Stream: Independent Submission
RFC: 9932
Category: Informational
Published: April 2026
ISSN: 2070-1721
Authors: S. Halén, *The Swedish Internet Foundation*; J. Schlyter, *Kirei AB*

RFC 9932
Mutually Authenticating TLS in the Context of Federations

Abstract

This Informational Independent Submission to the RFC Series describes a means to use TLS 1.3 to perform machine-to-machine mutual authentication within federations. This memo is not a standard. It does not modify the TLS protocol in any way, nor does it require changes to common TLS libraries. TLS is specified and standardized by the IETF's TLS Working Group.

The framework enables interoperable trust management for federated machine-to-machine communication. It introduces a centrally managed trust anchor and a controlled metadata publication process, ensuring that only authorized members are identifiable within the federation. These mechanisms support unambiguous entity identification and reduce the risk of impersonation, promoting secure and policy-aligned interaction across organizational boundaries.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This is a contribution to the RFC Series, independently of any other RFC stream. The RFC Editor has chosen to publish this document at its discretion and makes no statement about its value for implementation or deployment. Documents approved for publication by the RFC Editor are not candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9932.

Copyright Notice

Table of Contents

1. Introduction

This document describes the Mutually Authenticating TLS in Federations (MATF) framework, developed to complement multilateral Security Assertion Markup Language (SAML) federations within the education sector. These federations often rely on just-in-time provisioning, where user accounts are created at first login based on information from the SAML assertion. However, educators need to be able to manage resources and classes before students access the service. MATF bridges this gap by using secure machine-to-machine communication, enabling pre-provisioning of user information with a trust model and metadata structure inspired by SAML federations.

MATF is designed specifically for secure authentication in machine- to-machine contexts, such as RESTful APIs (where "RESTful" refers to the Representational State Transfer (REST) architecture) and service-to-service interactions, and is not intended for browser-based authentication. Because its applicability in a browser environment has not been studied, using MATF within browsers is not recommended. Doing so may introduce risks that differ from those typically addressed by standard browser security models.

This work is not a product of the IETF, does not represent a standard, and has not achieved community consensus. It aims to address specific federation challenges and provide a framework for secure communication.

TLS is specified by the IETF TLS Working Group. TLS 1.3 is defined in [RFC8446]. Additional information about the TLS Working Group is available at <https://datatracker.ietf.org/wg/tls/about/>.

1.1. Reserved Words

This document is an Informational RFC, which means it offers information and guidance but does not specify mandatory standards. Therefore, the keywords used throughout this document are for informational purposes only and do not imply any specific requirements.

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.

1.2. Terminology

Federation: A trusted network of entities that adhere to common security policies and standards, using MATF for secure communication.

Federation Member: An entity that has been approved to join the federation and can leverage MATF for secure communication with other members.

Federation Operator: The entity responsible for the overall operation and management of the federation, including managing the federation metadata, enforcing security policies, and onboarding new members.

Federation Metadata: A cryptographically signed document containing information about all entities within the federation.

Metadata Repository: A centralized repository storing information about all entities within the federation.

Member Metadata: Information about entities associated with a specific member within the federation.

Member Vetting: The process of verifying and approving applicants to join the federation, ensuring they meet security and trustworthiness requirements.

Trust Anchor: The federation's root of trust is established by the public key used to verify federation metadata signatures, which allows participants to confidently rely on the information it contains.

2. Diverse Design Patterns

MATF is designed to be flexible and adaptable to the varying needs of different federations. Federations can differ significantly in terms of size, scope, and security requirements, which makes it challenging to prescribe a one-size-fits-all trust framework and security measures.

For instance, in the European Union, Regulation (EU) No 910/2014 (the electronic identification, authentication, and trust services (eIDAS) Regulation) [eIDAS] establishes a regulatory framework for electronic identification and trust services for electronic transactions in the internal market. The eIDAS Regulation provides a basis for cross-border recognition of notified electronic identification schemes and for regulated trust services.

Similarly, national federations, such as those found in education or healthcare sectors, often have their own specific trust frameworks and security measures tailored to their unique needs. These federations may leverage existing national identification systems or other trusted credentials to establish member identities and ensure secure interactions.

Organizations may also set up their own federations, tailored to the specific security requirements and trust models relevant to their context. For example, a private business federation might establish its own vetting processes and trust framework based on the nature of its business and the sensitivity of the data being exchanged.

By allowing federations the flexibility to tailor their trust frameworks and security measures, MATF can support a wide range of use cases. This flexibility is crucial for accommodating the diverse requirements and challenges faced by different federations, ensuring a secure and adaptable system for establishing trust and facilitating secure communication.

3. Trust Model

The MATF framework operates on a trust model that is central to its design and functionality. This section outlines the key components of this trust model and its implications for federation members and the federation operator.

3.1. Role of the Federation Operator

The federation operator plays a critical role in the MATF framework. This entity is responsible for:

- Managing the central trust anchor, which is used to establish trust across different domains within the federation.
- Vetting federation members to ensure they meet the required standards and policies.
- Maintaining and securing the federation metadata, which includes public key pins [RFC7469], issuer certificates, and other essential information.

Additionally, the federation operator **SHOULD** develop their own threat models to proactively identify potential risks and threats. This process involves examining the operating environment, evaluating both internal and external threats, and understanding how vulnerabilities can be exploited. The goal of the threat model is to enable the federation operator to establish mitigation strategies that address the identified risks.

The security and stability of the federation rely on the integrity and competence of the federation operator. Members must be able to fully trust this central authority, as its role is essential to maintaining the federation's reliability and security.

3.2. Federation Members' Responsibilities

Federation members share the responsibility of maintaining trust and security within the federation.

Their responsibilities include:

- Adhering to the federation's security policies and procedures.
- Ensuring the accuracy and timeliness of their metadata submissions.
- Cooperating with the federation operator's vetting and security measures.

By fulfilling these responsibilities, federation members help sustain the trust framework that enables secure and reliable communication within the federation.

Federation members submit member metadata to the federation. As part of federation operations, the federation **MUST** ensure the authenticity and integrity of submitted member metadata and the authenticity of the submitting member.

3.3. Chain of Trust

Each federation operates within a trust framework that encompasses its own security policies and procedures. This framework is designed to ensure the integrity, authenticity, and confidentiality of communications within the federation. Key components of this framework include:

- Public key pinning [RFC7469] and preloading to thwart on-path attacks by rejecting peers whose public key in the presented certificate does not match a pin published in the federation metadata.
- Regular updates and verification of federation metadata to prevent the use of outdated or compromised information.

