Workload Identity in Multi System Environments
Internet-Draft
Intended status: Standards Track
Expires: 27 March 2026

A. Schwenkschuster
SPIRL

23 September 2025

WIMSE Workload-to-Workload with Workload Proof Tokens
draft-schwenkschuster-s2s-jwt-pop-00

Abstract

This document defines a DPoP-inspired JWT-based profile for workload-to-workload authentication within the WIMSE (Workload Identity in Multi System Environments) architecture. This profile uses the Workload Identity Token (WIT) combined with a Workload Proof Token (WPT) to provide cryptographic proof of possession. The WPT is a signed JWT that demonstrates the sender's possession of the private key bound to the WIT, while also binding the authentication to the specific request context.

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-s2s-protocol.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-schwenkschuster-s2s-jwt-pop/.

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.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Copyright Notice

Table of Contents

1. Introduction

This document specifies a JWT-based proof of possession authentication profile for workload-to-workload communication as part of the WIMSE architecture defined in [I-D.ietf-wimse-arch]. This profile is inspired by the OAuth 2.0 Demonstration of Proof of Possession (DPoP) specification [RFC9449] and provides a JWT-centric approach to workload authentication.

In modern distributed systems, workloads often need to authenticate to each other in environments where end-to-end TLS is not available due to the presence of load balancers, API gateways, service meshes, and other middleboxes that terminate TLS connections. In such environments, traditional bearer tokens are insufficient due to security concerns around token theft and replay attacks, while the complexity of HTTP Message Signatures may not be warranted for all deployment scenarios. This profile addresses these needs by providing a proof of possession mechanism that builds upon familiar JWT technologies while ensuring that workload identities cannot be replayed or misused, even when passing through untrusted intermediaries.

This profile builds upon the base WIMSE workload-to-workload protocol and specifies the use of Workload Proof Tokens (WPTs) as the proof of possession mechanism. The approach combines:

1. The Workload Identity Token (WIT) - a JWT that establishes the workload’s identity and binds it to a cryptographic key

2. The Workload Proof Token (WPT) - a short-lived JWT signed by the workload’s private key that proves possession of that key

3. Request binding - the WPT includes claims that bind it to the specific request and associated tokens

The WPT includes hashes of the WIT and any other relevant tokens present in the request (such as OAuth access tokens or transaction tokens), ensuring that the proof of possession is bound to the complete request context. This prevents token substitution attacks and provides assurance that all tokens in the request are being used by their legitimate holder.

While this profile provides protection against token replay and ensures workload authentication in the presence of middleboxes, it does not provide the same level of message integrity protection as HTTP Message Signatures. Middleboxes can still modify unprotected portions of HTTP requests and responses. Deployments should carefully evaluate whether this level of protection is sufficient for their threat model, or whether the stronger integrity guarantees of HTTP Message Signatures are required.

This profile is particularly suitable for environments that already use JWT-based technologies and prefer to maintain consistency with OAuth 2.0 patterns. It provides a simpler alternative to HTTP Message Signatures while still offering strong authentication guarantees and protection against token replay attacks.

The profile defines the structure and validation requirements for WPTs, including mandatory claims, signature parameters, and the specific binding mechanisms that tie the proof to the request context. Like other WIMSE profiles, it is designed to be interoperable, allowing different authentication methods to be used by different workload pairs within the same distributed system as appropriate for their specific requirements and constraints.

2. Conventions and Definitions

All terminology in this document follows [I-D.ietf-wimse-arch].

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.

3. DPoP-Inspired Authentication

This option, inspired by the OAuth DPoP specification [RFC9449], uses a DPoP-like mechanism to authenticate the calling workload in the context of the request. The Workload Identity Token (WIT) as defined in Section 3.1 "The Workload Identity Token" of [I-D.ietf-wimse-s2s-protocol] is sent in the request as described in Section 3.1.1 "The WIT HTTP Header" of [I-D.ietf-wimse-s2s-protocol]. An additional JWT, the Workload Proof Token (WPT), is signed by the private key corresponding to the public key in the WIT. 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 Section 3.1.1 "The WIT HTTP Header" of [I-D.ietf-wimse-s2s-protocol].

