 3P originator, the transaction's overall assurance level remains low. This prevents Identity Laundering, where unverified external agents bypass security guardrails by proxying requests through internal services. The Transaction Token Service (TTS) determines the `min_assurance_level` by performing a comparative risk analysis during the token replacement flow. It essentially identifies the "weakest link" in the execution chain by comparing the posture of the incoming token with the verified posture of the new requesting agent.

4.2.1. Delegation via Replacement Flow

When an internal agent (the "Delegatee") requires a Transaction Token to continue a chain initiated by another actor (the "Delegator"), it MUST follow the replacement flow procedures defined in OAUTH-TXN-TOKENS ([https://drafts.ietf.org/oauth-txn-tokens.html](https://drafts.ietf.org/oauth-txn-tokens)) with the following modifications:

- * ***Subject Immutability***: The `txn` and `sub` (principal) claims MUST be copied from the `subject_token` to the new Transaction Token without modification.
- * ***Lineage Preservation***: The TTS MUST extract the identity of the Delegator from the incoming token and append it to the `txn_path` array.
- * ***Context Enrichment***: The TTS MUST populate the `current_actor` object with verified telemetry (TEE posture, manifest hashes) corresponding to the Delegatee.

4.2.2. Multi-agent example JWT body claims

This example represents a delegated state: a 3rd-party Assistant (3p-assistant-ext-99) has initiated a task, which is now being executed by an internal, 1st-party Billing Agent (1p-billing-svc-v2) running in a secure enclave.

```

{
  "txn": "abc-123-xyz",
  "sub": "user_8821@example.com",
  "iss": "[https://txn-svc.trust-domain.example](https://txn-svc.trust-domain.example)
",
  "iat": 1712850000,
  "exp": 1712850300,
  "req_wl": "apigateway.trust-domain.example", // API gateway requests Txn-token
  "agentic_ctx": {
    "current_actor": {
      "identity": {
        "workload_id": "1p-billing-svc-v2",
        "origin_node": "internal-cluster-alpha-node-4"
      },
      "posture": {
        "tee": "aws-nitro-enclave",
        "assurance": "high"
      },
      "prov": {
        "manifest_hash": "sha256:4455..."
      }
    },
    "originator": {
      "identity": { "workload_id": "3p-assistant-ext-99" },
      "posture": {
        "tee": "unverified",
        "assurance": "low"
      },
      "prov": { "manifest_hash": "none" }
    },
    "chain_metadata": {
      "hop_count": 2,
      "min_assurance_level": "low",
      "txn_path": ["3p-assistant-ext-99", "1p-billing-svc-v2"]
    }
  }
}

```

4.2.3. Loop prevention

To prevent infinite recursion in autonomous agentic loops, the `txn_path` MUST be updated at every hop. The TTS MUST append the current `workload_id` to the path. If an agent receives a token where its own `workload_id` is already present in the `txn_path`, the request MUST be rejected.

5. Security Considerations

1. All the security considerations mentioned in OAUTH-TXN-TOKENS (<https://drafts.oauth.net/oauth-transaction-tokens/draft-ietf-oauth-transaction-tokens.html>) apply.
2. Token Replay Protection Implementations MUST enforce strict token lifetime validation. The short-lived nature of Transaction Tokens helps mitigate replay attacks, but implementations SHOULD also consider:
 - * Implementing token tracking mechanisms within trust domains
 - * Validating token usage context
3. Actor Identity Security
 - * Implementations MUST validate act claims in tokens
 - * The Txn-Token Service MUST verify the authenticity of actor context before token issuance
 - * During replacement flow, Txn-Token Service MUST NOT modify the act field in the incoming Txn-Token
4. Principal Context Protection
 - * Systems MUST prevent unauthorized modifications to the sub claim during token propagation. Txn-Tokens are cryptographically signed to ensure integrity.
 - * During replacement flow, Txn-Token Service MUST NOT modify the sub claim in the incoming Txn-Token
 - * The Txn-Token Service MUST follow the subject population rules defined in OAUTH-TXN-TOKENS (<https://drafts.oauth.net/oauth-transaction-tokens/draft-ietf-oauth-transaction-tokens.html>) to ensure proper principal representation
5. Transaction Chain Integrity
 - * Implementations MUST maintain cryptographic integrity of the token chain
 - * Services MUST validate tokens at trust domain boundaries
 - * Systems MUST implement protection against token tampering during service-to-service communication

6. AI Agent Specific Controls

- * Implementations MUST enforce scope boundaries for AI agent operations
- * Systems SHOULD implement behavioral monitoring for AI agent activities by logging act and sub claims in audit logs
- * Systems MUST maintain audit trails of AI agent activities

7. Token Transformation Security

- * The Txn-Token Service MUST validate all claims during access token to Txn-Token conversion
- * Implementations MUST verify signatures and formats of all tokens
- * Systems MUST prevent unauthorized manipulation during token transformation
- * The Txn-Token Service MUST ensure that the act field accurately represents the agent identity from the access token