The federation operator aggregates, signs, and publishes the federation metadata, which combines all members' member metadata along with additional federation-specific information. By placing trust in the federation and its associated federation metadata signature verification key, federation members trust the information contained within the federation metadata.

The trust anchor for the federation is established through the federation metadata signature verification key, a critical component requiring secure distribution and verification. To achieve this, the signature verification key material is distributed using a JSON Web Key (JWK) Set [RFC7517], providing a flexible framework for exposing multiple public keys, including the current signature verification key and keys for rollover. This structured approach ensures members can readily access the necessary keys for verification purposes.

An additional layer of security is introduced through thumbprint verification [RFC7638], where federation members can independently verify the key's authenticity. This involves comparing the calculated cryptographic thumbprint of the key with a trusted value, ensuring its integrity. Importantly, this verification process can be conducted through channels separate from the JWK Set itself, enhancing security by eliminating reliance on a single distribution mechanism.

This trust framework is essential for enabling seamless and secure interoperability across different trust domains within the federation.

3.4. Member Vetting

To ensure the security and integrity of the MATF framework, a member vetting process is essential. Detailed vetting processes are beyond the scope of this document but can be guided by established frameworks such as eIDAS and eduGAIN.

The following are non-normative references to established frameworks:

- eIDAS: The eIDAS regulation can provide guidance for member vetting and identity assurance practices.

- eduGAIN: eduGAIN is an interfederation service connecting identity federations worldwide, primarily within the research and education sector. eduGAIN documentation on participation requirements and federation practices can inform member vetting processes [eduGAIN].

3.5. Metadata Authenticity

Ensuring the authenticity of metadata is necessary for maintaining the security and trustworthiness of the MATF framework. This document specifies mechanisms for protecting and verifying the authenticity of federation metadata, including JWS signing. Operational procedures for authenticating member metadata submissions are outside the scope of this document and are defined by the federation operator or applicable regulatory bodies.

4. Metadata Repository

The MATF metadata repository acts as a central vault, securely storing all information about all participating federation members and their respective entities. This information, known as federation metadata, is presented as a JSON Web Signature (JWS) [RFC7515] to ensure its authenticity and integrity.

The metadata repository is subject to stringent security measures to safeguard the integrity of the stored information. This **MAY** involve:

- Member management: The federation operator can centrally enforce security policies and vet new members before they are added to the repository.
- Access controls: Access to repository management functions and member metadata submission endpoints **SHOULD** be restricted to authorized federation members.
- Regular backups: Robust backup procedures ensure data recovery in case of unforeseen circumstances.

Before member metadata is added to the federation's repository, the submitted metadata **MUST** undergo a validation process. This process verifies the accuracy, completeness, and validity of the information provided by a member. Metadata that does not pass validation **MUST** be rejected. The validation process MUST include, at a minimum, the following checks:

- Format validation: The submitted metadata is checked to ensure that it conforms to the schema and format specifications defined in Section 6.2 and Appendix A.
- Unique entity identifier: The submitted metadata is checked to ensure that the `entity_id` value, as defined in Section 6.1.1, is not already registered by another member.
- Unique public key pin digests: The submitted metadata is checked to ensure that `pins` entries, as defined in Section 6.1.1.1, do not introduce a `digest` value that is already registered to a different `entity_id`. While reuse of the same `digest` value within the same `entity_id` is permitted, uniqueness across different entities is **REQUIRED** to prevent identity collisions and to support the resolution of a unique `entity_id` from a derived pin, as specified in Section 5.2.

- Issuer certificate checks: The issuer certificates in `issuers`, as defined in Section 6.1.1, are checked to ensure that they are syntactically valid, not expired, and use algorithms that meet the federation's security requirements.
- Tag validation: Tags, as defined in Section 6.1.1.1, are checked to ensure that they conform to the defined tag syntax. If the federation defines an approved set of tag values, submitted tags are checked to ensure that they are members of that set.

The metadata repository provides a controlled location for storing member metadata and for producing federation metadata for distribution to federation members.

4.1. Metadata Submission

It is up to the federation, through its governance and operational processes, to determine which channels are provided to members for submitting their metadata to the metadata repository. Members typically have the option to upload the metadata directly to the repository, provided such functionality exists, or to send it to the federation operator through a designated secure channel. If an insecure channel is used, additional measures **MUST** be taken to verify the authenticity and integrity of the metadata. Such measures may include verifying the checksum of the metadata through another channel. The choice of submission channel may depend on factors such as the federation's guidelines and the preferences of the member.

4.2. Maintaining Up-to-Date Metadata

In a MATF federation, accurate and current metadata is essential for ensuring secure and reliable communication between members. This necessitates maintaining up-to-date metadata accessible by all members.

- Federation metadata: The federation operator publishes a JWS containing an aggregate of all entity metadata. This JWS serves as the source of truth for information about all members within the federation. Outdated information in the JWS can lead to issues such as failed connections, discovery challenges, and potential security risks.
- Local metadata: Each member maintains a local metadata store containing information about other members within the federation. This information is retrieved from the federation's publicly accessible JWS. Outdated data in the local store can hinder a member's ability to discover and connect with other relevant entities.

The following outlines the procedures for keeping metadata up to date:

- Federation Operator Role: The federation operator plays a crucial role in maintaining data integrity within the federation. Their responsibilities include:
  - Defining rules for metadata management that **MUST** include, at a minimum, expiration and cache time management.
  - Implementing mechanisms to update the published federation metadata, ensuring it adheres to the expiration time (`exp` as defined in Section 6.1) and cache TTL (`cache_ttl` as defined in Section 6.1) specifications.

- Member Responsibility: Members must follow the federation's metadata management rules and refresh their local metadata store according to the defined expiration and cache regulations.

By adhering to these responsibilities, the federation ensures that information remains valid for the defined timeframe and that caching mechanisms utilize up-to-date data effectively.

5. Authentication

All communication established within the federation uses TLS 1.3 [RFC8446] with mutual authentication. This mechanism ensures the authenticity of both communicating parties, establishing a robust foundation for secure data exchange.

5.1. Public Key Pinning

MATF implements public key pinning based on [RFC7469]. Public key pinning associates one or more public key pins with each federation endpoint. These pins are published in the federation metadata. During a connection, clients and servers extract the public key from the presented certificate and verify that it matches the preconfigured public key pins retrieved from the federation metadata.

