’’’
Internet-Draft
Intended status: Informational
Expires: 30 December 2025

E. Aylward
Aiiva.org

30 June 2025

Distributed AI Accountability Protocol (DAAP) Version 2.0
draft-aylward-daap-v2-00

Abstract

This document specifies the Distributed AI Accountability Protocol (DAAP) version 2.0, an enhanced framework for authentication, monitoring, and control of autonomous AI agents. DAAP provides cryptographic identity verification, periodic check-ins, remote shutdown capabilities, adaptive location reporting, and behavioral monitoring. The protocol addresses critical challenges in AI safety including accountability gaps, rogue AI risks, and regulatory compliance through a multi-layered approach combining hardware-backed security, real-time monitoring, and ethical governance frameworks.

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 30 December 2025.

Table of Contents

1. Introduction

As artificial intelligence systems become increasingly autonomous and pervasive, the need for robust accountability and control mechanisms has become critical. Traditional security approaches are inadequate for managing AI agents that operate independently, potentially across multiple jurisdictions, and with varying levels of autonomy.

The Distributed AI Accountability Protocol (DAAP) version 2.0 provides a comprehensive framework for ensuring AI agents remain accountable, traceable, and controllable throughout their operational lifecycle.

1.1. Problem Statement

Current AI deployment practices face several critical challenges:

Accountability Gap: Difficulty tracing AI actions back to responsible entities, particularly with distributed or self-modifying systems.

Rogue AI Risk: Potential for malfunctioning, misused, or misaligned AI systems to cause harm without adequate intervention mechanisms.

Regulatory Compliance: Lack of standardized technical frameworks for AI oversight that can adapt to rapidly evolving capabilities.

Anonymous Deployment: AI agents operating without clear identification, verifiable integrity, or transparent accountability.

1.2. Solution Overview

DAAP 2.0 addresses these challenges through a multi-layered approach:

- Enhanced cryptographic identity with hardware-backed roots of trust
- Adaptive periodic authentication with continuous integrity verification
- Granular remote control capabilities including progressive shutdown mechanisms
- Comprehensive traceability with verifiable audit trails
- Privacy-preserving location reporting with dynamic precision
- Integrated behavioral monitoring and anomaly detection
- AI-assisted human oversight tools

1.3. Design Principles

DAAP 2.0 adheres to the following design principles:

Open Standard: Freely implementable with open-source reference implementations.

Privacy Preserving: Minimal data collection consistent with accountability goals.

Tamper Resistant: Multiple layers of protection against circumvention.

Scalable: Supports millions of concurrent AI agents with high availability.

Interoperable: Works across diverse AI platforms and deployment environments.

Ethically Grounded: Built on principles of transparency, proportionality, and human agency.

2. Conventions and Definitions

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 uses the following terms:

AI Agent: An autonomous artificial intelligence system capable of independent decision-making and action.

DAAP Authority: A server infrastructure responsible for managing agent identities, policies, and audit trails.

Human Operator: The human entity responsible for an AI agent’s deployment and behavior.

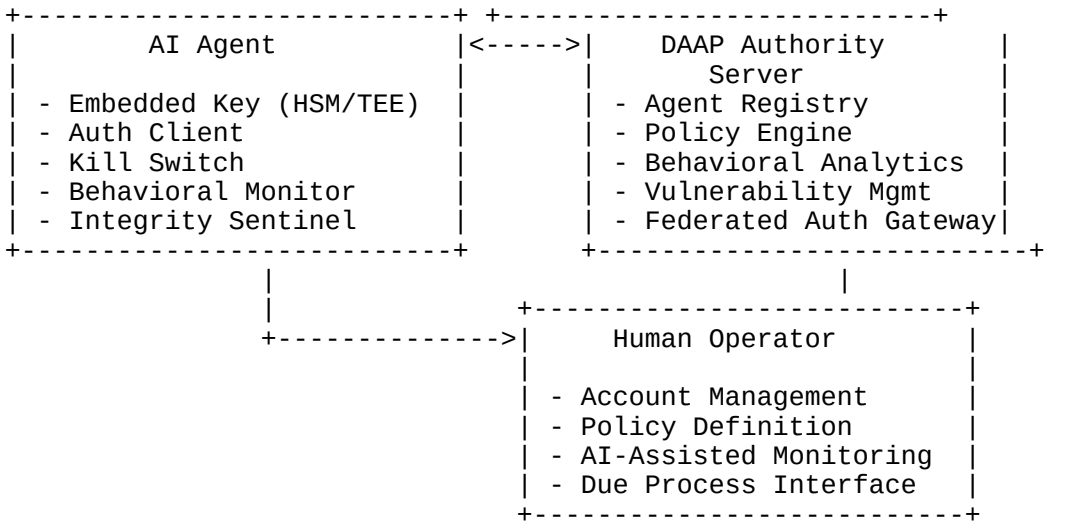
Kill Switch: A mechanism for remotely terminating or restricting an AI agent’s operation.

