WIMSE
Internet-Draft
Intended status: Informational
Expires: 29 August 2026

R. Krishnan
JPMorgan Chase & Co
A. Prasad
Oracle
D. Lopez
Telefonica
S. Addepalli
Aryaka

25 February 2026

Transitive Attestation for Sovereign Workloads: A WIMSE Profile
draft-mw-wimse-transitive-attestation-00

Abstract

This document defines a *WIMSE Profile* for Transitive Attestation within the Workload Identity in Multi-Service Environments (WIMSE) framework. It addresses the critical problem of *Identity Portability*, where software credentials (e.g., bearer tokens or keys) can be misappropriated and used from unauthorized environments—a risk amplified by the emergence of autonomous *AI Agents* that may move across jurisdictions or be hijacked via *prompt injection attacks*. By providing a standardized *Identity Conveyance* mechanism, this profile cryptographically binds software workloads to their local execution environment ("Proof of Residency") through a transitive chain of trust. This chain consumes *Evidence* from the underlying platform—supporting both *high-assurance* RATS-based profiles (e.g., [[!I-D.lkspa-wimse-verifiable-geo-fence]]) for residency verification and *standard* Workload Identity Agents for basic co-location proofs—to ensure that an identity is only valid when used from a verified, integral, or geographically compliant host. The integrity and hardware-rooted security of the Workload Identity Agent itself is considered out-of-scope for this document and is addressed in the *Verifiable Geofencing* profile [[!I-D.lkspa-wimse-verifiable-geo-fence]].

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

This document defines the *WIMSE Profile* for "Transitive Attestation", addressing a critical technical gap in the high-level *WIMSE Architecture* [[!I-D.ietf-wimse-arch]] regarding how platform-level trust is transitively extended to software workloads.

Currently, workload identities are often "context-agnostic"—once a credential (e.g., a bearer token) is issued, it can often be used from any environment. This *Identity Portability* allows an attacker who steals a token in one jurisdiction to use it from another, representing a significant *Sovereignty Violation* for workloads legally required to operate within specific boundaries. This is particularly critical for *AI Agents*, whose autonomous nature, susceptibility to hijacking via *prompt injection attacks*, and potential for rapid migration across cloud environments require strict, verifiable adherence to data residency and host integrity policies.

By addressing the *North-South* security axis of a workload's relationship with its local hosting environment, this profile establishes a *"Silicon-to-SVID"* chain of accountability. It ensures that the Workload Identity Agent (the local agent) is empowered to issue identities that are cryptographically bound to a verified execution context. This mechanism is flexible across assurance levels: it supports *High Assurance* residency verification rooted in hardware evidence (e.g., TPM/TEE), as well as *Standard Assurance* local co-location proofs provided by conventional workload agents.

This draft acts as the *Conveyance Layer* that integrates with the findings of a *RATS Profile* (such as *Verifiable Geofencing* [[!I-D.lkspa-wimse-verifiable-geo-fence]]) to establish two distinct levels of assurance:

* *Co-location Verification*: A logical binding that ensures the workload and its agent are currently co-located on the same host, typically enforced via operating system isolation and local communication channels (e.g., Unix Domain Sockets).
* *Residency Verification (High Assurance)*: A high-assurance binding where the Workload Identity Agent itself is proven to be integral and rooted in hardware. This ensures the identity is functionally "sticky" to the verified residence via a *Fast Path* renewal which uses cached attestation results.

A workload obtains a fresh signature or proof from a local Workload Identity Agent, typically utilizing a *Workload Fusion Nonce (N_fusion)* to prevent replay. This ensures the identity—and the usage of its associated credentials—is sensitive to the physical or logical residence of the workload, complementing the "East-West" delegation models. Re-attestation follows a *Tiered Schedule* (see [[!I-D.lkspa-wimse-verifiable-geo-fence]]), separating frequent identity renewal from heavyweight platform evidence collection.

2. 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]] [[RFC8174]] when, and only when, they appear in all capitals, as shown here.

This document leverages the terminology defined in the RATS Architecture [[RFC9334]] and the WIMSE Architecture [[!I-D.ietf-wimse-arch]].

Workload Identity Agent (Workload Identity Agent): A local entity that acts as an *Attester* or *Attestation Intermediate* in the RATS framework. It is responsible for providing Evidence or Attestation Results to a workload.

Proof of Residency (PoR) / Co-location: A cryptographic proof that binds a workload’s current execution session to a specific, verified local environment or host.

N_fusion (Workload Fusion Nonce): A nonce provided by the Relying Party specifically for the workload-to-agent fusion flow. It ensures the freshness of the residency proof and cryptographically binds the workload’s request to the local agent’s attestation.