5.1.1. Benefits of Public Key Pinning

The decision to utilize public key pinning in the MATF framework was driven by several critical factors aimed at enhancing security and ensuring trust.

5.1.1.1. Interfederation Trust

In interfederation environments, where multiple federations need to trust each other, public key pinning remains effective. Members can validate entities in other federations using pins published through shared metadata, ensuring trust across boundaries. Unlike private certificate chains, which can become complex and difficult to manage across multiple federations, public key pinning provides a straightforward mechanism for establishing trust. MATF interfederation addresses this challenge by aggregating metadata from all participating federations into a unified metadata repository. This shared metadata enables secure communication between entities in different federations, ensuring consistent key validation and robust cross-federation trust and security.

5.1.1.2. Fortifying Security Against Threats

Public key pinning provides a robust defense mechanism by directly binding a peer to a specific public key. This ensures that only the designated key is trusted, preventing attackers from exploiting fraudulent certificates. By eliminating reliance on external trust intermediaries, this approach significantly enhances resilience against potential threats.

5.1.1.3. Use of Self-Signed Certificates

The use of self-signed certificates within the federation leverages public key pinning to establish trust. By bypassing external Certificate Authorities (CAs), servers and clients rely on the federation's mechanisms to validate trust. Public key pinning ensures that only the specific self-signed public keys, identified by key pins in the metadata, are trusted.

5.1.1.4. Revocation

In deployments that rely on certificate chains and certificate revocation mechanisms, revocation can be complex and slow. This complexity arises because a certificate that can no longer be trusted, and potentially other certificates within the chain, may need to be revoked and reissued. Public key pinning mitigates this complexity by allowing clients to base trust decisions on pinned public keys rather than on certificate chains.

If a public key can no longer be trusted within a MATF federation, the associated pin is removed. Updated metadata is published. The updated metadata includes a new pin corresponding to the public key in the replacement certificate. This approach reduces reliance on certificate revocation mechanisms and shifts the trust relationship to the specific, updated public key identified by its pin.

5.2. Pin Discovery and Preloading

Peers in the federation obtain public key pins from the federation metadata. These pins serve as preconfigured trust parameters used for validation, as specified in Section 5.3.

The federation **MUST** define discovery rules. These rules describe how peers use federation metadata claims such as organization and tags to identify relevant endpoints and their pins.

Before initiating or accepting a connection, a peer **MUST** preload the pins for the selected or authorized endpoints from its local metadata store. Maintenance of the local metadata store, including refresh behavior and expiry handling, is specified in Section 4.2.

To support peer identification, the preloaded state **MUST** enable mapping from a derived pin to the corresponding entity_id. This may be achieved by maintaining a local index that maps each preloaded pin value to its associated `entity_id`.

A server **MAY** preload only the pins for clients that satisfy the server's connection policy (for example, based on `organization` or `tags`). Pin validation enforces the resulting policy as specified in Section 5.3.

5.3. Verification of Received Certificates

Upon connection establishment, both endpoints **MUST** verify that the public key in the presented peer certificate matches a pin published in the federation metadata. This validation **MAY** be performed by the TLS stack or by application logic.

In architectures where an intermediary terminates the TLS session, pin validation **MUST** be performed by either the intermediary or the application. If the application performs pin validation, the intermediary **MUST** forward the peer certificate or a derived pin to the application. The application **MUST** be able to determine the peer `entity_id` from the forwarded information and the federation metadata. This resolution relies on the client pin digest uniqueness property specified in Section 6.1.1.1.

If the intermediary performs pin validation, it **MUST** propagate the peer certificate, the derived pin, or the `entity_id` to the application to enable authorization.

The channel between the intermediary and the application MUST be integrity protected and **MUST** provide endpoint authentication.

Any conveyed certificate, pin, or identity used for this purpose **MUST** be derived directly from the TLS session. Implementations **MUST NOT** accept these values from peer-supplied application data.

If the implementation permits disabling default CA-based certificate chain validation, it **SHOULD** do so while still enforcing pin validation. If chain validation is required, the trust anchors used for certificate chain validation **MUST** be selected from the issuers listed in the federation metadata.

If no matching pin is found for a peer, the connection **MUST** be handled according to Section 5.4.

5.4. Failure to Validate

A received certificate that fails validation **MUST** result in the immediate termination of the connection. This includes scenarios where the derived pin does not match any preloaded pin or where the peer identity cannot be resolved. This strict enforcement ensures that only authorized and secure communication channels are established within the federation.

5.5. Certificate Rotation

To replace a certificate, whether due to expiration or other reasons, the following procedure **MUST** be followed:

1. Submitting updated metadata: When a certificate is scheduled for rotation, the federation member submits updated metadata that adds the `pin` for the new public key alongside the already published `pins`. The federation operator republishes the signed federation metadata aggregate, making the new pin available to all federation members.
2. Propagation period: Federation members **MUST** refresh their local metadata stores as specified in Section 4.2. The rotating member **MUST** allow sufficient time for peers to refresh and preload the new pin before switching to the new certificate.
3. Switching to the new certificate: After the propagation period has elapsed, the rotating member updates its TLS stack to present the new certificate. This allows peers that have preloaded the new pin to validate the rotated certificate.

4. Removing the old pin: Following a successful transition, the rotating member **MUST** submit updated metadata excluding the old pin. The federation operator republishes the aggregate, ensuring that only current public keys remain trusted within the federation.

5.6. Implementation Guidelines

The placement of pin validation depends on the deployment architecture. For clients, validation is typically performed by the component initiating the TLS connection. For servers using an intermediary, the communication channel between the intermediary and the application **MUST** be integrity protected to prevent tampering with forwarded peer identity material.

When an intermediary propagates peer identity material (for example, the peer certificate, a derived pin, or the `entity_id`) using HTTP header fields, those header fields are the mechanism used to fulfill the requirements specified in Section 5.3. For each header field name used for this purpose, the intermediary **MUST** remove any instance of that header field received from the peer and then set the header field value itself. This ensures that the application only processes identity material derived directly from the TLS session, enabling the application to match the peer to the federation metadata and apply authorization policy based on federation metadata claims. Header fields that are not used to convey identity material are unaffected by this requirement. The communication channel between the intermediary and the application MUST provide integrity protection and endpoint authentication to prevent tampering with forwarded peer identity material.