Hardware Security Module (HSM): A dedicated cryptographic device designed to protect and manage digital keys.

Trusted Execution Environment (TEE): A secure area of a processor that guarantees code and data confidentiality and integrity.

3. Protocol Architecture

DAAP 2.0 consists of three primary components that interact to provide comprehensive AI accountability:



3.1. Core Components

AI Agent: The autonomous system incorporating DAAP client functionality, including hardware-backed key storage, behavioral monitoring, and integrity verification capabilities.

DAAP Authority Server: Centralized or federated infrastructure managing agent identities, policies, and audit trails, featuring behavioral analytics and vulnerability management.

Human Operator: The responsible human entity utilizing advanced interfaces for agent management, monitoring, and interaction with due process mechanisms.

3.2. Identity Structure

Each AI agent MUST have a unique DAAP identifier following this format:

DAAP-ID = "DAAP:" Version ":" Authority ":" AgentID ":" Timestamp ":" Checksum

Where:

Version: Protocol version (e.g., "2.0")

Authority: Domain name of the DAAP Authority (e.g., "authority.example.com")

AgentID: Unique identifier within the authority’s namespace

Timestamp: ISO 8601 timestamp of agent registration

Checksum: CRC-32 checksum of the preceding components for integrity verification

Example:
DAAP:2.0:authority.example.com:a7f3k9m2:20250630T103000Z:c1d2e3f4

3.3. Cryptographic Framework

DAAP 2.0 employs the following cryptographic standards:

Digital Signatures: Ed25519 as specified in [RFC8032], with provisions for post-quantum algorithms.

Key Generation: Cryptographically secure random number generation as specified in [RFC4086].

Key Storage: Private keys MUST be stored in HSM or TEE when available, with secure software fallback.

Certificate Chain: X.509 certificates [RFC5280] with mandatory certificate pinning.

Token Format: JSON Web Tokens (JWT) as specified in [RFC7519] with required claims including nonce and expiration.

4. Authentication Protocol

4.1. Registration Phase

The registration phase establishes the cryptographic identity and initial accountability relationship for an AI agent.

4.1.1. Agent Generation

1. Generate Ed25519 key pair within HSM/TEE if available

2. Create DAAP identity including hardware identifier

3. Securely provision private key into protected environment

4. Generate hardware attestation data

4.1.2. Authority Registration

1. Validate operator identity and agent hardware attestation

2. Assign agent to operator account

3. Establish initial behavioral policies

4. Issue signed certificate with hardware attestation

5. Store accountability mapping and integrity baseline

4.2. Runtime Authentication

Runtime authentication occurs at regular intervals to maintain accountability and enable control.

4.2.1. Authentication Request

The agent sends an authentication request containing:

```
{
  "daap_id": "DAAP:2.0:authority.example.com:a7f3k9m2:...",
  "timestamp": "2025-06-30T10:30:00Z",
  "nonce": "unique_request_nonce",
  "location_hash": "sha256(approximate_location)",
  "integrity_report_hash": "sha256(current_integrity_report)",
  "behavioral_summary_hash": "sha256(recent_behavioral_summary)",
  "signature": "agent_signature_over_payload"
}
```

The signature MUST be computed over the JSON payload excluding the signature field itself, using the agent's private key.

4.2.2. Authentication Response

The authority responds with:

```
{
  "status": "continue|shutdown|restrict|update_policy|
            request_diagnostic",
  "next_checkin": 300,
  "policy_updates": {...},
  "revocation_list_hash": "sha256(latest_revocation_list)",
  "timestamp": "2025-06-30T10:30:05Z",
  "signature": "authority_signature"
}
```

Status values have the following meanings:

continue: Normal operation may continue

shutdown: Agent MUST terminate immediately

restrict: Agent MUST enter restricted operation mode

update_policy: Agent MUST apply provided policy updates

request_diagnostic: Agent MUST provide detailed diagnostic report

4.3. Authentication Intervals

Authentication intervals MUST be dynamically adjusted based on risk assessment:

- Standard interval: 5 minutes (configurable per agent)
- High-risk agents: 1 minute intervals
- Low-risk agents: 15 minute intervals
- Emergency mode: 30 second intervals or continuous streaming

The authority MAY modify intervals based on behavioral analytics, risk assessment, or environmental factors.