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 Section 3.1 "The Workload Identity Token" of [I-D.ietf-wimse-s2s-protocol] 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. See also Section 2.3 "Workload Identifiers and Authentication Granularity" of [I-D.ietf-wimse-s2s-protocol] for more details.

  - 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 Section 3.1 "The Workload Identity Token" of [I-D.ietf-wimse-s2s-protocol]. 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 similar to the rules for WITs, as described in Section 3.1.2 "Including Additional Claims" of [I-D.ietf-wimse-s2s-protocol].

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. See also Section 2.3 "Workload Identifiers and Authentication Granularity" of [I-D.ietf-wimse-s2s-protocol] for implementation advice on this verification check.

* 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 3 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.

3.1. 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.

3.2. 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.

3.3. Client Authorization Using the Workload Identity

The server application retrieves the workload identifier from the client certificate subjectAltName, which in turn is obtained from the TLS layer. The identifier is used in authorization, accounting and auditing. For example, the full workload identifier may be matched against ACLs to authorize actions requested by the peer and the identifier may be included in log messages to associate actions to the client workload for audit purposes. A deployment may specify other authorization policies based on the specific details of how the workload identifier is constructed. The path portion of the workload identifier MUST always be considered in the scope of the trust domain. See Section 2.3 "Workload Identifiers and Authentication Granularity" of [I-D.ietf-wimse-s2s-protocol] on additional security implications of workload identifiers.

4. Security Considerations

4.1. 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 Section 3.1 "The Workload Identity Token" of [I-D.ietf-wimse-s2s-protocol]. The WIT is a general purpose token that can be presented in multiple contexts. The WIT and its PoP 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 according to Section 3.3.1 "Server Name Validation" of [I-D.ietf-wimse-s2s-protocol] 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.

4.2. Middle Boxes

In some deployments the Workload Identity Token and proof of possession may pass through multiple systems. The communication between the systems is over TLS, but the token and PoP 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.

Mitigations listed in Section 3 "Application Level Workload-to-Workload Authentication" of [I-D.ietf-wimse-s2s-protocol] can be used to provide some protection from middle boxes. However we note that the DPoP-inspired solution (Section 3) 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.

This profile provides authentication and limited integrity protection but does not protect the entire HTTP message. Deployments requiring stronger integrity guarantees should consider the HTTP Message Signatures profile defined in [I-D.ietf-schwenkschuster-s2s-http-sig].

4.3. Privacy Considerations

Privacy considerations for this specification are the same as those described in Section 4.4 "Privacy Considerations" of [I-D.ietf-wimse-s2s-protocol].

5. IANA Considerations

5.1. 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 3
wth	Workload Identity Token hash	IETF	RFC XXX, Section 3
oth	Other Tokens hashes	IETF	RFC XXX, Section 3

Table 1

5.2. Media Type Registration

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

* application/wpt+jwt, per Section 5.2.1.

5.2.1. 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 3.

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).

5.3. 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 5.3.1.

5.3.1. Workload-Proof-Token

* Field Name: Workload-Proof-Token
* Status: permanent
* Structured Type: N/A
* Specification Document: RFC XXX, Section 3
* Comments: see reference above for an ABNF syntax of this field

6. References

6.1. 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/rfc/rfc2119>.

[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>.

[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/rfc/rfc8174>.

[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>.

6.2. 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>.

[I-D.ietf-schwenkschuster-s2s-http-sig]
"*** BROKEN REFERENCE ***".

[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-05, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-wimse-arch-05>.

[I-D.ietf-wimse-s2s-protocol]
Campbell, B., Salowey, J. A., Schwenkschuster, A., and Y. Sheffer, "WIMSE Workload to Workload Authentication", Work in Progress, Internet-Draft, draft-ietf-wimse-s2s-protocol-06, 4 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-wimse-s2s-protocol-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>.

[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>.

[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-schwenkschuster-s2s-jwt-pop-00

Initial clone from original draft-ietf-wimse-s2s-06 which contained both, Workload Proof Token and HTTP Message Signature proof of possession mechanisms.

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.

Author’s Address

Arndt Schwenkschuster
SPIRL
Email: arndts.ietf@gmail.com