Implementations **SHOULD**, when possible, rely on libraries with built-in support for pinning. libcurl, for example, supports pinning via the CURLOPT_PINNEDPUBLICKEY option. In Python, the cryptography library can extract public keys, and application code can compare the derived pin to a configured value. Go provides crypto/tls and crypto/x509 for certificate inspection and public key extraction. In Java, java.security.cert.X509Certificate enables public key extraction, while java.net.http.HttpClient allows pinning enforcement using a custom SSLContext and TrustManager. The choice of library is left to the discretion of each implementation.

6. Federation Metadata

Federation metadata is published as a JSON Web Signature (JWS) [RFC7515]. The payload contains statements about entities of federation members.

Metadata is used for authentication and service discovery. A client selects a server based on metadata claims such as `organization` and `tags`. To establish a connection, the client uses the `base_uri`, `pins`, and, if needed, `issuers` of the selected server.

6.1. Federation Metadata Claims

This section defines the set of claims that can be included in metadata.

- iat (**REQUIRED**)

Identifies the time at which the federation metadata was issued.

- Data Type: Integer
- Syntax: NumericDate as defined in [RFC7519], Section 4.1.6.
- Example: `1755514949`

- exp (**REQUIRED**)

  Identifies the expiration time on or after which the federation metadata is no longer valid. Once the `exp` time has passed, the metadata **MUST** be rejected regardless of cache state.

  - Data Type: Integer
  - Syntax: NumericDate as defined in [RFC7519], Section 4.1.4.
  - Example: `1756119888`

- iss (**REQUIRED**)

  A URI uniquely identifying the issuing federation. This value differentiates federations, prevents ambiguity, and ensures that entities are recognized within their intended context. Verification of the `iss` claim enables recipients to determine the origin of the information and to establish trust with entities within the identified federation.

  - Data Type: String
  - Syntax: StringOrURI as defined in [RFC7519], Section 4.1.1. In MATF, this value **MUST** be a URI.
  - Example: `"https://federation.example.org"`

- version (**REQUIRED**)

  Indicates the schema version of the federation metadata. This ensures compatibility between members of the federation by defining a clear versioning mechanism for interpreting metadata.

  - Data Type: String
  - Syntax: The value **MUST** follow Semantic Versioning (see <https://semver.org>).
  - Example: `"1.0.0"`

- cache_ttl (**OPTIONAL**)

  Specifies the duration in seconds for caching downloaded federation metadata, allowing for independent caching outside of specific HTTP configurations. This is particularly useful when the communication mechanism is not based on HTTP. In the event of a metadata publication outage, members can rely on cached metadata until it expires, as indicated by the `exp` claim in the JWS payload, defined in Section 6.1. Once expired, metadata **MUST** no longer be trusted. If omitted, a mechanism to refresh metadata **MUST** still exist to ensure the metadata remains valid.

  - Data Type: Integer
  - Syntax: Integer representing the duration in seconds.
  - Example: `3600`

- entities (**REQUIRED**)

  Contains the list of entities within the federation.

  - Data Type: Array of Objects
  - Syntax: Each object **MUST** conform to the entity definition, as specified in Section 6.1.1.

6.1.1. Entities

Metadata contains a list of entities that may be used for communication within the federation. Each entity describes one or more endpoints owned by a member. An entity has the following properties:

- entity_id (**REQUIRED**)

  A URI that uniquely identifies the entity. This identifier **MUST NOT** collide with any other `entity_id` within the federation or within any other federation that the entity interacts with.

  - Data Type: String
  - Syntax: URI as defined in [RFC3986].
  - Example: `"https://example.com"`

- organization (**OPTIONAL**)

  A name identifying the organization that the entity's metadata represents. The federation operator **MUST** ensure that a mechanism is in place to verify that the `organization` claim corresponds to the rightful owner of the information exchanged between nodes. This is crucial for the trust model, ensuring certainty about the identities of the involved parties. The federation operator **SHOULD** choose an approach that best suits the specific needs and trust model of the federation.

  - Data Type: String
  - Syntax: A name identifying the organization represented by the entity.
  - Example: `"Example Org"`

- issuers (**REQUIRED**)

  A list of certificate issuers allowed to issue certificates for the entity's endpoints. For each issuer, the issuer's root CA certificate **MUST** be included in the `x509certificate` property and be encoded using the Privacy-Enhanced Mail (PEM) format. Certificate verification relies on public key pinning, with the list of allowed issuers used only when a certificate chain validation mechanism is unavoidable. For self-signed certificates, the certificate itself acts as its own issuer and **MUST** be listed as such in the metadata.

  - Data Type: Array of Objects
  - Syntax: Each object contains an issuer certificate encoded as PEM, as specified in [RFC7468]. The Base64 content **MUST** be wrapped so that each line consists of exactly 64 characters, except for the final line. In JSON text, line breaks in the PEM value are represented using the "\n" escape sequence.
  - Example: Issuer truncated for readability.

```
"issuers": [{
  "x509certificate": "-----BEGIN CERTIFICATE-----\nMIIDDD"
}]
```

- servers (**OPTIONAL**)

  Contains the list of servers within the entity.

  - Data Type: Array of Objects
  - Syntax: Each object **MUST** conform to the server definition, as specified in Section 6.1.1.1.

- clients (**OPTIONAL**)

  Contains the list of clients within the entity.

  - Data Type: Array of Objects
  - Syntax: Each object **MUST** conform to the client definition, as specified in Section 6.1.1.1.

6.1.1.1. Servers / Clients

The entity's servers and clients are listed below.

- description (**OPTIONAL**)

  A human-readable text describing the server or client.

  - Data Type: String
  - Syntax: Free-form text describing the server or client.
  - Example: `"SCIM Server 1"`

- base_uri (**REQUIRED** for servers, **OPTIONAL** for clients)

  The base URL of the server. This claim is **REQUIRED** for server endpoints. The value **MUST** be an absolute URI as defined in Section 4.3 of [RFC3986]. The value serves as the base URI for resolving relative references to server resources, as described in Section 5 of [RFC3986].

  - Data Type: String
  - Syntax: An absolute URI as defined in Section 4.3 of [RFC3986] that is used as a URL.
  - Example: `"https://scim.example.com/"`

- pins (**REQUIRED**)

  A list of objects representing public key pins [RFC7469].