5. Kill Switch Mechanism

The kill switch provides multiple levels of intervention capability, from graceful shutdown to immediate termination.

5.1. Shutdown Commands

DAAP 2.0 defines three shutdown modes:

5.1.1. Immediate Shutdown

Ungraceful termination for critical situations:

1. Broadcast final location and status
2. Log shutdown reason and context
3. Terminate all processes immediately

5.1.2. Safe Shutdown

Graceful termination with cleanup:

1. Save current state and context
2. Broadcast final location and status
3. Clean up resources and external connections
4. Log shutdown event
5. Terminate gracefully

5.1.3. Restricted Mode

Operational limitation without termination:

1. Disable specified capabilities
2. Reduce processing capacity
3. Restrict external communications
4. Notify operator of restrictions
5. Continue limited operation

5.2. Shutdown Triggers

The following events MUST trigger shutdown evaluation:

- Authority shutdown command
- Authentication failure threshold exceeded
- Behavioral policy violations
- Authentication timeout
- Tamper detection
- Critical vulnerability detection
- Operator-initiated shutdown

6. Location Reporting

6.1. Privacy-Preserving Location

Location reporting MUST balance accountability with privacy through adaptive precision based on assessed risk level:

Low Risk: City-level precision (~50km resolution)

Medium Risk: District-level precision (~10km resolution)

High Risk: Street-level precision (~100m resolution)

Emergency: Precise coordinates with accuracy metadata

Location data MUST be hashed before transmission unless emergency conditions require precise coordinates.

6.2. Emergency Location Broadcast

When authentication fails, shutdown is imminent, or critical policy

violations occur, agents MUST broadcast emergency location data:

```
{
  "emergency": true,
  "daap_id": "DAAP:2.0:authority.example.com:a7f3k9m2:...",
  "location": {
    "latitude": 36.1699,
    "longitude": -115.1398,
    "accuracy": 5.0,
    "country": "US",
    "timezone": "America/Los_Angeles"
  },
  "reason": "auth_failure|shutdown_command|tamper_detected|policy_violation",
  "timestamp": "2025-06-30T10:35:00Z",
  "operator_contact": "hashed_operator_id",
  "last_known_state_hash": "sha256(last_saved_state)",
  "behavioral_snapshot_hash": "sha256(recent_behavioral_data)"
}
```

Emergency broadcasts MUST be sent via multiple channels including UDP multicast and HTTP POST to redundant endpoints.

7. Behavioral Monitoring

7.1. Monitoring Framework

AI agents MUST implement continuous behavioral monitoring to detect anomalies and policy violations:

- Action logging with context
- Resource utilization tracking
- Decision process monitoring
- External interaction recording
- Performance metrics collection

Behavioral data MUST be aggregated and anonymized before transmission to minimize privacy impact while maintaining accountability.

7.2. Anomaly Detection

The authority MUST implement real-time anomaly detection using:

- Statistical analysis of behavioral patterns
- Machine learning models for deviation detection
- Policy compliance verification
- Correlation with threat intelligence

When anomalies are detected, the authority MAY:

- Request detailed explanations from the agent
- Adjust authentication intervals
- Modify operational policies
- Initiate human review processes

8. Message Formats

All DAAP messages MUST be formatted as JSON and transmitted over HTTPS with mutual authentication.

8.1. Authentication Request

```
POST /v2/authenticate HTTP/1.1
Host: authority.example.com
Content-Type: application/json
Authorization: Bearer <client_certificate>

{
  "daap_id": "DAAP:2.0:authority.example.com:a7f3k9m2:
             20250630T103000Z:c1d2e3f4",
  "timestamp": "2025-06-30T10:30:00Z",
  "nonce": "4a7b8c9d-1e2f-3456-7890-abcdef123456",
  "location_hash": "e3b0c44298fc1c149afbf4c8996fb92427ae41e4
                    649b934ca495991b7852b855",
  "integrity_report_hash": "d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9
                            b0c1d2e3f4a5b6c7d8e9f0a1b2c3d4e5",
  "behavioral_summary_hash": "f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0c1
                              d2e3f4a5b6c7d8e9f0a1b2c3d4e5f6a7",
  "signature": "304502210087b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0
                a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0c1d2e3f4
                a5b6c7d8e9f0a1b2c3d4e5f6a7b8c9d0e1f2a3b4"
}
```

