Workload Identity in Multi System Environments
Internet-Draft
Intended status: Standards Track
Expires: 6 November 2026

B. Campbell
Ping Identity
J. Salowey
CyberArk
A. Schwenkschuster
Defakto Security
Y. Sheffer
Intuit
Y. Rosomakho
Zscaler
5 May 2026

WIMSE Workload Credentials
draft-ietf-wimse-workload-creds-01

Abstract

The WIMSE architecture defines authentication and authorization for software workloads in a variety of runtime environments, from the most basic ones up to complex multi-service, multi-cloud, multi-tenant deployments.

This document defines the credentials that workloads use to represent their identity. They can be used in various protocols to authenticate workloads to each other. To use these credentials, workloads must provide proof of possession of the associated private key material, which is covered in other documents. This document focuses on the credentials alone, independent of the proof-of-possession mechanism.

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-workload-creds.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-wimse-workload-creds/.

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

Copyright Notice

Table of Contents

1. Introduction

This document defines authentication and authorization in the context of interaction between two workloads. This is the core component of the WIMSE architecture [I-D.ietf-wimse-arch]. This document focuses on the credentials that carry workload identity and bind the key material to the identity. The presentation of the proof of possession of the key material is part of other documents and out of scope for this one.

In this document, two credentials are defined:

* The Workload Identity Token (WIT) is a JWT that represents the identity of a workload and binds a public key to that identity.

* The Workload Identity Certificate (WIC) is an X.509 certificate that represents the identity of a workload and binds a public key to that identity.

The Workload Identity Token is targeted for application-level protocols. The Workload Identity Certificate is targeted for transport-level protocols. This does not preclude the use of the WIT in transport-level protocols or the WIC in application-level protocols, but these are the primary intended uses.

The various protocol bindings that use these credentials to authenticate workloads to each other are out of scope for this document. At the time of writing, three such protocols are defined:

* Transport level authentication mutual TLS using the Workload Identity Certificate.

* Application level authentication using the Workload Identity Token in conjunction with a JWT-based proof of possession, the Workload Proof Token (WPT).

* Application level authentication using the Workload Identity Token in conjunction with HTTP Message Signatures.

1.1. Use In Other Protocols

The credentials defined in this document are designed to be used in various protocols. This document does not define the protocols themselves, but rather the credentials that can be used within them. Additional protocols MAY be defined in the future that use these credentials for workload authentication.

1.2. Deployment Architecture and Message Flow

Independent of the transport between the workloads, we assume the following logical architecture (numbers refer to the sequence of steps listed below):

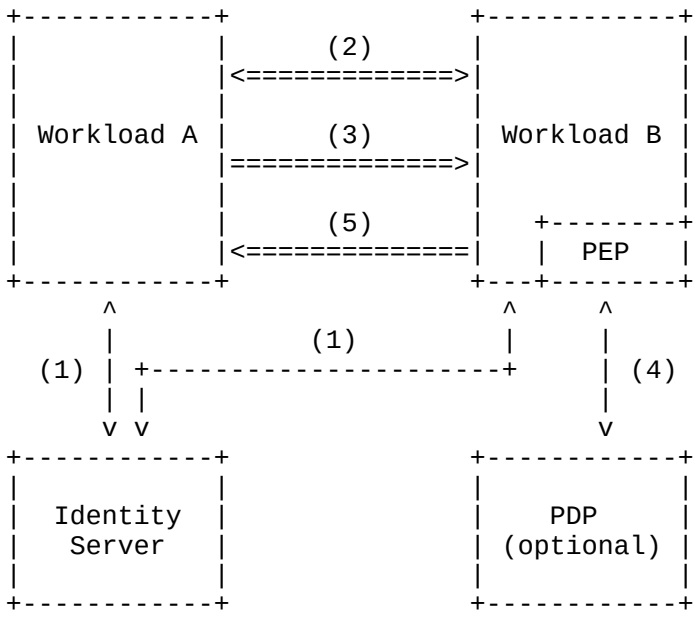

Figure 1: Sequence of Operations

The Identity Server provisions credentials to each of the workloads. At least Workload A (and possibly both) must be provisioned with a credential before the call can proceed. Details of communication with the Identity Server are out of scope of this document, however we do describe the credential received by the workload.