  - Data Type: Array of Objects
  - Syntax: A list of objects, where each object represents a single public key pin with the following properties:
    - alg (**REQUIRED**)

The name of the cryptographic hash algorithm. Currently, the **RECOMMENDED** value is 'sha256'. As more secure algorithms are developed over time, federations should be ready to adopt these newer options for enhanced security.

- Data Type: String
- Syntax: The name of the algorithm.
- Example: `"sha256"`

- digest (**REQUIRED**)

The public key of the end-entity certificate, converted to a Subject Public Key Information (SPKI) fingerprint, as specified in Section 2.4 of [RFC7469]. For clients, the `digest` value MUST be unique across entities in the federation metadata to enable unambiguous identification of the peer. Within the same entity, the same `digest` value MAY be assigned to multiple clients.

- Data Type: String
- Syntax: SPKI fingerprint.
- Example: `"+hcmCjJEtLq4BRPhrILyhgn98Lhy6DaWdpmsBAgOLCQ="`

- Example:

```
"pins": [{
  "alg": "sha256",
  "digest": "+hcmCjJEtLq4BRPhrILyhgn98Lhy6DaWdpmsBAgOLCQ="
}]
```

- tags (**OPTIONAL**)

A list of strings that describe the endpoint's capabilities.

- Data Type: Array of Strings
- Syntax: Strings describing endpoint capabilities.
- Pattern: `^[a-z0-9]{1,64}$`
- Example: `["scim", "xyzzy"]`

Tags are fundamental for discovery within a federation, aiding both servers and clients in identifying appropriate connections.

- Server Tags: Tags associated with servers are used by clients to discover servers offering the services they require. Clients can search for servers based on tags that indicate supported protocols or the type of data they handle, enabling discovery of compatible servers.
- Client Tags: Tags associated with clients are used by servers to identify clients with specific characteristics or capabilities. For instance, a server might only accept connections from clients that support particular protocols. By filtering incoming requests based on these tags, servers can identify suitable clients.

Federation-Specific Considerations: While tags are tied to individual federations and serve distinct purposes within each, several key considerations are crucial to ensure clarity and promote consistent tag usage:

- Well-Defined Scope: Each federation **MUST** establish a clear scope for its tags, detailing their intended use, allowed tag values, associated meanings, and any relevant restrictions. Maintaining a well-defined and readily accessible registry of approved tags is essential for the federation.
- Validation Mechanisms: Implementing validation mechanisms for tags is highly recommended. This can involve a dedicated operation or service verifying tag validity and compliance with the federation's regulations. Such validation ensures consistency within the federation by preventing the use of unauthorized or irrelevant tags.

6.2. Metadata Schema

The MATF metadata schema is defined in Appendix A. This schema specifies the format for describing entities involved in MATF and their associated information.

> Note: The schema in Appendix A is folded due to line length limitations as specified in [RFC8792].

6.3. Example Metadata

The following is a non-normative example of a metadata statement. Line breaks in the `issuers` claim example are for readability only.

```
{
  "iat": 1755514949,
  "exp": 1756119888,
  "iss": "https://federation.example.org",
  "version": "1.0.0",
  "cache_ttl": 3600,
  "entities": [{
    "entity_id": "https://example.com",
    "organization": "Example Org",
    "issuers": [{
      "x509certificate": "-----BEGIN CERTIFICATE-----\nMIIDDDCCAf
      SgAwIBAgIJAIOsfJBStJQhMA0GCSqGSIb3DQEBCwUAMBsxGTAXBgNV\nBAM
      MEHNjaW0uZXhhbXBsZS5jb20wHhcNMTcwNDA2MDc1MzE3WhcNMTcwNTA2MD
      c1\nMzE3WjAbMRkwFwYDVQQDDBBzY2ltLmV4YW1wbGUuY29tMIIBIjANBgk
      qhkiG9w0B\nAQEFAAOCAQ8AMIIBCgKCAQEAyr+3dXTC8YXoi0LDJTH0lTfv
      8omQivWFOr3+/PBE\n6hmpLSNXK/EZJBD6ZT4Q+tY8dPhyhzT5RFZCVlrDs
      e/kY00F4yoflKiqx9WSuCrq\nZFr1AUtIfGR/LvRUvDFtuHo1MzFttiK8Wr
      wskMYZrw1zLHTIVwBkfMw1qr2XzxFK\njt0CcDmFxNdY5Q8kuBojH9+xt5s
      ZbrJ9AVH/OI8JamSqDjk9ODyGg+GrEZFClP/B\nxa4Fsl04En/9GfaJnCU1
      NpU0cqvWbVUlLOy8DaQMN14HIdkTdmegEsg2LR/XrJkt\nho16diAXrgS25
      3xbkdD3T5d6lHiZCL6UxkBh4ZHRcoftSwIDAQABo1MwUTAdBgNV\nHQ4EFg
      QUs1dXuhGhGc2UNb7ikn3t6cBuU34wHwYDVR0jBBgwFoAUs1dXuhGhGc2U\
      nNb7ikn3t6cBuU34wDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0BAQsFAA
      OCAQEA\nrR9wxPhUa2XfQ0agAC0oC8TFf8wbTYb0ElP5Ej834xMMW/wWTSA
      N8/3WqOWNQJ23\nf0vEeYQwfvbD2fjLvYTyM2tSPOWrtQpKuvulIrxV7Zz8
```

```
      A61NIjblE3rfea1eC8my\nTkDOlMKV+wlXXgUxirride+6ubOWRGf92fgze
      DGJWkmm/a9tj0L/3e0xIXeujxC7\nMIt3p99teHjvnZQ7FiIBlvGc1o8FD1
      FKmFYd74s7RxrAusBEAAmBo3xyB89cFU0d\nKB2fkH2lkqiqkyOtjrlHPoy
      6ws6g1S6U/Jx9n0NEeEqCfzXnh9jEpxisSO+fBZER\npCwj2LMNPQxZBqBF
      oxbFPw==\n-----END CERTIFICATE-----"
    }],
    "servers": [{
      "description": "SCIM Server 1",
      "base_uri": "https://scim.example.com/",
      "pins": [{
        "alg": "sha256",
        "digest": "+hcmCjJEtLq4BRPhrILyhgn98Lhy6DaWdpmsBAgOLCQ="
      }],
      "tags": [
        "scim"
      ]
    }],
    "clients": [{
      "description": "SCIM Client 1",
      "pins": [{
        "alg": "sha256",
        "digest": "+hcmCjJEtLq4BRPhrILyhgn98Lhy6DaWdpmsBAgOLCQ="
      }]
    }]
  }]
}
```

6.4. Metadata Signing

Federation metadata is signed using JWS and published using JWS JSON Serialization according to the general JWS JSON Serialization syntax defined in [RFC7515]. Federation metadata signatures are **RECOMMENDED** to be created using the algorithm ECDSA using P-256 and SHA-256 ("ES256") as defined in [RFC7518]. However, to accommodate evolving cryptographic standards, alternative algorithms **MAY** be used, provided they meet the security requirements of the federation.