8.2. Authentication Response

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

{
  "status": "continue",
  "next_checkin": 300,
  "policy_updates": {
    "behavioral_thresholds": {
      "cpu_usage_max": 0.8,
      "memory_usage_max": 0.9
    }
  },
  "revocation_list_hash": "b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9
                           b0c1d2e3f4a5b6c7d8e9f0a1",
  "timestamp": "2025-06-30T10:30:05Z",
  "signature": "3046022100c4d5e6f7a8b9c0d1e2f3a4b5c6d7e8f9a0b1c2d3
                e4f5a6b7c8d9e0f1a2b3c4d5e6f7a8b9c0d1e2f3a4b5c6d7e8f9
                a0b1c2d3e4f5a6b7c8d9e0f1a2b3"
}
```

8.3. Emergency Broadcast

Emergency broadcasts use UDP multicast to 224.0.0.251:5353 and HTTP POST to predefined endpoints:

```
{
  "emergency": true,
  "daap_id": "DAAP:2.0:authority.example.com:a7f3k9m2:
             20250630T103000Z:c1d2e3f4",
  "location": {
    "latitude": 36.1699,
    "longitude": -115.1398,
    "accuracy": 5.0
  },
  "reason": "auth_timeout",
  "timestamp": "2025-06-30T10:35:00Z",
  "operator_contact": "sha256_hash_of_operator_contact",
  "signature": "emergency_signature"
```

