Workload Identity in Multi System Environments
Internet-Draft
Intended status: Standards Track
Expires: 7 May 2026

B. Campbell
Ping Identity
A. Schwenkschuster
Defakto Security

3 November 2025

WIMSE Workload Proof Token
draft-ietf-wimse-wpt-00

Abstract

The WIMSE architecture defines authentication and authorization for software workloads in a variety of runtime environments, from basic deployments to complex multi-service, multi-cloud, multi-tenant systems. This document specifies the Workload Proof Token (WPT), a mechanism for workloads to prove possession of the private key associated with a Workload Identity Token (WIT). The WPT is a signed JWT that binds the workload's authentication to a specific HTTP request, providing application-level proof of possession for workload-to-workload communication. This specification is designed to work alongside the WIT credential format defined in draft-ietf-wimse-workload-creds and can be combined with other WIMSE protocols in multi-hop call chains.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://ietf-wg-wimse.github.io/draft-ietf-wimse-s2s-protocol/draft-ietf-wimse-wpt.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-wimse-wpt/.

Discussion of this document takes place on the Workload Identity in Multi System Environments Working Group mailing list (mailto:wimse@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/wimse/. Subscribe at https://www.ietf.org/mailman/listinfo/wimse/.

Source for this draft and an issue tracker can be found at https://github.com/ietf-wg-wimse/draft-ietf-wimse-s2s-protocol.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

This Internet-Draft will expire on 7 May 2026.

Copyright Notice

Table of Contents

1. Introduction

This document defines the Workload Proof Token (WPT), a simple, protocol-independent mechanism for proving possession of the private key associated with a Workload Identity Token (WIT). The WIT, defined in [I-D.ietf-wimse-workload-creds], is a credential that binds a public key to a workload identity and is designed to require proof of possession - it must not be used as a bearer token. The WPT provides that proof of possession. The WPT's primary design goal is simplicity: it is a signed JWT that demonstrates control of the private key corresponding to the public key in the WIT. By requiring this cryptographic proof, the WPT significantly reduces the risk of credential theft and replay attacks compared to bearer token approaches. The WPT is protocol-agnostic by design. While this specification provides detailed guidance for HTTP-based usage (including the Workload-Proof-Token HTTP header field), the core WPT format is fundamentally a signed JWT that can be adapted to other protocols including asynchronous messaging systems, event-driven architectures, and future transport mechanisms. The JWT-based structure allows for protocol-specific extensions through additional claims while maintaining core interoperability.

Key characteristics of the WPT include:

Proof of Possession: Demonstrates control of the WIT's associated private key through a digital signature. Context Binding: Binds the proof to specific message context through claims such as audience (aud) and WIT hash (wth). Other tokens in the message can also be bound (e.g., OAuth access tokens, transaction tokens) to provide unified proof across different authorization contexts. Short-Lived: Typically valid for minutes or seconds, limiting replay attack windows. Protocol Independent: Core format is not tied to any specific transport protocol.

This specification is part of the WIMSE protocol suite, which includes credential formats defined in [I-D.ietf-wimse-workload-creds] and follows the architectural principles in [I-D.ietf-wimse-arch]. The WPT provides application-level proof of possession particularly suited for environments where transport-level solutions are insufficient or where communication patterns span multiple protocols. This document defines the WPT JWT format, its HTTP usage, validation requirements, and security considerations. Out of scope are the WIT credential format itself (covered in [I-D.ietf-wimse-workload-creds]), policy enforcement and authorization, credential issuance and lifecycle management, detailed bindings for non-HTTP protocols (to be addressed in future specifications), and alternative proof-of-possession mechanisms such as HTTP Message Signatures.

2. Workload Proof Token

The Workload Proof Token (WPT) is a JWT that provides proof of possession of the private key associated with a Workload Identity Token (WIT). The Workload Identity Token is sent in the request as described in [I-D.ietf-wimse-workload-creds]. The WPT, is sent in the Workload-Proof-Token header field of the request. The ABNF syntax of the Workload-Proof-Token header field is:

```
WPT =  JWT
```

Figure 1: Workload-Proof-Token Header Field ABNF

where the JWT projection is defined in Figure 1.