8. Replacement Token Considerations

- * Systems MUST verify the authenticity and validity of original tokens before replacement
- * Systems MUST implement controls to prevent unauthorized replacement requests
- * The immutability of act and sub claims during replacement ensures consistent identity context throughout the transaction lifecycle

9. Infrastructure Security

- * All component communications MUST use secure channels
- * Implementations MUST enforce strong authentication of the Authorization Server
- * Systems MUST implement regular rotation of cryptographic keys
- * Trust domain boundaries MUST be clearly defined and enforced

10. Prevention of Identity Laundering

- * Implementations MUST enforce Monotonic Attenuation of the `min_assurance_level`.
- * The TTS MUST NOT allow a replacement token to have a higher assurance level than the incoming subject token, even if the current actor is running in a High-Assurance environment.
- * This prevents a low-trust 3rd-party originator from "laundering" its identity through a high-trust internal agent to bypass security guardrails at the Resource Server.

11. Integrity of the Agent Registry

- * The security of the prov (Provenance) claims relies entirely on the integrity of the Agent Registry. If an attacker can modify the registry to associate a malicious `manifest_hash` with a legitimate `workload_id`, they can trick the TTS into asserting high software integrity for tampered code.
- * Access to the Agent Registry MUST be restricted to authorized deployment pipelines and protected with strong integrity controls.

12. Hardware Attestation Forgery

- * When posture claims indicate the use of a Trusted Execution Environment (TEE), the TTS MUST verify the underlying Attestation Document against the hardware manufacturer's Root of Trust.
- * Failure to verify these signatures allows a compromised host to spoof a "High" assurance posture, leading to unauthorized access to sensitive data.

13. Loop Detection and Recursion Limits

- * The use of the `txn_path` is REQUIRED to prevent infinite recursion in autonomous agentic workflows.
- * Resource Servers and the TTS SHOULD also enforce a maximum `hop_count` to prevent resource exhaustion attacks. If the path exceeds a defined threshold, the transaction MUST be terminated.

14. Data Leakage in Lineage Propagation

- * The `agentic_ctx` may contain sensitive internal information, such as `origin_node` or specific `workload_id` structures.
- * When a 1st-party agent calls an external 3rd-party service, the TTS MUST strip these internal-only fields from the token to prevent infrastructure leakage.
- * Only the minimum necessary identity context should be egressed from the trust domain.

6. References

6.1. Normative References

RFC2119 (<https://datatracker.ietf.org/doc/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> (<https://www.rfc-editor.org/rfc/rfc2119>).

RFC8174 (<https://datatracker.ietf.org/doc/html/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>

RFC6749 (<https://tools.ietf.org/html/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> (<https://www.rfc-editor.org/rfc/rfc6749>).

RFC7519 (<https://tools.ietf.org/html/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> (<https://www.rfc-editor.org/rfc/rfc7519>).

RFC7515 (<https://tools.ietf.org/html/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/rfc/rfc7515> (<https://www.rfc-editor.org/rfc/rfc7515>).

RFC8693 (<https://tools.ietf.org/html/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> (<https://www.rfc-editor.org/rfc/rfc8693>).

RFC9068 (<https://tools.ietf.org/html/rfc9068>) Bertocci, V., "JSON Web Token (JWT) Profile for OAuth 2.0 Access Tokens", RFC 9068, DOI 10.17487/RFC9068, October 2021, <https://www.rfc-editor.org/rfc/rfc9068> (<https://www.rfc-editor.org/rfc/rfc9068>).

RFC9396 (<https://datatracker.ietf.org/doc/html/rfc9396>) T. Lodderstedt, J. Richer, B. Campbell, "OAuth 2.0 Rich Authorization Requests", RFC 9396, DOI 10.17487/RFC9396, May 2023, <https://www.rfc-editor.org/rfc/rfc9396> (<https://www.rfc-editor.org/rfc/rfc9396>).

OAUTH-TXN-TOKENS (<https://datatracker.ietf.org/doc/draft-tulshibagwale-oauth-transaction-tokens>) Atul Tulshibagwale, George Fletcher, Pieter Kasselmann, "OAuth Transaction Tokens", <https://drafts.oauth.net/oauth-transaction-tokens/draft-ietf-oauth-transaction-tokens.html> (<https://drafts.oauth.net/oauth-transaction-tokens/draft-ietf-oauth-transaction-tokens.html>)

Appendix A. Acknowledgments

name: Dr. Chunchi (Peter) Liu email: Liuchunchi(Peter)
<liuchunchi=40huawei.com@dmarc.ietf.org>

Appendix B. Contributors

name: Atul Tulshibagwale org: SGNL email: atul@sgnl.ai

Author's Address

ASHAY RAUT
Email: asharaut@amazon.com