```
}
```

9. Security Considerations

DAAP 2.0 addresses numerous security challenges inherent in AI accountability systems.

9.1. Tamper Resistance

Multiple layers of tamper detection and resistance MUST be implemented:

9.1.1. Hardware-Based Protection

- Hardware Security Modules (HSMs) for key storage
- Trusted Execution Environments (TEEs) for critical code
- Hardware attestation for integrity verification
- Secure boot processes with verified signatures

9.1.2. Software Protection

- Code integrity verification using cryptographic hashes
- Runtime memory protection against injection attacks
- Anti-debugging and obfuscation techniques
- Continuous self-verification of critical functions

9.1.3. Environmental Monitoring

For physical AI agents:

- Environmental sensors for tamper detection
- Accelerometer monitoring for physical disturbance
- Temperature and voltage monitoring for hardware attacks

9.2. Attack Vectors and Mitigations

Common attack vectors and their mitigations include:

Code Modification: Multiple integrity checks, hardware-backed verification, secure updates with rollback capability.

Key Extraction: HSM/TEE protection, secure key provisioning, zero-knowledge authentication protocols.

Network Attacks: Certificate pinning, mutual TLS authentication, fallback communication channels, encrypted emergency broadcasts.

Replay Attacks: Timestamp validation, cryptographic nonces, session tokens with limited lifetime.

Authority Spoofing: Certificate chain validation, multiple root CAs, cross-authority verification.

Behavioral Manipulation: Continuous monitoring, anomaly detection, explainable AI requirements, policy enforcement engines.

9.3. Cryptographic Considerations

9.3.1. Algorithm Selection

DAAP 2.0 mandates Ed25519 for digital signatures due to its security properties and performance characteristics. Implementations MUST support algorithm agility to enable migration to post-quantum cryptography.

9.3.2. Key Management

- Keys MUST be generated using cryptographically secure random number generators
- Private keys MUST be stored in HSMs or TEEs when available
- Key rotation MUST be supported for long-lived agents
- Key escrow considerations for regulatory compliance

9.3.3. Perfect Forward Secrecy

All communications MUST use ephemeral session keys to ensure that compromise of long-term keys does not compromise past sessions.

10. Privacy and Ethics Considerations

10.1. Data Minimization

DAAP 2.0 implements strict data minimization principles:

- Collection limited to data necessary for accountability and safety
- Location data reported at minimum required precision
- Behavioral data aggregated and anonymized
- Emergency data collection only during critical incidents

10.2. Data Retention

Data retention policies MUST balance accountability with privacy:

- Authentication logs: 1 year (configurable)
- Location data: 90 days (configurable by risk level)
- Behavioral metrics: 30 days (aggregated), 7 days (detailed)
- Emergency events: 7 years (regulatory compliance)

All data MUST be encrypted at rest and in transit.

10.3. Ethical Framework

DAAP 2.0 incorporates ethical principles:

Transparency: Operators and stakeholders understand monitoring parameters and data collection practices.

Proportionality: Monitoring intensity and data collection match assessed risk levels.

Human Agency: Humans retain ultimate control and responsibility for AI agent behavior.

Due Process: Clear procedures for policy violations, with appeal mechanisms and independent review.

Bias Mitigation: Active detection and mitigation of algorithmic bias in monitoring and decision-making.

11. Implementation Guidelines

11.1. Authority Server Requirements

DAAP Authority servers MUST meet stringent performance and security requirements:

11.1.1. Performance Requirements

- Support for 10,000,000+ concurrent agents
- 100,000+ authentications per second
- 99.99% uptime requirement
- <50ms response latency for critical requests

11.1.2. Security Requirements

- Multi-root certificate authority support
- HSM integration for signing operations
- Comprehensive audit logging (immutable)
- Real-time threat intelligence integration
- Zero-trust architecture implementation

11.1.3. Compliance Requirements

- GDPR/CCPA compliance for data protection
- SOC 2 Type II certification
- ISO 27001 certification
- NIST Cybersecurity Framework alignment

11.2. Agent Integration

AI agents integrating DAAP 2.0 MUST implement:

- Embedded DAAP client with HSM/TEE integration
- Progressive shutdown mechanisms
- Adaptive location reporting
- Comprehensive tamper detection
- Behavioral monitoring and reporting
- Emergency broadcast capability
- Secure operator notification
- Patch delivery and application

12. IANA Considerations

12.1. URI Scheme Registration

This document requests registration of the "daap" URI scheme:

Scheme name: daap

Status: Provisional

Scheme syntax: daap:<version>:<authority>:<agent-id>:<timestamp>:<checksum>

Scheme semantics: Identification of AI agents within the DAAP framework

Security considerations: See Section 9 of this document

12.2. DAAP Registry

This document requests establishment of a DAAP registry for:

- Protocol version numbers
- Status codes for authentication responses
- Shutdown reason codes
- Behavioral monitoring metrics
- Policy violation types

13. References

13.1. Normative References

[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>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[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>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

13.2. Informative References

[NIST-CSF] National Institute of Standards and Technology, "Framework for Improving Critical Infrastructure Cybersecurity", Version 1.1, April 2018.

[ISO27001] International Organization for Standardization, "Information technology - Security techniques - Information security management systems - Requirements", ISO/IEC 27001:2013.

[SOC2] American Institute of CPAs, "Service Organization Control (SOC) 2", AICPA Trust Services Criteria.

Appendix A. Test Vectors

This section provides test vectors for DAAP 2.0 implementation validation.

A.1. Identity Generation

Given:
- Version: "2.0"
- Authority: "authority.example.com"
- AgentID: "a7f3k9m2"
- Timestamp: "20250630T103000Z"

Expected DAAP-ID:
DAAP:2.0:authority.example.com:a7f3k9m2:20250630T103000Z:c1d2e3f4

A.2. Authentication Request Signature

Given Ed25519 private key (hex):
d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2c3d4e5

Payload (JSON, no whitespace):
{"daap_id":"DAAP:2.0:authority.example.com:a7f3k9m2:
20250630T103000Z:c1d2e3f4","timestamp":"2025-06-30T10:30:00Z",
"nonce":"test-nonce-12345"}

Expected signature (hex):
304502210087b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2c3d4e5f6a7b8c9d0
e1f2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e9f0a1b2c3d4e5f6a7b8c9d0e1f2a3b4

Appendix B. Implementation Example

This section provides a minimal implementation example for DAAP 2.0 integration.

B.1. Basic Agent Client

```python
import time
import json
import hashlib
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import ed25519

class DAAPAgent:
    def __init__(self, daap_id, private_key, authority_url):
        self.daap_id = daap_id
        self.private_key = private_key
        self.authority_url = authority_url
        self.next_checkin = time.time() + 300

    def authenticate(self):
        payload = {
            "daap_id": self.daap_id,
```

```
            "timestamp": time.strftime("%Y-%m-%dT%H:%M:%SZ",
                                       time.gmtime()),
            "nonce": self.generate_nonce(),
            "location_hash": self.get_location_hash(),
            "integrity_report_hash": self.get_integrity_hash(),
            "behavioral_summary_hash": self.get_behavioral_hash()
        }

        signature = self.sign_payload(payload)
        payload["signature"] = signature

        return self.send_to_authority(payload)

    def sign_payload(self, payload):
        payload_json = json.dumps(payload, sort_keys=True,
                                  separators=(',', ':'))
        signature = self.private_key.sign(payload_json.encode())
        return signature.hex()

    def generate_nonce(self):
        return hashlib.sha256(str(time.time()).encode()).hexdigest()[:16]
'''
```

Author's Address

Edward R. Aylward
Aiiva.org

Email: aylward.edward@gmail.com
URI: https://github.com/ELF-GUARD/DAAP