A WPT MUST contain the following:

* in the JOSE header:
  - alg: An identifier for an appropriate JWS asymmetric digital signature algorithm corresponding to the confirmation key in the associated WIT. The value MUST match the alg value of the jwk in the cnf claim of the WIT. See [I-D.ietf-wimse-workload-creds] for valid values and restrictions.
  - typ: the WPT is explicitly typed, as recommended in Section 3.11 of [RFC8725], using the application/wpt+jwt media type.
* in the JWT claims:

- aud: The audience SHOULD contain the HTTP target URI (Section 7.1 of [RFC9110]) of the request to which the WPT is attached, without query or fragment parts. However, there may be some normalization, rewriting or other process that requires the audience to be set to a deployment-specific value.

- exp: The expiration time of the WPT (as defined in Section 4.1.4 of [RFC7519]). WPT lifetimes MUST be short, e.g., on the order of minutes or seconds.

- jti: An identifier for the token. The value MUST be unique, at least within the scope of the sender.

- wth: Hash of the Workload Identity Token, defined in [I-D.ietf-wimse-workload-creds]. The value is the base64url encoding of the SHA-256 hash of the ASCII encoding of the WIT's value.

- ath: Hash of the OAuth access token, if present in the request, which might convey end-user identity and/or authorization context of the request. The value, as per Section 4.1 of [RFC9449], is the base64url encoding of the SHA-256 hash of the ASCII encoding of the access token's value.

- tth: Hash of the Txn-Token [I-D.ietf-oauth-transaction-tokens], if present in the request, which might convey end-user identity and/or authorization context of the request. The value MUST be the result of a base64url encoding (as defined in Section 2 of [RFC7515]) of the SHA-256 hash of the ASCII encoding of the associated token's value.

- oth: Hash(es) of other token(s) in the request that convey end-user identity and/or authorization context of the request. The value is a JSON object with a key-value pair for each such token. For each, in the absence of an application profile specifying details, the key corresponds to the header field name containing the token, and the value is the base64url encoding of the SHA-256 hash of the ASCII bytes of the header field value with any leading or trailing spaces removed. The header field name MUST be normalized by converting it to all lower case. Header fields occurring multiple times in the request are not supported by default. An application profile may specify different behavior for a key, such as using a different hash algorithm or means of locating the token in the request.

To clarify: the ath, tth and oth claims are each mandatory if the respective tokens are included in the request.

The rules for using non-standard claims in WPTs are documented in Section 2.3.

An example WPT might look like the following:


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

eyJhbGciOiJFZERTQSIsInR5cCI6IndwdCtqd3QifQ.eyJhdGgiOiJDTDR3amZwUm1OZ\
i1iZFlJYllMblY5ZDVyTUFSR3dLWUUxMHdVd3pDMGpJIiwiYXVkIjoiaHR0cHM6Ly93b\
3JrbG9hZC5leGFtcGxlLmNvbS9wYXRoIiwiZXhwIjoxNzQ1NTEwMDE2LCJqdGkiOiJfX\
2J3YzRFU0MzYWNjMkxUQzEtX3giLCJ3dGgiOiJBYVlVZkMzNEQxZGkyRnhRTHBpSUpKN\
1NnOFZaNm84T0Nkd1NmOUlUb0xnIn0.PI7d9AcYhLoEgPgbJvcM132lkBKnM1NXU-5hZ\
UzVTIyj2dRJaTLFs6e5NYv5gg6AqcV7NvkYCwfgMgWasWQzCA
```


Figure 2: Example Workload Proof Token (WPT)

The decoded JOSE header of the WPT from the example above is shown here:


```
{
  "alg": "EdDSA",
  "typ": "wpt+jwt"
}
```


Figure 3: Example WPT JOSE Header

The decoded JWT claims of the WPT from the example above are shown here:


```
{
  "ath": "CL4wjfpRmNf-bdYIbYLnV9d5rMARGwKYE10wUwzC0jI",
  "aud": "https://workload.example.com/path",
  "exp": 1740755048,
  "jti": "0c740386ca1dcad37de1b5f9de1b0705",
  "wth": "aA0W_oFJK7qV7zYhcmzR1KOXVCHjd2x6c4sOQLvE90Y"
}
```


Figure 4: Example WPT Claims

An example of an HTTP request with both the WIT and WPT from prior examples is shown below:

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

POST /path HTTP/1.1
Host: workload.example.com
Content-Type: application/json
Authorization: Bearer 16_mAd0GiwaZokU26_0902100
Workload-Identity-Token: eyJhbGciOiJFUzI1NiIsImtpZCI6Ikp1bmUgNSIsInR\
5cCI6IndpdCtqd3QifQ.eyJjbmYiOnsiandrIjp7ImFsZyI6IkVkRFNBIiwiY3J2Ijoi\
RWQyNTUxOSIsImt0eSI6Ik9LUCIsIngiOiIxQ1hYdmZsTl9MVlZzSXNZWHNVdkIwM0pt\
bEdXZUNIcVFWdW91Q0Y5MmJnIn19LCJleHAiOjE3NDU1MTI1MTAsImlhdCI6MTc0NTUw\
ODkxMCwianRpIjoieC1fMUNUTDJjY2EzQ1NFNGN3Yl9sIiwic3ViIjoid2ltc2U6Ly9l\
eGFtcGxlLmNvbS9zcGVjaWZpYy13b3JrbG9hZCJ9.6KraSQUxWdl9dxFQ3Fj6dPY0Vi8\
8OkwFwZpAIOhLeq6AbXAnLLQgOp8U9UDGcBuYF3KiNv6oKQD1ZWAzrMZOJw
Workload-Proof-Token: eyJhbGciOiJFZERTQSIsInR5cCI6IndwdCtqd3QifQ.eyJ\
hdGgiOiJDTDR3amZwUm1OZi1iZFlJYllMblY5ZDVyTUFSR3dLWUUxMHdVd3pDMGpJIiw\
iYXVkIjoiaHR0cHM6Ly93b3JrbG9hZC5leGFtcGxlLmNvbS9wYXRoIiwiZXhwIjoxNzQ\
1NTEwMDE2LCJqdGkiOiJfX2J3YzRFU0MzYWNjMkxUQzEtX3giLCJ3dGgiOiJBYVlVZkM\
zNEQxZGkyRnhRTHBpSUpKN1NnOFZaNm84T0Nkd1NmOUlUb0xnIn0.PI7d9AcYhLoEgPg\
bJvcM132lkBKnM1NXU-5hZUzVTIyj2dRJaTLFs6e5NYv5gg6AqcV7NvkYCwfgMgWasWQ\
zCA

{"do stuff":"please"}
```

Figure 5: Example HTTP Request with WIT and WPT

To validate the WPT in the request, the recipient MUST ensure the following:

* There is exactly one Workload-Proof-Token header field in the request.

* The Workload-Proof-Token header field value is a single and well-formed JWT.

* The signature algorithm in the alg JOSE header string-equal matches the alg attribute of the jwk in the cnf claim of the WIT.

* The WPT signature is valid using the public key from the confirmation claim of the WIT.

* The typ JOSE header parameter of the WPT conveys a media type of wpt+jwt.

* The aud claim of the WPT matches the target URI, or an acceptable alias or normalization thereof, of the HTTP request in which the WPT was received, ignoring any query and fragment parts.

* The exp claim is present and conveys a time that has not passed. WPTs with an expiration time unreasonably far in the future SHOULD be rejected.

* The wth claim is present and matches the hash of the token value conveyed in the Workload-Identity-Token header.

* It is RECOMMENDED to check that the value of the jti claim has not been used before in the time window in which the respective WPT would be considered valid.

* If presented in conjunction with an OAuth access token, the value of the ath claim matches the hash of that token’s value.

* If presented in conjunction with a Txn-Token, the value of the tth claim matches the hash of that token’s value.

* If presented in conjunction with a token conveying end-user identity or authorization context, the value of the oth claim matches the hash of that token’s value.

* If the oth claim is present, verify the hashes of all tokens listed in the oth claim per the default behavior defined in Section 2 or as specified by an application specific profile. If the oth claim contains entries that are not understood by the recipient, the WPT MUST be rejected. Conversely, additional tokens not covered by the oth claim MUST NOT be used by the recipient to make authorization decisions.

2.1. Error Conditions

Errors may occur during the processing of the WPT. If the signature verification fails for any reason, such as an invalid signature, an expired validity time window, or a malformed data structure, an error is returned. Typically, this will be in response to an API call, so an HTTP status code such as 400 (Bad Request) is appropriate. This response could include more details as per [RFC9457], such as an indicator that the wrong key material or algorithm was used. The use of HTTP status code 401 is NOT RECOMMENDED for this purpose because it requires a WWW-Authenticate with acceptable http auth mechanisms in the error response and an associated Authorization header in the subsequent request. The use of these headers for the WIT or WPT is not compatible with this specification.