Workload identities are often represented by bearer tokens or keys that, once compromised, can be used by an attacker from any environment. This "portability" allows an attacker who achieves RCE on a workload (e.g., in Region A) or hijacks an AI agent’s logic via *prompt injection* to use the stolen keys or intercepted tokens from an attacking machine (e.g., in Region B).

In the context of *Sovereign Workloads*, this portability is more than a technical vulnerability; it is a *Sovereignty Violation*. If a workload is legally or logically required to operate within a specific jurisdiction, any identity that can be successfully verified from outside that jurisdiction represents a failure of the sovereign boundary. Relying Parties currently lack a standardized way to ensure that a presented identity is being used from the same verified host that was originally authorized.

3. Scope and Boundaries

This document focuses on the *Identity Conveyance* and *Session Binding* mechanisms required for Transitive Attestation. It standardizes how a workload proves its co-location with a local agent and how that proof is bound to application sessions (e.g., mTLS or DPoP).

The following areas are explicitly *OUT OF SCOPE* for this profile:

* *Agent Identity and Integrity*: The mechanism by which a Workload Identity Agent (e.g., SPIRE Agent) proves its own identity and code integrity to a remote verifier.
* *Platform Attestation*: The collection and verification of hardware-level evidence (TPM quotes, TEE measurements) for the host platform.
* *Workload Identity Manager Interactions*: The specific protocols and APIs for obtaining nonces or credentials from a *Workload Identity Manager (WIM Manager)* during the issuance phase (e.g., SVID CSRs).

These security guarantees are provided by the underlying *RATS Profile* for *Verifiable Geofencing* [[!I-D.lkspa-wimse-verifiable-geo-fence]]. This WIMSE profile assumes that the Workload Identity Agent is operating in a verified, integral state as established by the Geofencing layer.

4. The Solution: Transitive Attestation for Proof of Residency

"Transitive Attestation" establishes a chain of trust from a hardware root through a local agent to the workload. The Workload Identity Agent provides the workload with a "live" proof that it is currently resident on the verified host. This local peer-to-peer connection is typically enforced through a *Unix Domain Socket (UDS)*, providing a kernel-level guarantee that the workload is co-located with the hardware-rooted agent.

4.1. mTLS-based Transitive Attestation

In an mTLS environment, the Proof of Residency (PoR) is bound to the mutually authenticated session and the local execution context via a transitive chain of trust.

4.1.1. mTLS PoR Protocol Flow

The mTLS-based flow integrates residency verification into the session establishment and validation phase:

1. *Certificate Extensions*: The client (workload) supplies an X.509 certificate during the mTLS handshake containing a custom extension. This extension includes the public key or SVID details of the local Workload Identity Agent (Attester).
2. *Post-Handshake Nonce*: After the mTLS handshake is successfully completed, the client requests a residency-specific nonce from the *Verifier* (e.g., Relying Party) to ensure anti-replay.

3. *Local Attestation Binding*: The client constructs a PoR assertion payload containing:
   * A cryptographic hash of the TLS Exporter value [[RFC5705]].
   * The residency nonce provided by the server.
   * A timestamp representing the current time of assertion creation.

| [!NOTE] By binding to the TLS Exporter instead of the application traffic keys, the Proof of Residency remains valid across TLS 1.3 KeyUpdate operations. Standard key rotation refreshes traffic keys but does not change the exporter master secret, thus avoiding unnecessary re-attestation cycles while maintaining strong cryptographic binding to the session.

4. *Agent Signature*: The client sends this payload to the local Workload Identity Agent (typically via a Unix Domain Socket). The Workload Identity Agent verifies the local peer environment and signs the payload with its private key.
5. *PoR Submission*: The client sends this attested response to the resource server for verification.
6. *Server Verification*: The *Verifier* performs a joint verification of identity and residency:
   * *Identity*: Verifies the client certificate as part of standard mTLS.
   * *Residency*: Verifies the PoR assertion signature against the Workload Identity Agent public key found in the client's certificate extension.
   * *Binding and Freshness*: Ensures that the TLS Exporter hash, the nonce, and the timestamp match the current active session and are within an acceptable freshness window.

Upon successful verification, the resource server has proof that the client identity (presented via mTLS) is currently resident in the same authorized environment as the verified Workload Identity Agent.

4.2. Demonstrating Proof of Residency (DPoR)

"Demonstrating Proof of Residency" (DPoR) is an enhancement to the Demonstrating Proof-of-Possession (DPoP) mechanism defined in [[RFC9449]]. While DPoP ensures _possession_ of a private key held by the client, DPoR ensures the physical or logical _residency_ of the workload using that key by binding the request to a local attestation.

