Network Working Group
Internet-Draft
Intended status: Informational
Expires: 24 July 2026

A. RAUT
Amazon

20 January 2026

# Transaction Tokens For Agents
## draft-oauth-transaction-tokens-for-agents-03

## Abstract

This document specifies an extension to the OAuth Transaction Tokens framework (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 two new context fields: 'actor' and 'principal'. The 'actor' field identifies the agent performing the action, while the 'principal' field identifies the human or system entity that initiated the agent's action. For autonomous agents operating independently, the 'principal' field MAY be omitted. These additional context fields 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-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/.

## Copyright Notice



## Table of Contents

1. Introduction . . . . . 3
2. Terminology . . . . . 4
3. Protocol overview . . . . . 4
  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 . . . . . 6
    3.3.1. Principal-Initiated Flow . . . . . 6
    3.3.2. Autonomous Flow . . . . . 8
  3.4. Replacement tokens . . . . . 9
  3.5. Txn-Token Format . . . . . 9
    3.5.1. JWT Header . . . . . 9
    3.5.2. JWT Body Claims . . . . . 9
  3.6. Agentic Context . . . . . 10
4. Security Considerations . . . . . 10
5. References . . . . . 12
  5.1. Normative References . . . . . 12
Appendix A. Acknowledgments . . . . . 12
Appendix B. Contributors . . . . . 12
Author's Address . . . . . 12

## 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 two new contexts within the Transaction Token to address these limitations:

1. The actor context, which identifies the AI agent performing the action

2. The principal context, which identifies the human or system entity on whose behalf the actor operates

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

There is an opportunity here to add ’agentic context’ in the Txn Token too. The Txn-Token MAY contain an agentic_ctx claim. The value of this claim, if present, MUST be a JSON object. T The agentic_ctx claim conveys attributes about the agent and its operational constraints that are relevant to authorization, auditing, and policy evaluation.

## 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 to the Txn-Token Service. Note that this received access token is different rather the access token which external endpoint has available to call Txn-Token Service itself. So the received access token is actually a parameter required to call Txn-token Service

5. The Txn-Token Service validates the access token.

6. As specified in OAUTH-TXN-TOKENS (https://drafts.oauth.net/oauth-transaction-tokens/draft-ietf-oauth-transaction-tokens.html), the Txn-Token Service uses the access token’s ’aud’ claim to populate the Txn-Token’s ’sub’ claim.

7. The Txn-Token Service copies the access token’s ’actor’ or ’clientId’ claim to the Txn-Token’s ’actor’ context. Any nested structure within the ’actor’ claim is preserved.

8. The Txn-Token Service uses the access token’s ’sub’ claim to populate the Txn-Token’s ’principal’ context.

#### 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 to the Txn-Token Service. Note that this received access token is different rather the access token which external endpoint has available to call Txn-Token Service itself. So the received access token is actually a parameter required to call Txn-token Service

6. The Txn-Token Service validates the access token and extracts the actor and subject identities.

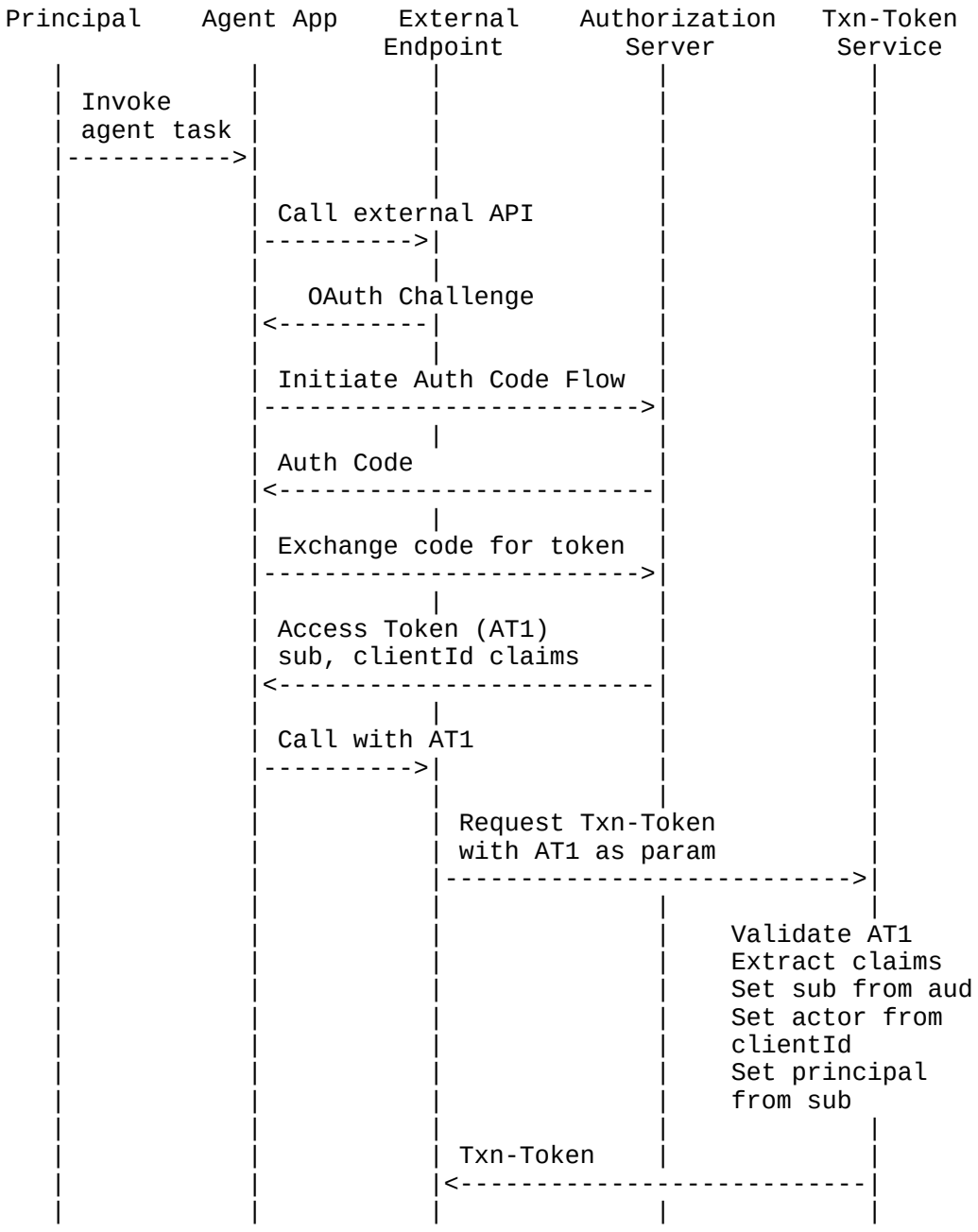
7. As specified in OAUTH-TXN-TOKENS (https://drafts.oauth.net/oauth-transaction-tokens/draft-ietf-oauth-transaction-tokens.html), the Txn-Token Service uses the access token’s ’aud’ claim to populate the Txn-Token’s ’sub’ claim.

