Workload Identity in Multi System Environments
Internet-Draft
Intended status: Informational
Expires: 4 January 2026

M. Levy
SPIRL
H. Singhal
GitHub

3 July 2025

WIMSE Headless JWT Authentication and Authorization
draft-levy-wimse-headless-jwt-authentication-01

Abstract

In workload-to-service communication, a common pattern is for a workload to present a JSON Web Token (JWT) to a remote endpoint in order to obtain a temporary credential for the service it ultimately needs to access. It is a partial adaptation for workloads of existing flows designed for users. Implementing this pattern combines multiple existing standards from different working groups and standards bodies. Since this pattern is not described in a specification, it leads to variability in interoperability. The purpose of this document is to capture this common workload identity practice as an RFC in order to obtain consistency and promote interoperability in industry.

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://heymarcel.github.io/draft-ietf-wimse-headless-jwt-authentication/draft-levy-wimse-headless-jwt-authentication-practices.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-levy-wimse-headless-jwt-authentication/.

Discussion of this document takes place on the Workload Identity in Multi System Environments 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/heymarcel/draft-ietf-wimse-headless-jwt-authentication.

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 4 January 2026.

Copyright Notice

Table of Contents

1. Introduction

In workload-to-service communication, a common pattern is for a workload to use a JSON Web Token (JWT) to identify and authenticate itself as part of a process to obtain a temporary credential for the service it ultimately needs to access. This is done by having the workload present an asynchronously-provisioned bearer token in the form of a signed JWT to a remote endpoint that is associated with the target service. The remote endpoint verifies the JWT and then provides a temporary credential that the target service understands. The "bootstrap" problem of discovering the original JWT issuer is solved by requesting a JSON configuration document using the process described in OpenID Connect Discovery [OIDC.Discovery] or OAuth 2.0 Authorization Server Metadata [RFC8414].

Since this pattern is not described in a specification, it leads to variability in interoperability. The purpose of this document is to capture this common workload identity practice as an RFC in order to obtain consistency and promote interoperability in industry.

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.

2.1. Terminology

This document uses the following terms:

* Workload Platform

The underlying system which manages the deployment and scheduling of a Workload. This includes but is not limited to operating systems, orchestration services, and cloud providers.

* Tenant

A logically isolated entity within a Workload Platform that represents a distinct organizational or administrative boundary [OIPD]. A Workload Platform may have a single Tenant, or multiple Tenants.

* Exchange Service

A remote endpoint responsible for authenticating the identity of its callers, and subsequently issuing a temporary credential that is compatible with other services within the exchange service's domain. Examples of an exchange service include an OAuth Authorization Server and AWS Security Token Service.

* Target Service

The service that the workload ultimately wants to access. The target service accepts temporary credentials issued to the workload by the exchange service. Examples include an OAuth resource server or AWS S3.

3. Architecture and Message Flow

Figure 1 illustrates the OIDC-based message flow described in Section 4:

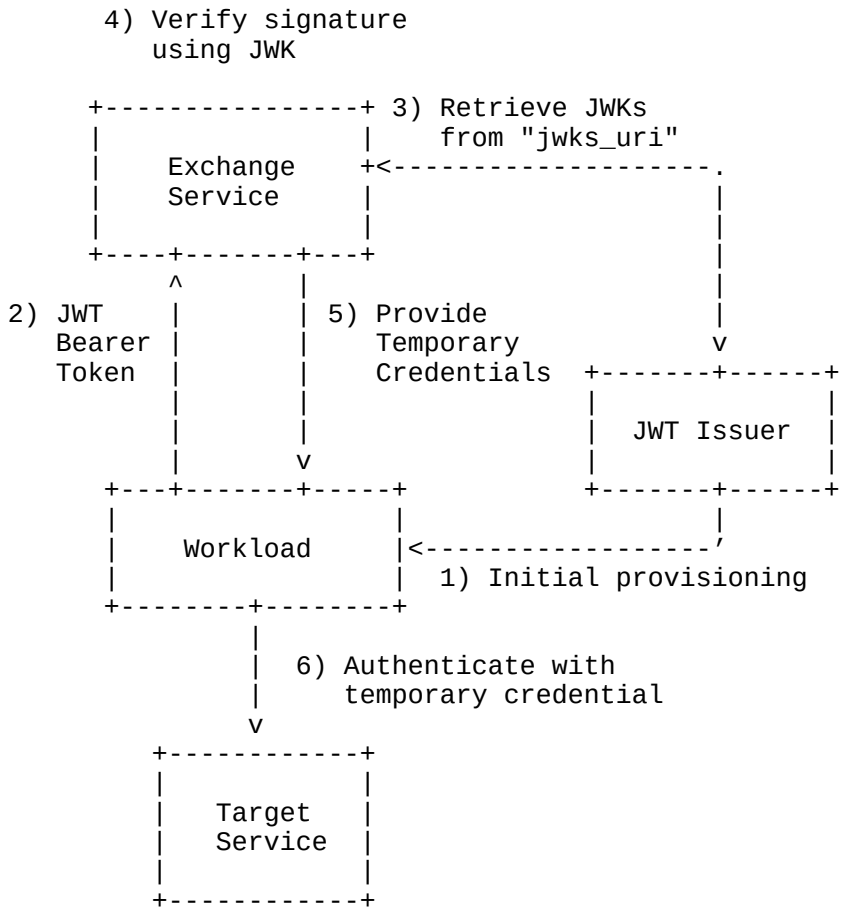


Figure 1: OIDC message flow when used in a headless environment

4. JWT used for Authentication

The overall message flow is seen in Figure 1, and this section explains it in more detail. It assumes the workload has previously acquired a JWT adhering to the profile specified in [RFC7523]. JWT provisioning assumptions are described in more detail in Section 6.3.

1. The workload calls a remote Exchange Service that is associated with the target service and presents a JWT Bearer Token as specified in Section 4 of [RFC7523].

2. The Exchange Service takes the value from the iss claim and appends /.well-known/openid-configuration to retrieve the JWT Issuer's configuration via HTTP, as specified in [OIDC.Discovery]. Alternatively, the OAuth 2.0 Authorization Server Metadata endpoint [RFC8414] may be used.

3. The Exchange Service then retrieves the JWKs via HTTP from the jwks_uri declared in the JWT Issuer's configuration response.

4. Using the appropiate issuer key, the Exchange Service verifies the signature of the JWT Bearer Token, and validates that the workload is authorized to receive a temporary credential in its domain.

5. Assuming successful verification, the Exchange Service then responds to the workload with a temporary credential suitable for use with the target service.

6. The Workload then authenticates with the target service using the temporary credential.

This document limits discussion to HTTP, as this is the protocol predominantly used. Although other protocols are out of scope, this should not be read as a limit on their future use.

5. Key Discovery

Issuer key discovery follows the steps outlined in Section 4 of [OIDC.Discovery]. The Exchange Service makes a request to a location that is well-known according to [RFC5785]:

```
GET /.well-known/openid-configuration HTTP/1.1
Host: example.com
```

Figure 2: Example request to issuer to obtain OIDC configuration

For OAuth 2.0, the equivalent location is /.well-known/oauth-authorization-server. In both cases, the requester expects a JSON document containing configuration information. An example is provided here:

```
{
  "issuer": "https://example.com",
  "jwks_uri": "https://example.com/.well-known/jwks.json",
  "authorization_endpoint": "https://example.com/auth",
  "token_endpoint": "https://example.com/token"
}
```

Figure 3: Example issuer configuration response

For the sake of the pattern described in this document, only the issuer and jwks_uri fields are relevant.

Note that this key discovery mechanism does not address the question of whether the key itself should be trusted. This is a registration problem, and is discussed further in Section 8.

6. JWT Format and Processing Requirements

6.1. JWT Format

An example JWT adhering to [RFC7523] is seen in Figure 4. Although the example uses a WIMSE workload identifier ([I-D.ietf-wimse-s2s-protocol]) in the subject ("sub") claim, this is not a requirement.

```
{
  "iss": "https://issuer.example.org",
  "sub": "spiffe://example.org/ns/default/sa/customer-router-agent",
  "aud": "https://auth.example.com/token",
  "jti": "jwt-grant-id-x1y2z3a4",
  "exp": 1744845092,
  "iat": 1744841036
}
```

Figure 4: Example RFC7523 JWT

6.2. JWT Processsing

6.2.1. Exchange Service Processing

The Exchange Service validates the JWT according to Section 3 in [RFC7523], with the following exceptions:

1. The "sub" (subject) claim does not identify either a resource owner or an anonymous user.

2. The "sub" claim need not correspond to the "client id" of an OAuth client.

The Exchange Service validates the signature using the key discovered by the process described in Section 5.

6.2.2. Workload Processing

The workload is considered the client in this interaction. It can treat the JWT acquired during provisioning as an opaque token. It must handle any error reponse from the exchange service as per Section 3.2 in [RFC7523].

6.3. JWT Provisioning

The workload is provisioned with a JWT from a trusted source. This can be the underlying Workload Platform, or a separate issuing system. Regardless of the actual mechanism, JWT provisioning relies on a registration mechanism that establishes mutually-trusted, secure connections between the workload and the JWT provisioner.