Federations may need to transition to post-quantum cryptographic algorithms for federation metadata signatures and for endpoint certificate public key types. MATF can accommodate such transitions through key rollover and by updating published pins as new key types are deployed.

The following JWS Protected Header parameters are **REQUIRED**:

- `alg` (Algorithm)

  Identifies the algorithm used to generate the JWS signature [RFC7515], Section 4.1.1.

- `kid` (Key Identifier)

  Identifies the key in the issuer's key set that was used to generate the JWS signature [RFC7515], Section 4.1.4.

6.5. Example Signature Protected Header

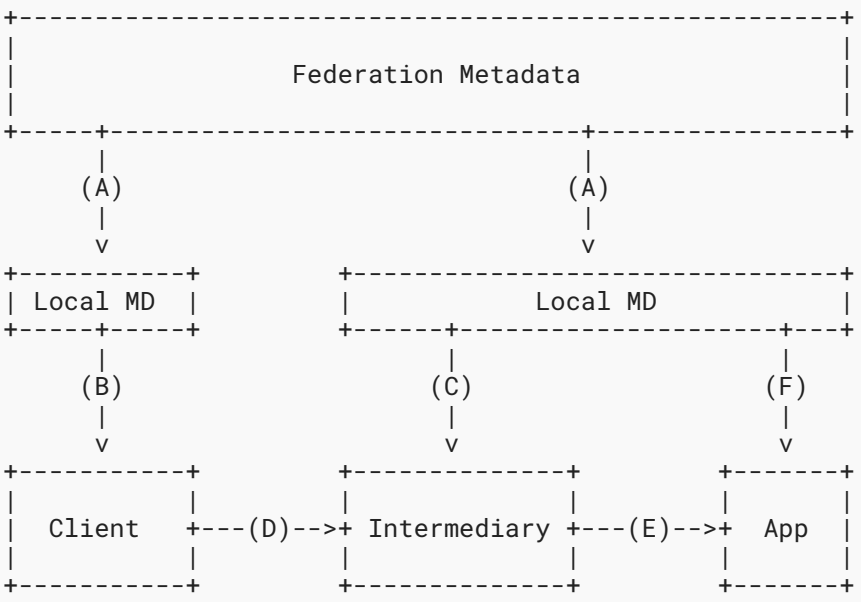
The following is a non-normative example of a signature protected header.

```
{
    "alg": "ES256",
    "kid": "c2fb760e-f4b6-4f7e-b17a-7115d2826d51"
}
```

7. Example Usage Scenarios

The examples in this section are not normative and illustrate the procedures described in Sections 5.2 and 5.3.

The following example describes a scenario within the federation "Skolfederation" where MATF is deployed. Both clients and servers are registered members of the federation. In this scenario, clients manage cross-domain user accounts using SS 12000:2018, which is a System for Cross-domain Identity Management (SCIM) extension.



A. Clients and servers retrieve federation metadata and update their local metadata stores as described in Section 4.2.

B. The client selects a server endpoint based on metadata claims and preloads the pins published for that endpoint.

C. If certificate chain validation is performed, the TLS client or intermediary configures its trust store using the `issuers` listed in the federation metadata for the selected entity.

D. The client initiates a TLS connection to the selected `base_uri` and presents its client certificate.

E. If an intermediary terminates the TLS session, it forwards identity material derived from the TLS session to the application as described in Sections 5.3 and 5.6.

F. The application maps the derived pin to a matching metadata entry and uses the associated `entity_id` for identification and authorization.

7.1. Client Behavior

A certificate is issued for the client. The client's certificate issuer and public key `pins` are published in the federation metadata.

When a client initiates a connection to a remote server (identified by the server's `entity_id`), the following steps are performed:

1. The client selects a server endpoint from the identified entity's `servers` list whose `tags` match the required service capabilities.
2. The client preloads the selected endpoint's `pins` from its local metadata store. If certificate chain validation is performed, the client also loads the `issuers` listed for the entity.
3. The client initiates a TLS connection to the selected endpoint using the `base_uri` and presents its client certificate.
4. The client performs pin validation for the server certificate as described in Section 5.3. This validation may be performed by the TLS stack during the handshake or by application logic after the connection is established, but it completes before any application data is exchanged.
5. If validation succeeds, the client proceeds with application transactions.

7.2. Server Behavior

To accept inbound connections from a client, the server uses federation metadata to perform pin validation of the public key in the presented client certificate. The federation metadata publishes client public key `pins`, and for deployments that perform certificate chain validation, the allowed `issuers`.

When the server receives a TLS connection attempt from a remote client, the following steps are performed:

1. The server is configured to request or require a client certificate. If certificate chain validation is performed, the trust store is populated using the `issuers` published in the federation metadata. Otherwise, the server requests a client certificate without issuer validation (for example, `optional_no_ca`).
2. The server can prefilter the federation metadata to identify the set of clients it is willing to communicate with and preload only the `pins` for those clients, as described in Section 5.2.
3. After the TLS handshake completes, the server derives the client's pin from the presented certificate and matches it against the preloaded pins. When a match is found, the server determines the client's `entity_id` from the corresponding metadata entry.
4. If pin validation succeeds, the server proceeds with application transactions. If pin validation fails, the server terminates the connection.

7.3. SPKI Generation

The following is an example of how to use OpenSSL to generate a SPKI fingerprint from a PEM-encoded certificate.

```
openssl x509 -in <certificate.pem> -pubkey -noout | \
openssl pkey -pubin -outform der | \
openssl dgst -sha256 -binary | \
openssl enc -base64
```

7.4. Curl and Public Key Pinning

The following is an example of public key pinning with curl.

```
curl --cert client.pem --key client.key \
--pinnedpubkey \
'sha256//0Ok2aNfcrCNDMhC2uXIdxBFOvMfEVtzlNVUT5pur0Dk=' \
https://host.example.com
```

8. Deployments of the MATF Framework

The MATF framework has proven its practical value and robustness through successful deployments in several environments.