8. The Txn-Token Service copies the ’sub’ field from within the access token’s ’actor’ claim to the Txn-Token’s ’actor’ context. Any nested structure is preserved.

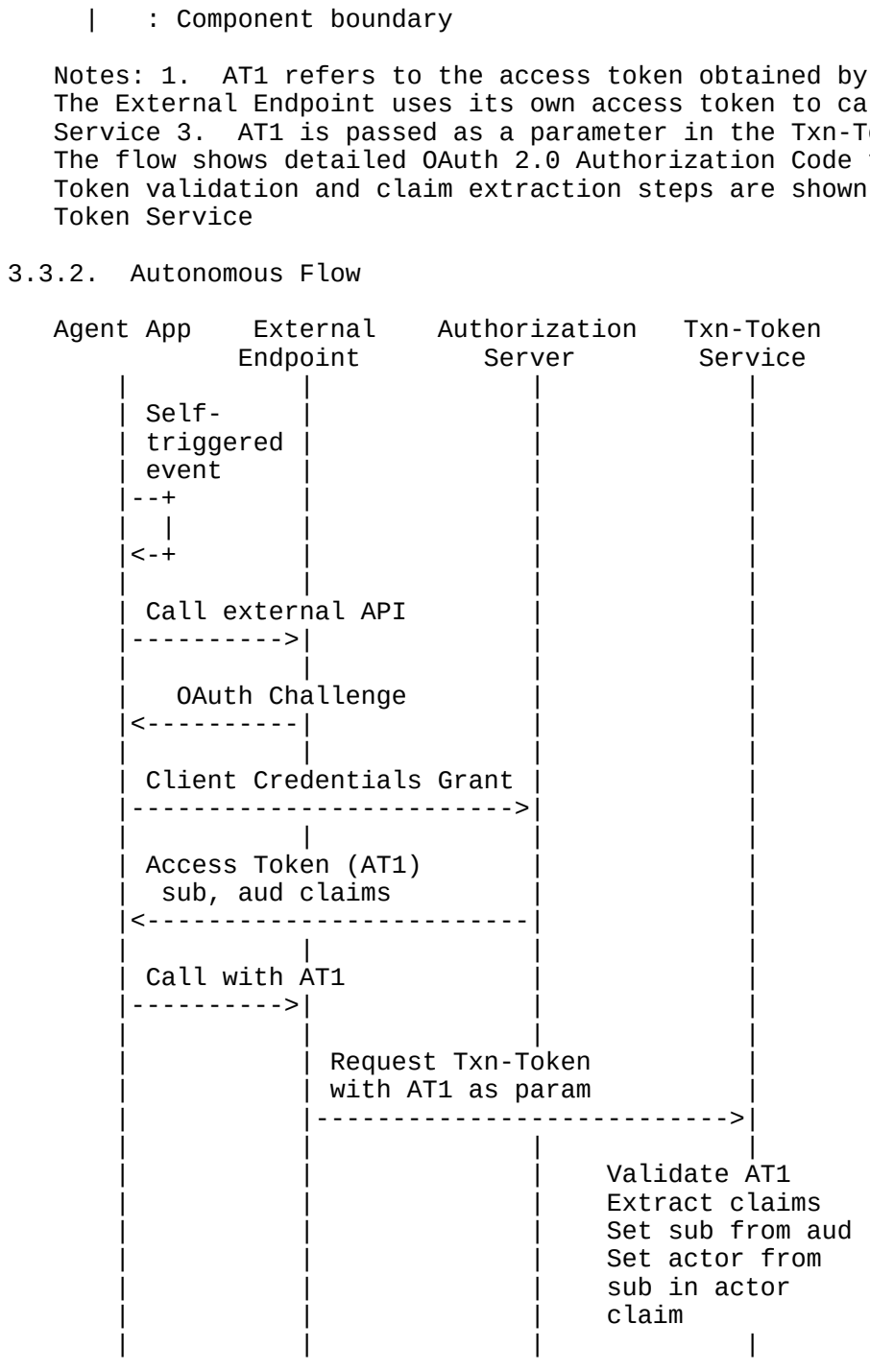
### 3.3. Flow Diagrams

#### 3.3.1. Principal-Initiated Flow

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

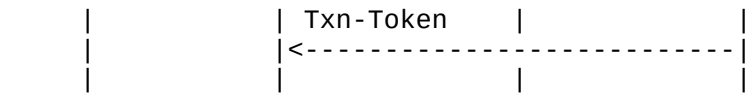



Legend:
----> : Request flow
<---- : Response flow




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 its own credentials for 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 two new top-level claims for agent context: actor and principal. Existing claims like txn, sub, aud, iss, iat, exp, purp, tctx, and req_wl retain identical semantics, population rules, and immutability guarantees.

```
{
  "txn": "c2dc3992-2d65-483a-93b5-2dd9f02c276e",
  "sub": "api-gw.trust-domain.example",
  "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",
  "actor": {
    "agent_id": "agent-1234",
    "version": "v2.1.0",
    "deployment": "prod-us-east-1"
  },
  "principal": "user:alice@example.com"
}
```

### 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)based 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.

```
"agentic_ctx": {
  "agent_type": "planner+tool-orchestrator", // A string describing the functional role of the agent (for example, “planner”, “tool-orchestrator”, “data-assistant”, “code-execution-agent”). The semantics and allowed values are deployment-specific.
  "agent_version": "3.4.2", // A string indicating a version or configuration identifier for the agent. This value can be used to associate the transaction with a particular, reviewed agent policy or release
  "intent": "enumerate and validate production search services before Q4 traffic spike", // A string describing the high-level purpose of the transaction from the agent’s perspective (for example, “trade.stocks”, “enumerate.search.services”, “generate.billing.report”). This value is intended to support coarse-grained, intent-aware authorization policies.
  "allowed_actions": ["read"],
  "environment_constraints": { "environment": "prod", "region": "us" },
}
```

## 4. 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: 2.1 Implementing token tracking mechanisms within trust domains 2.2 Validating token usage context

3. Actor Identity Security 3.1. Implementations MUST validate actor claims in tokens 3.2. The Txn-Token Service MUST verify the authenticity of actor context before token issuance 3.3. During replacement flow, Txn-Token Service MUST avoid replacing actor context in the incoming Txn-Token.