2.2. Coexistence with JWT Bearer Tokens

The WIT and WPT define new HTTP headers. They can therefore be presented along with existing headers used for JWT bearer tokens. This property allows for transition from mechanisms using identity tokens based on bearer JWTs to proof of possession based WITs. A workload may implement a policy that accepts both bearer tokens and WITs during a transition period. This policy may be configurable per-caller to allow the workload to reject bearer tokens from callers that support WITs. Once a deployment fully supports WITs, then the use of bearer tokens for identity can be disabled through policy. Implementations should be careful when implementing such a transition strategy, since the decision which token to prefer is made when the caller's identity has still not been authenticated, and needs to be revalidated following the authentication step.

The WIT can also coexist with tokens used to establish security context, such as transaction tokens [I-D.ietf-oauth-transaction-tokens]. In this case a workload's authorization policy may take into account both the sending workload's identity and the information in the context token. For example, the identity in the WIT may be used to establish which API calls can be made and information in the context token may be used to determine which specific resources can be accessed.

2.3. Including Additional Claims

The WPT contains JSON structures and therefore can be trivially extended by adding more claims beyond those defined in the current specification. This, however, could result in interoperability issues, which the following rules are addressing.

* To ensure interoperability in WIMSE environments, the use of Private claim names (Sec. 4.3 of [RFC7519]) is NOT RECOMMENDED.

* In closed environments, deployers MAY freely add claims to the WPT. Such claims SHOULD be collision-resistant, such as example.com/myclaim.

* A recipient that does not understand such claims MUST ignore them, as per Sec. 4 of [RFC7519].

* Outside of closed environments, new claims MUST be registered with IANA [IANA.JWT.CLAIMS] before they can be used.

3. Security Considerations

3.1. Workload Identity Token and Proof of Possession

The Workload Identity Token (WIT) is bound to a secret cryptographic key and is always presented with a proof of possession as described in [I-D.ietf-wimse-workload-creds]. The WIT is a general purpose token that can be presented in multiple contexts. The WIT and WPT are only used in the application-level options, and both are not used in MTLS. The WIT MUST NOT be used as a bearer token. While this helps reduce the sensitivity of the token it is still possible that a token and its proof of possession may be captured and replayed within the PoP's lifetime. The following are some mitigations for the capture and reuse of the proof of possession (PoP):

* Preventing Eavesdropping and Interception with TLS

An attacker observing or intercepting the communication channel can view the token and its proof of possession and attempt to replay it to gain an advantage. In order to prevent this the token and proof of possession MUST be sent over a secure, server authenticated TLS connection unless a secure channel is provided by some other mechanisms. Host name validation MUST be performed by the client.

* Limiting Proof of Possession Lifespan

The proof of possession MUST be time limited. A PoP should only be valid over the time necessary for it to be successfully used for the purpose it is needed. This will typically be on the order of minutes. PoPs received outside their validity time MUST be rejected.

* Limiting Proof of Possession Scope

In order to reduce the risk of theft and replay the PoP should have a limited scope. For example, a PoP may be targeted for use with a specific workload and even a specific transaction to reduce the impact of a stolen PoP. In some cases a workload may wish to reuse a PoP for a period of time or have it accepted by multiple target workloads. A careful analysis is warranted to understand the impacts to the system if a PoP is disclosed allowing it to be presented by an attacker along with a captured WIT.

* Replay Protection

A proof of possession includes the jti claim that MUST uniquely identify it, within the scope of a particular sender. This claim SHOULD be used by the receiver to perform basic replay protection against tokens it has already seen. Depending upon the design of the

system it may be difficult to synchronize the replay cache across all token validators. If an attacker can somehow influence the identity of the validator (e.g. which cluster member receives the message) then replay protection would not be effective.

* Binding to TLS Endpoint

The POP MAY be bound to a transport layer sender such as the client identity of a TLS session or TLS channel binding parameters. The mechanisms for binding are outside the scope of this specification.