PEP is a Policy Enforcement Point, the component that allows the message to go through or blocks it. PDP is an optional Policy Decision Point, which may be deployed in architectures where policy management is centralized. All details of policy management and message authorization are out of scope of this document.

The high-level message flow is as follows:

1. Workload A (and similarly, Workload B) obtains a credential from the Identity Server. This happens periodically, e.g. once every 24 hours.

2. A transport connection is set up. This may already include the use of the Workload Identity Certificate with transport-level security, such as TLS.

3. Workload A prepares to call Workload B. This may include adding application-level authentication information, such as the Workload Identity Token and proof of possession. Workload B authenticates Workload A.

4. Workload B authorizes the call. This policy enforcement (Policy Enforcement Point, PEP) may include consulting with an external server (Policy Decision Point, PDP) when making this decision.

5. Workload B returns a response to Workload A, which may be an error response or a regular one.

Depending on the protocol, the workload authentication may happen during step (2) at the transport-level or at step (3) at the application-level, or both.

1.3. Workload Identifiers and Authentication Granularity

The Workload Identifier is a URI and its baseline syntax and processing requirements are defined in [WIMSE-ID]. While deployments define how they assign identifiers and what the path portion means, implementations MUST enforce the URI requirements outlined in Section 4.1 of [WIMSE-ID].

Prior to WIMSE, many use cases did not allow for fully granular authentication in containerized runtime platforms. For instance, with mutual TLS, there’s often no clear way to map the request’s external access reference (e.g., Kubernetes Ingress path, service name, or host header) to the SubjectAltName value in the server certificate. This means that the client could only verify if the server certificate is valid within a trust domain, not if it’s tied to a specific workload.

To enable mutual and granular authentication between workloads, two things must be in place:

* Each workload must know its own identifier.

* There needs to be an explicit mapping from the external handle used to access a workload (such as an Ingress path or service DNS name) to its workload identifier.

Once these conditions are met, the methods described in this document can be used for the caller and callee to mutually authenticate.

Implementations MUST allow for defining this mapping between the workload's access path and the workload identifier (e.g., through callback functions). Deployments SHOULD use these features to establish a consistent set of identifiers within their environment.

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. Application Level Workload-to-Workload Authentication

In many deployments communication between workloads cannot use end-to-end transport security such as TLS. For these deployment styles, this document proposes a credential that can be used at the application level.

3.1. The Workload Identity Token

The Workload Identity Token (WIT) is a JWS [RFC7515] signed JWT [RFC7519] that represents the identity of a workload. It is issued by the Identity Server and binds a public key to the workload identity. See Section 6.3 for security considerations.

A WIT MUST contain the following content, except where noted:

* in the JOSE header:
  - alg: An identifier for a JWS asymmetric digital signature algorithm (registered algorithm identifiers are listed in the IANA JOSE Algorithms registry [IANA.JOSE.ALGS]). The value none MUST NOT be used.
  - typ: the WIT is explicitly typed, as recommended in Section 3.11 of [RFC8725], using the wit+jwt media type.
* in the JWT claims:

- iss: The issuer of the token, which is the Identity Server, represented by a URI. The iss claim is RECOMMENDED but optional; when present, it is particularly useful for auditing and operations (for example, identifying which Identity Server issued the WIT in logs or compliance records). See Section 3.1.3 for key distribution and validation context.

- sub: The subject of the token, which is the identity of the workload, represented by a Workload Identifier (a URI) as defined in [WIMSE-ID]. Section 1.3 provides additional requirements associated with these identifiers, so they can be used to secure workload-to-workload communication.

- exp: The expiration time of the token (as defined in Section 4.1.4 of [RFC7519]). WITs should be refreshed regularly, e.g. on the order of hours.

- jti: A unique identifier for the token. This claim is OPTIONAL. The jti claim is frequently useful for auditing issuance of individual WITs or to revoke them, but some token generation environments do not support it.

- cnf: A confirmation claim referencing the public key of the workload.

  o jwk: Within the cnf claim, a jwk key MUST be present that contains the public key of the workload as defined in Section 3.2 of [RFC7800]. The workload MUST prove possession of the corresponding private key when presenting the WIT to another party. As such, it MUST NOT be used as a bearer token and is not intended for use in the Authorization header.