This provisioning mechanism illustrates a key difference from flows defined in [RFC6749] and [OIDC.Core], in that there are no client credentials or other shared secrets required to bootstrap the flow.

7. Trust Relationships

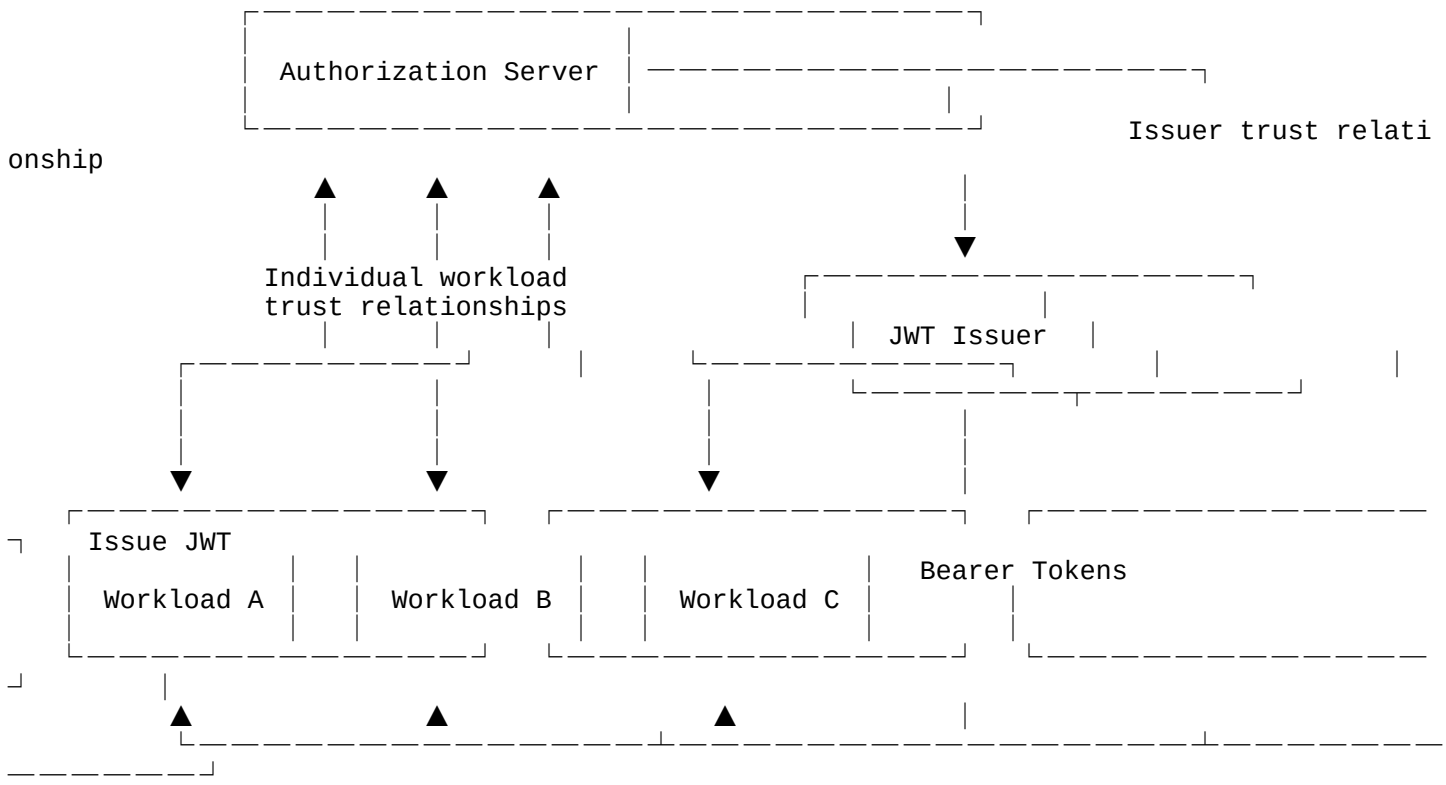
For functional headless JWT authentication, trust must be established at multiple levels for the authentication flow to function. For an authorization server to issue an access token to a workload, two distinct trust relationships must exist:

1. The authorization server MUST trust the JWT issuer.

2. The authorization server MUST trust the specific workload based on claims within the JWT bearer token.

These two trust relationships serve different purposes and SHOULD be managed independently as outlined below.

An example of the differences between issuer and workload trust relationships are three workload instances (A, B and C) that are all presenting JWT Bearer Tokens issued from the same JWT Issuer. This means that they build upon the trust relationship between the JWT issuer and authorization server. One workload instance has a specific workload trust relationship with the authorization server based on its subject identifier (the sub claim). It requires a very specific identifier which needs to match exactly. Another one makes use of a hierarchy within the subject identifier and the last one uses a combination of subject, audience and a custom claim as a basis for the trust relationship.