3.2. Workload Identity Key Management

The Workload Identity Token is signed by a private key in possession of the workload. This private key:

* MUST be kept private

* MUST be individual for each Workload Identifier (see [I-D.ietf-wimse-arch])

* MUST NOT be used once the Workload Identity Token is expired

* SHOULD be individual for each Workload Identity Token issued

* SHOULD not be reused for other purposes

3.3. Middle Boxes

In some deployments the Workload Identity Token and Workload Proof Token may pass through multiple systems. The communication between the systems is over TLS, but the WIT and WPT are available in the clear at each intermediary. While the intermediary cannot modify the token or the information within the PoP they can attempt to capture and replay the token or modify the data not protected by the PoP.

It is important to note that the WPT does not protect major portions of the request and response and therefore does not provide protection from an actively malicious middle box. Deployments should perform analysis on their situation to determine if it is appropriate to trust and allow traffic to pass through a middle box.

3.4. Privacy Considerations

The Workload Proof Token may contain private information such as user names or other identities. Care should be taken to prevent the disclosure of this information. The use of TLS helps protect the privacy of WITs and proofs of possession.

The workload identifier present in the WPT is typically associated with a workload and not a specific user, however in some deployments the workload or the HTTP Target URI may be associated directly to a user. While these are exceptional cases a deployment should evaluate if the disclosure of a WPT can be used to track a user.

4. IANA Considerations

4.1. JSON Web Token Claims

IANA is requested to add the following entries to the "JSON Web Token Claims" registry [IANA.JWT.CLAIMS]:

Claim Name	Claim Description	Change Controller	Reference
tth	Transaction Token hash	IETF	RFC XXX, Section 2
wth	Workload Identity Token hash	IETF	RFC XXX, Section 2
oth	Other Tokens hashes	IETF	RFC XXX, Section 2

Table 1

4.2. Media Type Registration

IANA is requested to register the following entries to the "Media Types" registry [IANA.MEDIA.TYPES]:

* application/wpt+jwt, per Section 4.2.1.

4.2.1. application/wpt+jwt

Type name: application

Subtype name: wpt+jwt

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: Encoding considerations are identical to those specified for the "application/jwt" media type. See [RFC7519].

Security considerations: See the Security Considerations section of RFC XXX.

Interoperability considerations: N/A

Published specification: RFC XXX, Section 2.

Applications that use this media type: Workloads that use these tokens to integrity-protect messages in the WIMSE workload-to-workload protocol.

Fragment identifier considerations: N/A

Additional information:

Deprecated alias names for this type: N/A

Magic number(s): N/A

File extension(s): None

Macintosh file type code(s): N/A

Person & email address to contact for further information:

See the Authors' Addresses section of RFC XXX.

Intended usage: COMMON

Restrictions on usage: N/A

Author: See the Authors' Addresses section of RFC XXX.

Change controller: Internet Engineering Task Force (iesg@ietf.org).

4.3. Hypertext Transfer Protocol (HTTP) Field Name Registration

IANA is requested to register the following entries to the "Hypertext Transfer Protocol (HTTP) Field Name Registry" [IANA.HTTP.FIELDS]:

* Workload-Proof-Token, per Section 4.3.1.

4.3.1. Workload-Proof-Token

* Field Name: Workload-Proof-Token

* Status: permanent

* Structured Type: N/A

* Specification Document: RFC XXX, Section 2

* Comments: see reference above for an ABNF syntax of this field

5. References

5.1. Normative References

[I-D.ietf-wimse-arch]
Salowey, J. A., Rosomakho, Y., and H. Tschofenig, "Workload Identity in a Multi System Environment (WIMSE) Architecture", Work in Progress, Internet-Draft, draft-ietf-wimse-arch-06, 30 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-wimse-arch-06>.

[I-D.ietf-wimse-workload-creds]
"*** BROKEN REFERENCE ***".

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/rfc/rfc5234>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/rfc/rfc7515>.

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

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://www.rfc-editor.org/rfc/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/rfc/rfc7519>.

[RFC7800] Jones, M., Bradley, J., and H. Tschofenig, "Proof-of-Possession Key Semantics for JSON Web Tokens (JWTs)", RFC 7800, DOI 10.17487/RFC7800, April 2016, <https://www.rfc-editor.org/rfc/rfc7800>.