    + alg: Within the jwk object, an alg field MUST be present. Allowed values are listed in the IANA "JSON Web Signature and Encryption Algorithms" registry established by [RFC7518]. The presented proof MUST be produced with the algorithm specified in this field. The value none MUST NOT be used. Algorithms used in combination with symmetric keys MUST NOT be used. Also encryption algorithms MUST NOT be used as this would require additional key distribution outside of the WIT. To promote interoperability, the ES256 signing algorithm MUST be supported by general purpose implementations of this document.

As noted in [WIMSE-ID], a workload identifier is a URI that includes a trust domain in the authority component. The runtime environment often determines which URI scheme is used, e.g. if SPIFFE is used to authenticate workloads, it mandates "spiffe" URIs. For deployments that do not use an environment-specific scheme, the wimse URI scheme MAY be used; it is defined in [WIMSE-ID], which also registers it with IANA.

An example WIT might look like this:


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

eyJhbGciOiJFUzI1NiIsImtpZCI6Ikp1bmUgNSIsInR5cCI6IndpdCtqd3QifQ.eyJjb\
mYiOnsiandrIjp7ImFsZyI6IkVkRFNBIiwiY3J2IjoiRWQyNTUxOSIsImt0eSI6Ik9LU\
CIsIngiOiIxQ1hYdmZsTl9MVlZzSXNZWHNVdkIwM0ptbEdXZUNIcVFWdW91Q0Y5MmJnI\
n19LCJleHAiOjE3NDU1MTI1MTAsImlhdCI6MTc0NTUwODkxMCwianRpIjoieC1fMUNUT\
DJjY2EzQ1NFNGN3Yl9sIiwic3ViIjoid2ltc2U6Ly9leGFtcGxlLmNvbS9zcGVjaWZpY\
y13b3JrbG9hZCJ9.6KraSQUxWdl9dxFQ3Fj6dPY0Vi88OkwFwZpAIOhLeq6AbXAnLLQg\
Op8U9UDGcBuYF3KiNv6oKQD1ZWAzrMZOJw
```


Figure 2: An example Workload Identity Token (WIT)

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


```
{
  "alg": "ES256",
  "kid": "June 5",
  "typ": "wit+jwt"
}
```


Figure 3: Example WIT JOSE Header

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

```
{
  "cnf": {
    "jwk": {
      "alg": "EdDSA",
      "crv": "Ed25519",
      "kty": "OKP",
      "x": "1CXXvflN_LVVsIsYXsUvB03JmlGWeCHqQVuouCF92bg"
    }
  },
  "exp": 1745512510,
  "iat": 1745508910,
  "jti": "x-_1CTL2cca3CSE4cwb_l",
  "sub": "wimse://example.com/specific-workload"
}
```


Figure 4: Example WIT Claims

The claims indicate that the example WIT:

* is valid until Thu Apr 24 2025 16:35:10 GMT (represented as NumericDate Section 2 of [RFC7519] value 1745512510).

* identifies the workload to which the token was issued as wimse://example.com/specific-workload.

* has a unique identifier of x-_1CTL2cca3CSE4cwb_l.

* binds the public key represented by the jwk confirmation method to the workload wimse://example.com/specific-workload.

* requires the proof to be produced with the EdDSA signature algorithm.

For elucidative purposes only, the workload’s key, including the private part, is shown below in JWK [RFC7517] format:


```
{
 "kty": "OKP",
 "crv": "Ed25519",
 "x": "1CXXvflN_LVVsIsYXsUvB03JmlGWeCHqQVuouCF92bg",
 "d": "sdLX8yCYKqo_XvGBLn-ZWeKT7llYeeQpgeCaXVxb5kY"
}
```


Figure 5: Example Workload’s Key

The afore-exampled WIT is signed with the private key of the Identity Server. The public key(s) of the Identity Server need to be known to all workloads in order to verify the signature of the WIT. The Identity Server's public key from this example is shown below in JWK [RFC7517] format:

```
{
 "kty": "EC",
 "kid": "June 5",
 "crv": "P-256",
 "x": "kXqnA2Op7hgd4zRMbw0iFcc_hDxUxhojxOFVGjE2gks",
 "y": "n__VndPMR021-59UAs0b9qDTFT-EZtT6xSNs_xFskLo"
}
```

Figure 6: Example Identity Server Key

3.1.1. The WIT HTTP Header

A WIT is conveyed in an HTTP header field named Workload-Identity-Token.

ABNF [RFC5234] for the value of Workload-Identity-Token header field is provided in Figure 7:

```
ALPHA = %x41-5A / %x61-7A ; A-Z / a-z
DIGIT = %x30-39 ; 0-9
base64url = 1*(ALPHA / DIGIT / "-" / "_")
JWT =  base64url "." base64url "." base64url
WIT =  JWT
```

Figure 7: Workload-Identity-Token Header Field ABNF

The following shows the WIT from Figure 2 in an example of a Workload-Identity-Token header field:

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

Workload-Identity-Token: eyJhbGciOiJFUzI1NiIsImtpZCI6Ikp1bmUgNSIsInR\
5cCI6IndpdCtqd3QifQ.eyJjbmYiOnsiandrIjp7ImFsZyI6IkVkRFNBIiwiY3J2Ijoi\
RWQyNTUxOSIsImt0eSI6Ik9LUCIsIngiOiIxQ1hYdmZsTl9MVlZzSXNZWHNVdkIwM0pt\
bEdXZUNIcVFWdW91Q0Y5MmJnIn19LCJleHAiOjE3NDU1MTI1MTAsImlhdCI6MTc0NTUw\
ODkxMCwianRpIjoieC1fMUNUTDJjY2EzQ1NFNGN3Yl9sIiwic3ViIjoid2ltc2U6Ly9l\
eGFtcGxlLmNvbS9zcGVjaWZpYy13b3JrbG9hZCJ9.6KraSQUxWdl9dxFQ3Fj6dPY0Vi8\
8OkwFwZpAIOhLeq6AbXAnLLQgOp8U9UDGcBuYF3KiNv6oKQD1ZWAzrMZOJw
```

Figure 8: An example Workload Identity Token HTTP Header Field

Note that per [RFC9110], header field names are case insensitive; thus, Workload-Identity-Token, workload-identity-token, WORKLOAD-IDENTITY-TOKEN, etc., are all valid and equivalent header field names. However, case is significant in the header field value.

3.1.2. Including Additional Claims

The WIT 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 WIT. 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.

3.1.3. A note on iss claim and key distribution

It is RECOMMENDED that the WIT carries an iss claim, including for the auditing and operational uses described above. Validators are not required to use iss when validating the WIT or establishing the workload identity: the trust domain and workload identity are carried in the mandatory sub claim ([WIMSE-ID]). Implementations MAY include the iss claim in the form of a https URL to facilitate key distribution via mechanisms like the jwks_uri from [RFC8414]; alternative key distribution methods may use only the trust domain from the sub claim.

3.2. Error Conditions

Errors may occur during the processing of the WIT. If the WIT validation 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 is not compatible with this specification.

3.3. Coexistence with JWT Bearer Tokens

The WIT defines new HTTP headers. It 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.

4. Transport Level Workload-to-Workload Authentication

As noted in the introduction, for many deployments, transport-level protection of application traffic is ideal.

4.1. The Workload Identity Certificate

The Workload Identity Certificate is an X.509 certificate. The workload identity MUST be encoded in a SubjectAltName extension of type URI. There MUST be only one SubjectAltName extension of type URI in a Workload Identity Certificate. If the workload will act as a TLS server for clients that do not understand workload identities it is RECOMMENDED that the Workload Identity Certificate contain a SubjectAltName of type DNSName with the appropriate DNS names for the server. The certificate MAY contain SubjectAltName extensions of other types.

4.2. Client Authorization Using the Workload Identity

The server application retrieves the workload identifier from the client certificate subjectAltName. 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 1.3 on additional security implications of workload identifiers.

5. Implementation Status

// Note to RFC Editor: please remove this section, as well as the // reference to RFC 7942, before publication. This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

SPIFFE (Standard)

* Organization: CNCF

* Maturity:

  - Workload Identity Certificate: fully compatible with the X509-SVID and widely used.