4.2.1. DPoR Protocol Flow

The DPoR flow integrates residency verification into the per-request application-level authorization:

1. *Nonced Request*: The *Verifier* SHOULD provide a residency-specific nonce (e.g., via a DPoR-Nonce header) to the client to ensure anti-replay of the residency proof.
2. *Local Attestation Binding*: The client constructs a DPoR assertion payload containing:
   * The hash of the DPoP public key used for the request (e.g., the jkt thumbprint).
   * The residency nonce provided by the server.
   * A timestamp representing the current time of assertion creation.
3. *Agent Signature*: The client sends this payload to the local Workload Identity Agent (typically via a Unix Domain Socket). The Workload Identity Agent (acting as an Attester) verifies the local execution context and signs the payload with its private key.
4. *DPoR Assertion Submission*: The client includes the resulting signature in a DPoR header or as an extension to the DPoP JWT.
5. *Server Verification*: The *Verifier* performs a joint verification of possession and residency:
   * *Possession*: Verifies the DPoP proof as per [[RFC9449]].
   * *Residency*: Verifies the DPoR assertion signature against the Workload Identity Agent public key.
   * *Binding and Freshness*: Ensures that the jkt (DPoP key thumbprint), the nonce, and the timestamp in the residency proof match the current request and are within an acceptable freshness window.

This binding ensures that a DPoP key cannot be "exported" and used from a different machine, as the resource server would detect the lack of a valid, hardware-rooted residency proof for that specific key from the new environment.

4.3. Confidential Computing and Hardware-Rooted Binding

In Confidential Computing (CC) environments, the Relationship between high-level workload identities and hardware-rooted evidence is formalized through specific architectural patterns. This section details how Transitive Attestation bridges the gap between hardware quotes and application-layer identities.

4.3.1. Logical Quoting Enclave

In a Transitive Attestation model, trust is passed through a chain of components.

* *Hardware Layer*: A Confidential Computing environment (e.g., Intel TDX or SGX) provides a "Hardware Root of Trust."

* *Intermediate Layer (Workload Identity Agent)*: In this profile, the Workload Identity Agent (e.g., a SPIRE Agent) is treated as the trusted software component that "vouches" for the workloads.
* *Logical Equivalent*: By running the Workload Identity Agent inside a TEE (Trusted Execution Environment), the hardware can attest to the integrity of the agent. This allows the agent to act as a "Compliance Bridge" as defined in [[!I-D.lkspa-wimse-verifiable-geo-fence]]. Just as a hardware *Quoting Enclave (QE)* signs a local report to turn it into a globally verifiable "Quote," the Workload Identity Agent (running in a TEE) signs Workload Identity Documents (SVIDs/WITs). Effectively, the agent becomes a _logical_ quoting enclave that bridges hardware-rooted trust to software-level workload identities.

4.3.2. Channel Binding via TLS Exporter

To prevent man-in-the-middle (MITM) and replay attacks, the hardware evidence must be cryptographically bound to the communication channel. This is achieved using the *TLS Exporter* [[RFC5705]].

* *The Mechanism*: RFC 5705 allows both sides of a TLS session to derive a unique, secret value (the Exporter) that is bound to that specific handshake.
* *The Binding*: The workload places a hash of this TLS Exporter value into the *User Defined Data* field of the CC quote (e.g., ReportData in SGX or RTMR in TDX).
* *Security Property*: This cryptographically binds the hardware evidence to that specific TLS session. If an attacker attempts to reuse the same quote on a different TLS session, the Exporter values will not match, and the verification will fail.

4.3.3. Relying Party Verification Loop

The Relying Party (Verifier) completes the trust loop by performing the following steps:

> [!NOTE] The Relying Party (which may be the Resource Server or an API Gateway/Proxy acting on its behalf) extracts the TLS Exporter value.

1. *Hardware Verification*: Receives the CC Quote and verifies the hardware signature (proving the code is running in a real TEE).
2. *Extractor*: Extracts the TLS Exporter value (or its hash) from the quote’s metadata (User Defined Data).
3. *Comparison*: Compares this value to its own locally computed TLS Exporter value for the current active connection.

4. *Enforcement*: If they match, the Relying Party is certain that the entity it is talking to via TLS is the _exact same entity_ that generated the hardware attestation.

This architecture ensures that identity is not just a bearer token, but is cryptographically tied to the verified runtime environment and the specific communication channel.