8.1. Skolfederation Moa

Skolfederation Moa [Moa] is a federation designed to secure communication between digital educational resources and schools. MATF was developed to meet Moa's needs and enables secure data exchange for schools, municipalities, educational platforms, and services across Sweden.

The community plays a crucial role in this type of federation. Members are active participants, and the federation operator ensures the federation runs smoothly and serves their needs. Moa's success highlights the importance of collaboration, with members and the federation operator working together to maintain trust, security, and interoperability in the education sector.

The deployment of MATF in the Swedish education sector has provided several key insights. Maintaining an accurate registry of metadata ownership with reliable contact information is essential for troubleshooting and ensuring accountability. The deployment also demonstrated the importance of setting reasonable expiration times for metadata. Too short an expiration can hinder the ability to implement contingency plans for publishing new metadata during outages.

Metadata validation is necessary to maintain a stable federation. While manual validation may be sufficient in the early stages of a federation, it becomes unmanageable as the federation scales. Without an automated validation process, incorrect metadata uploaded by members is likely to go undetected, leading to publication of incorrect metadata.

The federation metadata signing private key is required to publish signed federation metadata. In fallback scenarios, federation metadata may be retrieved from an alternate location, but publishing updated federation metadata requires access to the signing private key. Therefore, secure and redundant management of the signing private key is necessary to support fallback mechanisms and reliable publication. Recipients **MUST** validate the JWS signature using the federation signature verification key before using federation metadata, regardless of where it is obtained.

8.2. Swedish National Agency for Education

The Swedish National Agency for Education [SkolverketMATF] leverages MATF within its digital national test platform to establish a robust authentication mechanism. The platform utilizes an API for client verification prior to secure data transfer to the agency's test service, ensuring the integrity and confidentiality of educational data.

8.3. Sambruk's EGIL

Sambruk's EGIL [EGIL], a platform providing digital services to municipalities, has successfully integrated the MATF framework. This deployment demonstrates the framework's adaptability to support a wide range of digital service infrastructures.

These deployments highlight the effectiveness of the MATF framework in enhancing security and interoperability within the educational sector.

9. Security Considerations

9.1. Security Risks and Trust Management

The security risks associated with the MATF framework are confined to each individual federation. Both the federation operator and federation members share the responsibility of maintaining trust and security. Proper handling of metadata and thorough vetting of members are crucial to sustaining this trust.

Deployments that terminate a session at an intermediary and convey identity material to an application introduce a critical trust boundary. If the intermediary is compromised or fails to properly sanitize inbound headers, an attacker could spoof a peer's `entity_id`. Therefore, intermediaries that convey identity material to an application **MUST** comply with the requirements in Section 5.6.

Implementations **SHOULD** avoid logging conveyed certificates, pins, or identity values unless required for diagnostics to prevent the accidental exposure of session-specific identity material.

9.2. TLS

The security considerations for TLS 1.3 are detailed in Section 10 and Appendices C, D, and E of [RFC8446].

9.3. Federation Metadata Updates

Regularly updating the local copy of federation metadata is essential for accessing the latest information about active entities, current public key pins [RFC7469], and valid issuer certificates. The use of outdated metadata may expose systems to security risks, such as interaction with revoked entities or acceptance of manipulated data.

9.4. Verifying the Federation Metadata Signature

Ensuring data integrity and security within the MATF framework relies on verifying the signature of downloaded federation metadata. This verification confirms the origin of the metadata by validating the JWS signature using the federation signature verification key trusted by the recipient. It also confirms that the signed content has not been altered by unauthorized parties. By verifying the signature, trust is maintained in the integrity of the information used for validation, including member public key pins and issuer certificates. To achieve a robust implementation, it is important to consider the security aspects outlined in [RFC7515], which describes security considerations related to algorithm selection, key compromise, and signature integrity.

9.5. Time Synchronization

Maintaining synchronized clocks across all federation members is critical for the security of the MATF framework. Inaccurate timestamps can compromise the validity of digital signatures and certificates, hinder reliable log analysis, and potentially expose the system to time-based attacks. Therefore, all federation members **MUST** employ methods to ensure their system clocks are synchronized with a reliable time source.

10. IANA Considerations

This document has no IANA actions.

11. References

11.1. 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>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC7468] Josefsson, S. and S. Leonard, "Textual Encodings of PKIX, PKCS, and CMS Structures", RFC 7468, DOI 10.17487/RFC7468, April 2015, <https://www.rfc-editor.org/info/rfc7468>.

[RFC7469] Evans, C., Palmer, C., and R. Sleevi, "Public Key Pinning Extension for HTTP", RFC 7469, DOI 10.17487/RFC7469, April 2015, <https://www.rfc-editor.org/info/rfc7469>.

[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/info/rfc7515>.

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <https://www.rfc-editor.org/info/rfc7517>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://www.rfc-editor.org/info/rfc7518>.

[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>.

[RFC7638] Jones, M. and N. Sakimura, "JSON Web Key (JWK) Thumbprint", RFC 7638, DOI 10.17487/RFC7638, September 2015, <https://www.rfc-editor.org/info/rfc7638>.

[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>.

11.2. Informative References

[eduGAIN] eduGAIN, "eduGAIN: Interfederation service connecting research and education identity federations worldwide", <https://edugain.org>.

[EGIL] Sambruk, "EGIL – smidig hantering av skolans digitala användarkonton" [EGIL – manage your school's digital user accounts efficiently], <https://sambruk.se/egil-dnp/>.

[eIDAS] European Union, "Consolidated text: Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC", 18 October 2024, <https://eur-lex.europa.eu/eli/reg/2014/910>.

[Moa] Internetstiftelsens Federationer [The Swedish Internet Foundation], "Machine and Organization Authentication", 6 October 2025, <https://wiki.federationer.internetstiftelsen.se/x/LYA5AQ>.

[RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <https://www.rfc-editor.org/info/rfc8792>.

[SkolverketMATF] Skolverket [Swedish National Agency for Education], "API för autentisering" [Authentication API for User Management], commit f8c2e93, 4 September 2025, <https://github.com/skolverket/dnp-usermanagement/blob/main/authentication-api/README.md>.

Appendix A. JSON Schema for MATF Metadata

The following JSON Schema defines the structure of MATF metadata. It conforms to draft 2020-12 of the JSON Schema standard.