4. Principal Context Protection 4.1. Systems MUST prevent unauthorized modifications to principal context during token propagation. Txn-Token is cryptographically signed. 4.3. During replacement flow, Txn-Token Service MUST avoid replacing principal context in the incoming Txn-Token.

5. Transaction Chain Integrity 5.1. Implementations MUST maintain cryptographic integrity of the token chain 5.2. Services MUST validate tokens at trust domain boundaries 5.3. Systems MUST implement protection against token tampering during service-to-service communication

6. AI Agent Specific Controls 6.1. Implementations MUST enforce scope boundaries for AI agent operations 6.2. Systems SHOULD implement behavioral monitoring for AI agent activities by logging actor, principal in logs. 6.3. Systems MUST maintain audit trails of AI agent activities

7. Token Transformation Security 7.1. The Txn-Token Service MUST validate all claims during access token to Txn-Token conversion 7.2. Implementations MUST verify signatures and formats of all tokens 7.3. Systems MUST prevent unauthorized manipulation during token transformation

8. Replacement Token Considerations 8.1. Systems MUST verify the authenticity and validity of original tokens before replacement 8.2. Systems MUST implement controls to prevent unauthorized replacement requests

9. Infrastructure Security 9.1. All component communications MUST use secure channels 9.2. Implementations MUST enforce strong authentication of the Authorization Server 9.3. Systems MUST implement regular rotation of cryptographic keys 9.4. Trust domain boundaries MUST be clearly defined and enforced

# 5. References

## 5.1. Normative References

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).

OAUTH-TXN-TOKENS (https://datatracker.ietf.org/doc/draft-tulshibagwale-oauth-transaction-tokens) Atul Tulshibagwale, George Fletcher, Pieter Kasselman, "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

The authors would like to thank the contributors and the OAuth working group members who gave valuable input to this draft.

# Appendix B. Contributors

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

# Author’s Address

ASHAY RAUT
Amazon
Email: asharaut@amazon.com