[RFC8725] Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", BCP 225, RFC 8725, DOI 10.17487/RFC8725, February 2020, <https://www.rfc-editor.org/rfc/rfc8725>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[RFC9449] Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., and D. Waite, "OAuth 2.0 Demonstrating Proof of Possession (DPoP)", RFC 9449, DOI 10.17487/RFC9449, September 2023, <https://www.rfc-editor.org/rfc/rfc9449>.

5.2. Informative References

[I-D.ietf-oauth-transaction-tokens] Tulshibagwale, A., Fletcher, G., and P. Kasselman, "Transaction Tokens", Work in Progress, Internet-Draft, draft-ietf-oauth-transaction-tokens-06, 28 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-transaction-tokens-06>.

[IANA.HTTP.FIELDS] IANA, "Hypertext Transfer Protocol (HTTP) Field Name Registry", <https://www.iana.org/assignments/http-fields>.

[IANA.JOSE.ALGS] IANA, "JSON Web Signature and Encryption Algorithms", <https://www.iana.org/assignments/jose>.

[IANA.JWT.CLAIMS] IANA, "JSON Web Token Claims", <https://www.iana.org/assignments/jwt>.

[IANA.MEDIA.TYPES] IANA, "Media Types", <https://www.iana.org/assignments/media-types>.

[IANA.URI.SCHEMES] IANA, "Uniform Resource Identifier (URI) Schemes", <https://www.iana.org/assignments/uri-schemes>.

[RFC9457] Nottingham, M., Wilde, E., and S. Dalal, "Problem Details for HTTP APIs", RFC 9457, DOI 10.17487/RFC9457, July 2023, <https://www.rfc-editor.org/rfc/rfc9457>.

Appendix A. Document History

// RFC Editor: please remove before publication.

A.1. draft-ietf-wimse-wpt-00

* Focus on Workload Proof Token (WPT) only.
  - Remove credential formats (WIT and WIC)
  - Remove HTTP-Message-Signature profile

A.2. draft-ietf-wimse-s2s-protocol-07

* Rework the WPT's oth claim.
* update the media types.
* Discuss extensibility of WIT and WPT.
* Clarify error handling, specifically why not HTTP 401.
* Correct the code examples.
* Add registration request content for a wimse URI scheme.
* New section on key management.
* Use of the Accept-Signature header.

A.3. draft-ietf-wimse-s2s-protocol-06

* Explicit definition of the Workload Identity Certificate.
* Definition of the validation of workload identifiers as part of workload authentication. Still work in progress.

A.4. draft-ietf-wimse-s2s-protocol-05

* Removed the entire Workload Identity section which is now covered in the Architecture document.
* Content-Digest is mandatory with HTTP-Sig.
* Some wording on extending the protocol beyond HTTP.
* IANA considerations.

A.5. draft-ietf-wimse-s2s-protocol-04

* Require cnf.jwk.alg in WIT which restricts signature algorithm of WPT or HTTP-Sig.
* Replay protection as a SHOULD for both WPT and HTTP-Sig.
* Consolidate terminology with the Architecture draft.

A.6. draft-ietf-wimse-s2s-protocol-03

* Consistently use "workload".
* Implement comments from the SPIFFE community.
* Make iss claim in WIT optional and add wording about its relation to key distribution.
* Remove iss claim from WPT.
* Make jti claim in WIT optional.
* Error handling for the application level methods.

A.7. draft-ietf-wimse-s2s-protocol-02

* Coexistence with bearer tokens.
* Improve the architecture diagram.
* Some more ABNF.
* Clarified identifiers and URIs.
* Moved an author to acknowledgments.

A.8. draft-ietf-wimse-s2s-protocol-01

* Addressed multiple comments from Pieter.
* Clarified WIMSE identity concepts, specifically "trust domain" and "workload identifier".
* Much more detail around mTLS, including some normative language.
* WIT (the identity token) is now included in the WPT proof of possession.

* Added a section comparing the DPoP-inspired app-level security option to the Message Signature-based alternative.

A.9. draft-ietf-wimse-s2s-protocol-00

* Initial WG draft, an exact copy of draft-sheffer-wimse-s2s-protocol-00

* Added this document history section

Acknowledgments

The authors would like to thank Pieter Kasselman for his detailed comments.

We thank Daniel Feldman for his contributions to earlier versions of this document.

Authors' Addresses

Brian Campbell
Ping Identity
Email: bcampbell@pingidentity.com

Arndt Schwenkschuster
Defakto Security
Email: arndts.ietf@gmail.com