RFC-5003: Authorization Flow & Verification Architecture

Title: Authorization Flow & Verification Architecture

Status: Draft

Category: Standards Track

Version: 1.0

Date: March 2026

Authors: Swoop Engineering Team

Related Specifications: * UniKey RFC 2001: Trust Packet Format & Canonicalization

- RFC-1300: Device Authority & Operating System Integration Model

Abstract

The UniKey Authorization Flow defines a decentralized, pre-execution verification model for digital actions. Unlike traditional "post-authentication" systems that rely on shared session state or centralized tokens, this architecture separates **Authority Verification** from **Action Execution**. By utilizing the Trust Packet as a transport-agnostic container, this specification enables any participating system—from AI agents to global payment rails—to independently and deterministically validate the intent and authorization of a request before commitment. This document establishes the normative sequence for Trust Packet generation, propagation, and stateless verification across heterogeneous networks.

---

1. Purpose

This specification defines the **authorization flow and verification architecture** used by UniKey to validate digital actions before execution.

The goal of the UniKey authorization flow is to ensure that digital actions carry verifiable proof of authority that can be validated by any participating system without requiring shared state, proprietary integrations, or centralized authorization services.

This document defines:

• the sequence of authorization steps
• participating system roles
• how Trust Packets are generated and propagated
• how verification occurs prior to execution

This specification intentionally does not define business policies, user interfaces, or settlement logic.

---

2. Architectural Principle

Most digital systems today execute actions after verifying credentials such as passwords, tokens, or API keys.

UniKey introduces **pre-execution authorization**, where systems verify device-bound cryptographic authority before the action is executed.

The architecture therefore separates:

• **authority verification**
• **execution and settlement**

Authority is verified first.

Execution follows only after verification succeeds.

---

3. Participating Roles

The UniKey authorization architecture includes the following logical participants.

3.1 Device Authority

The user’s device generates Trust Packets using the device authority model defined in RFC-006.

The device confirms authorization and signs the Trust Packet Hash (TPH).

3.2 Initiating System

The initiating system is the application or service requesting execution of an action.

Examples include:

• merchant applications
• AI agents
• service APIs
• enterprise platforms

3.3 Integration Layer

The integration layer forwards requests between initiating systems and verifiers.

Typical implementations include:

• payment service providers (PSPs)
• API gateways
• authentication proxies
• service middleware

The integration layer does not grant authority. It transports Trust Packets.

3.4 UniKey Verifier

The verifier evaluates Trust Packets according to the rules defined in RFC-001.

The verifier performs deterministic validation of:

• signatures
• action binding
• audience binding
• replay protection
• delegation chains

Verification is stateless and independently reproducible.

3.5 Execution System

The execution system performs the requested action after verification.

Examples include:

• payment networks
• banking systems
• blockchain smart contracts
• enterprise services
• API endpoints

Execution systems rely on verification results but do not generate Trust Packets.

---

4. Authorization Flow

The following sequence illustrates the standard UniKey authorization flow.

Step 1 — Action Initiation

A device or agent requests an action from an initiating system.

Example:

User device → merchant application

The initiating system constructs a request describing the action.

Step 2 — Request Forwarding

The initiating system forwards the request to an integration layer.

Example:

Merchant → PSP or API gateway

At this stage the action is not yet authorized.

Step 3 — Authorization Request

The integration layer requests authorization from the user device.

Example:

PSP → user device

The authorization request includes the canonical representation of the action.

Step 4 — Device Authorization

The device confirms the action according to device policy and produces a Trust Packet.

The device signs the Trust Packet Hash defined in RFC-001.

The Trust Packet contains:

• action binding
• audience binding
• freshness constraints
• device authority signature

Step 5 — Trust Packet Propagation

The Trust Packet is returned to the integration layer and forwarded along the transaction path.

Example:

Device → PSP → verifier

The Trust Packet accompanies the action request.

Step 6 — Verification

The verifier performs deterministic validation of the Trust Packet.

The verifier must validate:

• signature authenticity
• action binding
• audience match
• temporal validity
• replay protection
• delegation chain validity

If verification fails, the action must be rejected.

Step 7 — Execution

If verification succeeds, the action may be executed by the execution system.

Examples include:

• payment authorization
• API execution
• smart contract invocation

---

5. Authorization Chain

The UniKey architecture produces a **chain of authorization events**.

Each participant verifies the previous step and forwards the authorization context.

Example chain:

Device → Initiating System → Integration Layer → Execution System

Each step can independently verify the Trust Packet.

This creates a verifiable chain of authority without requiring a global ledger.

---

6. Trust Packet Propagation

Trust Packets may travel through multiple systems before reaching the execution layer.

Systems forwarding Trust Packets must not modify any fields covered by the Trust Packet Hash.

Forwarding systems may add transport metadata outside the packet structure.

Any modification to the Trust Packet invalidates the signatures.

---

7. Verification Responsibilities

Verifiers must perform the validation steps defined in RFC-001 before allowing execution.

Verifiers must enforce:

• audience binding
• action binding
• replay prevention
• delegation chain integrity

Verification must fail closed.

If validation is incomplete or ambiguous, the action must be rejected.

---

8. Stateless Verification

UniKey verification is designed to operate without shared session state.

Verifiers may maintain a replay detection store for nonce or packet identifiers but do not require centralized coordination.

Any conforming verifier can independently validate a Trust Packet.

---

9. Transport Independence

The authorization flow does not depend on a specific transport protocol.

Trust Packets may be conveyed using:

• HTTPS
• SMTP
• message queues
• event streams
• QR or NFC handoff

Security decisions must be based only on the canonical Trust Packet and verified signatures.

Transport metadata must not be trusted for authorization decisions.

---

10. Security Properties

The authorization architecture provides the following properties.

Pre-Execution Authorization
Actions are authorized before execution occurs.

Device Authority
Authorization requires possession of the device signing key.

Action Binding
Authorization is cryptographically bound to a specific action.

Audience Binding
Authorization is valid only for the intended verifier.

Replay Resistance
Authorization artifacts cannot be reused.

Delegation Containment
Delegated authority cannot expand scope.

---

11. Residual Risks

UniKey does not eliminate all attack scenarios.

Residual risks include:

• device compromise
• user deception
• domain key compromise
• malicious delegated agents

The architecture shifts attacks from scalable remote credential abuse to attacks requiring device compromise or user participation.

---

12. Conformance

An implementation is conformant with this specification if it:

• follows the authorization flow defined above
• generates Trust Packets as defined in RFC-001
• uses device authority defined in RFC-006
• verifies Trust Packets before executing actions

Non-conformant flows must not be treated as valid UniKey authorization.