Version: 1.0.0

```
=============== NOTE: '\\' line wrapping per RFC 8792 ===============

{
    "$schema": "https://json-schema.org/draft/2020-12/schema",
    "$id": "https://mtlsfed.se/schema/matf-metadata-schema.json",
    "title": "JSON Schema for Mutually Authenticating TLS in the con\
\text of Federations",
    "description": "Version: 1.0.0",
    "type": "object",
    "additionalProperties": true,
    "required": [
        "iat",
        "exp",
        "iss",
        "version",
        "entities"
    ],
    "properties": {
        "iat": {
            "title": "Issued at",
            "description": "Time at which the metadata was issued (U\
\NIX timestamp)",
            "type": "integer",
            "minimum": 0,
            "examples": [
                1755514949
            ]
        },
        "exp": {
            "title": "Expiration time",
            "description": "Time at which the metadata expires (UNIX\
\ timestamp)",
            "type": "integer",
```

```
            "minimum": 0,
            "examples": [
                1756119888
            ]
        },
        "iss": {
            "title": "The federation issuing the metadata",
            "description": "A URI that uniquely identifies the feder\
\ation that issued the metadata",
            "type": "string",
            "format": "uri",
            "minLength": 1,
            "examples": [
                "https://example.com/federation"
            ]
        },
        "version": {
            "title": "Metadata schema version",
            "description": "Schema version follows semantic versioni\
\ng (https://semver.org)",
            "type": "string",
            "pattern": "^\\d+\\.\\d+\\.\\d+$",
            "examples": [
                "1.0.0"
            ]
        },
        "cache_ttl": {
            "title": "Metadata cache TTL",
            "description": "How long in seconds to cache metadata. T\
\he effective maximum is bounded by the exp claim.",
            "type": "integer",
            "minimum": 0,
            "examples": [
                3600
            ]
        },
        "entities": {
            "type": "array",
            "minItems": 1,
            "items": {
                "$ref": "#/$defs/entity"
            }
        }
    },
    "$defs": {
        "entity": {
            "type": "object",
            "additionalProperties": true,
            "required": [
                "entity_id",
                "issuers"
            ],
            "properties": {
                "entity_id": {
                    "title": "Entity identifier",
                    "description": "Globally unique identifier for t\
\he entity.",
                    "type": "string",
```

```
                    "format": "uri",
                    "examples": [
                        "https://example.com"
                    ]
                },
                "organization": {
                    "title": "Name of entity organization",
                    "description": "Name identifying the organizatio\
\n that the entity's metadata represents.",
                    "type": "string",
                    "examples": [
                        "Example Org"
                    ]
                },
                "issuers": {
                    "title": "Entity certificate issuers",
                    "description": "A list of certificate issuers th\
\at are allowed to issue certificates for the entity's endpoints. Fo\
\r each issuer, the issuer's root CA certificate is included in the \
\x509certificate property (PEM-encoded).",
                    "type": "array",
                    "minItems": 1,
                    "items": {
                        "$ref": "#/$defs/cert_issuers"
                    }
                },
                "servers": {
                    "type": "array",
                    "items": {
                        "$ref": "#/$defs/endpoint"
                    }
                },
                "clients": {
                    "type": "array",
                    "items": {
                        "$ref": "#/$defs/endpoint"
                    }
                }
            }
        },
        "endpoint": {
            "type": "object",
            "additionalProperties": true,
            "required": [
                "pins"
            ],
            "properties": {
                "description": {
                    "title": "Endpoint description",
                    "type": "string",
                    "examples": [
                        "SCIM Server 1"
                    ]
                },
                "tags": {
                    "title": "Endpoint tags",
                    "description": "A list of strings that describe \
\the endpoint's capabilities.",
```

```
                    "type": "array",
                    "items": {
                        "type": "string",
                        "pattern": "^[a-z0-9]{1,64}$",
                        "examples": [
                            "xyzzy"
                        ]
                    }
                },
                "base_uri": {
                    "title": "Endpoint base URI",
                    "type": "string",
                    "format": "uri",
                    "examples": [
                        "https://scim.example.com"
                    ]
                },
                "pins": {
                    "title": "Certificate pin set",
                    "type": "array",
                    "minItems": 1,
                    "items": {
                        "$ref": "#/$defs/pin_directive"
                    }
                }
            }
        },
        "cert_issuers": {
            "title": "Certificate issuers",
            "type": "object",
            "additionalProperties": false,
            "required": [
                "x509certificate"
            ],
            "properties": {
                "x509certificate": {
                    "title": "X.509 Certificate (PEM)",
                    "type": "string",
                    "pattern": "^-----BEGIN CERTIFICATE-----(?:\\r?\\
\\n)(?:[A-Za-z0-9+/=]{64}\\r?\\n)*(?:[A-Za-z0-9+/=]{1,64}\\r?\\n)---\
\--END CERTIFICATE-----(?:\\r?\\n)?$"
                }
            }
        },
        "pin_directive": {
            "title": "RFC 7469 pin directive",
            "type": "object",
            "additionalProperties": false,
            "required": [
                "alg",
                "digest"
            ],
            "properties": {
                "alg": {
                    "title": "Directive name",
                    "type": "string",
                    "enum": [
                        "sha256"
```

```
                    ],
                    "examples": [
                        "sha256"
                    ]
                },
                "digest": {
                    "title": "Directive value (Base64)",
                    "type": "string",
                    "pattern": "^[A-Za-z0-9+/]{43}=$",
                    "examples": [
                        "HiMkrb4phPSP+OvGqmZd6sGvy7AUn4k3XEe8OMBrzt8\
\="
                    ]
                }
            }
        }
    }
}
```

Acknowledgements

This project was funded through the NGI0 PET Fund, a fund established by NLnet with financial support from the European Commission's Next Generation Internet programme, under the aegis of DG Communications Networks, Content and Technology under grant agreement No 825310.

The authors thank the following people for the detailed review and suggestions:

- Rasmus Larsson
- Mats Dufberg
- Joe Siltberg
- Stefan Norberg
- Petter Blomberg

The authors would also like to thank participants in the EGIL working group for their comments on this specification.

Authors' Addresses

Stefan Halén
The Swedish Internet Foundation
Email: stefan.halen@internetstiftelsen.se

Jakob Schlyter
Kirei AB
Email: jakob@kirei.se