  - Workload Identity Token: beta

* Coverage: Workload Identity Certificate, WIT

* Contact: SPIFFE sig-spec community (https://github.com/spiffe/spiffe/tree/main/community/sig-spec)

Defakto Security

* Organization: Defakto Security (prior SPIRL)

* Maturity:

  - Workload Identity Certificate: production

  - Workload Identity Token: alpha

* Coverage: Workload Identity Certificate, WIT

* Contact: arndt@defakto.security

Teleport - Machine & Workload Identity

* Organization: Teleport

* Maturity:

  - Workload Identity Certificate: production

  - Workload Identity Token: research

* Coverage: Workload Identity Certificate

* Contact: noah@goteleport.com

6. Security Considerations

6.1. Workload Identity

Workload Identifiers ([WIMSE-ID]) are scoped to a trust domain (the URI authority component) and MUST be interpreted in that trust domain context. Using a Workload Identifier without taking into account the trust domain could allow one domain to issue tokens to spoof identities in another domain. Consumers MUST bind each trust domain to an authorized issuer (or set of issuers) and to the corresponding cryptographic trust anchors used to validate credentials from that domain (for example using a JWKS or an X.509 certificate chain). The mapping from trust domain to authorized issuers and trust anchors MUST be distributed to consumers via a secure out-of-band mechanism.

When a deployment uses the iss claim for key distribution as described in Section 3.1.3, validators MUST enforce an allowlist of accepted issuers. Absent such a restriction, any entity could stand up an issuer, present a WIT with that issuer's iss value, and have it accepted by a validator that fetches and trusts the corresponding key material without verifying the issuer's legitimacy.

6.2. 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 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.

* 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

Proof of possession mechanisms should include replay protection to prevent reuse of a captured PoP. Without it an attacker can replay a captured PoP within its validity period.

* 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.

6.3. Workload Identity Key Management

Both the Workload Identity Token and the Workload Identity Certificate carry a public key. The corresponding private key:

* MUST be kept private

* MUST be bound to a single Workload Identifier ([WIMSE-ID])

* MUST NOT be used once the credential is expired

* SHOULD be re-generated for each new Workload Identity Token or Certificate.

6.4. 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 can be used to provide some protection from middle boxes.

Deployments should perform analysis on their situation to determine if it is appropriate to trust and allow traffic to pass through a middle box.

6.5. Privacy Considerations

WITs and the proofs of possession 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.

WITs and certificates with workload identifiers are typically associated with a workload and not a specific user, however in some deployments the workload may be associated directly to a user. While these are exceptional cases a deployment should evaluate if the disclosure of WITs or certificates can be used to track a user.

7. IANA Considerations

7.1. Media Type Registration

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

* application/wit+jwt, per Section 7.1.1.

7.1.1. application/wit+jwt

Type name: application

Subtype name: wit+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 3.1.

Applications that use this media type: Identity servers that vend Workload Identity Tokens, and Workloads that use these tokens to authenticate to each other.

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).

7.2. 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-Identity-Token, per Section 7.2.1.

7.2.1. Workload-Identity-Token

* Field Name: Workload-Identity-Token
* Status: permanent
* Structured Type: N/A
* Specification Document: RFC XXX, Section 3.1.1
* Comments: see reference above for an ABNF syntax of this field

8. References

8.1. 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>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/rfc/rfc7942>.

[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>.

[RFC8414] Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Authorization Server Metadata", RFC 8414, DOI 10.17487/RFC8414, June 2018, <https://www.rfc-editor.org/rfc/rfc8414>.

[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>.

[WIMSE-ID] Rosomakho, Y. and J. A. Salowey, "Workload Identifier", Work in Progress, Internet-Draft, draft-ietf-wimse-identifier-02, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-wimse-identifier-02>.

8.2. 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-08, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-transaction-tokens-08>.

[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-07, 2 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-wimse-arch-07>.

[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>.

[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-workload-creds-01

* Clarify that the iss claim is RECOMMENDED in part for auditing and operations, and that validation of workload identity uses sub, not iss.

* Remove the wimse URI scheme definition and IANA registration from this document; both are specified in [WIMSE-ID].

A.2. draft-ietf-wimse-workload-creds-00

* Remove WPT, HTTP-Sig and mutual TLS sections, which are going to be covered by individual documents. This includes re-phrasing of various sections to focus on the credentials only.

A.3. 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.4. 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.5. 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.6. 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.7. 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.8. 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.9. 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.10. 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

Joe Salowey
CyberArk
Email: joe@salowey.net

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

Yaron Sheffer
Intuit
Email: yaronf.ietf@gmail.com

Yaroslav Rosomakho
Zscaler
Email: yrosomakho@zscaler.com