5. Relation to Other IETF Work

Layer	Component	WG	Core Responsibility
Layer 1	*Transitive Attestation*	*WIMSE*	*Conveyance*: Binds identity to the local agent (Co-location/ Residency).
Layer 2	*Verifiable Geofencing*	*RATS*	*Platform Evidence*: Verifies host integrity and Workload Identity Agent hardware residency (TPM).
Layer 3	*Verifiable Geofencing*	*RATS*	*Location Evidence*: Verifies physical geography (GNSS/ZKP).
Delegation	*Actor Chain*	*SPICE*	Provides *East-West* identity delegation proof [[!I-D.draft-mw-spice-actor-chain]].
Shield	*SPICE*	*SPICE*	Employs Selective Disclosure (SD-CWT) to protect residency/ geographic privacy.

Table 1

1. *Transitive Attestation (WIMSE) - Layer 1 (Conveyance)*: This document act as the technical integrator profile for WIMSE. It standardizes how local context results are transitively extended to workloads. It reflects its status as the *Consumer* of attestation results to address the *North-South* identity portability problem.

2. *Verifiable Geofencing (RATS) - Layer 2 & 3 (Evidence)*: Defined in [[!I-D.lkspa-wimse-verifiable-geo-fence]]. This document acts as the *RATS Profile* that provides the hardware-rooted foundation (TPM, Silicon Root of Trust, GNSS) and the out-of-band monitoring required to verify the Workload Identity Agent itself. It generates the high-assurance evidence that Layer 1 consumes.
3. *Actor Chain - The Delegation*: Complements this draft by addressing the *East-West* axis of agent-to-agent communication [[!I-D.draft-mw-spice-actor-chain]]. While Transitive Attestation proves _where_ an actor is running (North-South), the Actor Chain proves _who_ called whom across the network (East-West).
4. *SPICE (Secure Patterns for Internet Credential Exchange) - The Shield*: Utilizes Selective Disclosure (SD-CWT) to protect sensitive location data. It allows a workload to prove residency within a broad "Sovereign Zone" without revealing precise GPS coordinates, balancing security with privacy.

Additional relationships include:

* *Verifiable Geofencing [[!I-D.lkspa-wimse-verifiable-geo-fence]]*: Provides the normative technical specification for Layer 2 and Layer 3 attestation. This draft (Transitive Attestation) acts as the application-layer profile that implements these residency proofs within mTLS and DPoP flows, fulfilling the "Silicon-to-SVID" chain.
* *WIMSE Architecture [[!I-D.ietf-wimse-arch]]*: This draft provides the technical fulfillment for the secure agent requirements and thread model mitigations (e.g., token theft) defined in the architecture.

6. Security Considerations

Proof of Residency or Co-location specifically mitigates the "Stolen Credential Portability" threat, which encompasses both stolen private keys and stolen bearer/DPoP tokens.

An attacker who steals a private key or intercepts an active token from a workload cannot use those credentials from an external environment. Any attempt to use the stolen credential requires a corresponding PoR assertion that is:

1. *Locally Attested*: Linked to the local Workload Identity Agent’s signing interface, typically protected by OS permissions or hardware roots.
2. *Context-Specific*: Bound to a fresh, server-provided nonce and a current timestamp.

3. *Protected*: Access to the Workload Identity Agent's signing capability is restricted by local Operating System permissions and logical isolation.

Consequently, credentials become functionally "sticky" to the verified residence; an attacker cannot generate a valid proof without achieving a compromise of the identity agent itself. While a software-based agent provides strong logical isolation, a hardware-rooted agent (see [[!I-D.lkspa-wimse-verifiable-geo-fence]]) provides the highest level of protection against agent cloning and export.

The security of this model relies on the *Workload Identity Agent* correctly providing a fresh assertion bound to the current workload session. While this document specifies the conveyance of N_fusion, the underlying platform-level freshness— including the binding to hardware-rooted platform nonces (N_platform)— is managed by the associated *RATS Profile* (see [[!I-D.lkspa-wimse-verifiable-geo-fence]]). If a verifier accepts a residency proof that lacks a fresh agent signature, the "Silicon-to-SVID" chain of trust is compromised.

TBD: Discussion on Workload Identity Agent compromise, nonce entropy requirements, and clock skew for timestamp verification.

7. IANA Considerations

This document has no IANA actions at this time.

Authors' Addresses

Ram Krishnan
JPMorgan Chase & Co
Email: ramkri123@gmail.com

A Prasad
Oracle
Email: a.prasad@oracle.com

Diego R. Lopez
Telefonica
Email: diego.r.lopez@telefonica.com

Srinivasa Addepalli
Aryaka
Email: srinivasa.addepalli@aryaka.com