7.1. Issuer Trust Relationship

This trust relationship is between the Authorization Server and the JWT Issuer only. It represents the foundational level of trust and determines if the JWT Bearer Token a workload presents is accepted.

How this relationship is established is out of scope for this document. A possible approach for this implementation is maintaining a list of OAuth Authorization Server Metadata endpoints which instruct the authorization server to trust a specific issuer including the discovery of valid keys for that issuer via jwks.

This trust is typically established at a global or tenant-wide level, is subject to organizational policy and governance controls. Changes to issuer trust affect all workloads associated with that issuer simultaneously.

7.2. Workload Trust Relationship

Workload trust establishment builds upon issuer trust and focuses specifically on the relationship between the authorization server and individual workloads. Once the Bearer JWT token is authenticated the authorization server must determine if the specific workload identified in the JWT claims should be authorized.

The specifics of the claims are described in Section 6.1.

This trust is typically established individually and subject to different policy and governance controls.

8. Interoperability Considerations

In order for the workload to access the target service,

1. The JWT Issuer must be recognized by the Exchange Service,

2. Claims in the JWT are inspected and used to determine the subject, or principal, of the temporary credential issued to access the target service,

3. And the resulting temporary credential must be authorized to access the target service.

Step #1 requires the prior configuration of an explicit trust relationship between the Exchange Service and the JWT Issuer, and depends on vendor-specific configuration. Dynamic client registration standards ([RFC7591] and [OIDC.Dynamic]) explicitly place it out of scope.

Step #2 is a processing rule that is also previously-configured in an implementation-dependent manner. As an example of current practice for configuration of Steps #2 and #3, see [GitHub].

9. Security Considerations

This document illustrates a common pattern in trust domain federation. However, the "identity exchange" Step #2 in Section 8 is not standardized. In practice, the Workload Platform and the Resource Server platform define principals differently, and the translation mechanism between the two identities is implemented differently by each Resource Server platform. This lack of standardization is not merely inconvenient; it is a rich source of privilege escalation attacks. This is particularly true when both the Workload Platform and the Resource Server platform are multi-tenanted.

The following recommendations apply to configurations that control the "identity exchange" step that controls the translation of the workload JWT to a Resource Server identity:

1. When a Workload Platform contains multiple Tenants, the configuration SHOULD rely on a JWT issuing key bound to a single Tenant of the workload platform, rather than a single JWT issuing key for the Workload Platform.

2. The configuration SHOULD use specific JWT claims to prevent any JWT signed by the JWT Issuer from being used to impersonate any Resource Server principal.

3. When a Workload Platform contains multiple Tenants, the configuration SHOULD NOT solely rely on JWT claims that can be controlled by any Tenant. The configuration MAY rely on a "tenant" claim, if the claim value is issuer-controlled and corresponds to a single Tenant.

4. The configuration SHOULD NOT permit the transcription of JWT claims to the Resource Server principal without performing additional validation.

The security considerations in section 8 of [RFC7521] generally apply. As bearer tokens, stolen JWTs are particularly valuable to attackers:

1. A secure channel (e.g. TLS) MUST be used when providing a JWT for authentication.

2. JWTs SHOULD be protected from unauthorized access using operating system or platform access controls.

3. JWT validity SHOULD be set to the shortest possible duration allowable by overall system availability constraints.

10. IANA Considerations

This document has no IANA actions.

11. Examples

This section documents examples of the JWT headless authentication and authorization pattern in use with various target service types.

11.1. OAuth Resource Server

To use headless JWT authentication and authorization with a protected resource, workloads use the following steps to obtain a suitable access token:

1. The workload calls an authorization server's token endpoint and presents a JWT bearer token as specified in Section 4 of [RFC7523].

2. The authorization server verifies the signature of the JWT bearer token following the procedure specified in this document, and validates that the workload is authorized to receive an access token.

3. Assuming successful verification, the authorization server then responds to the workload with an access token suitable for use with the resource server.

11.2. AWS Service (e.g. S3)

To use headless JWT authentication and authorization with an AWS service such as S3, workloads use the following steps to obtain a suitable access token:

1. The workload makes an AssumeRoleWithWebIdentity call to the AWS STS service and presents a JWT bearer token in addition to the ARN of the role that the workload wishes to assume.

2. AWS STS verifies the signature of the JWT bearer token following the procedure specified in this document, and validates that the workload is authorized to assume the requested role.

3. Assuming successful verification, AWS STS then responds to the workload with temporary credentials, including a secret access key, for use with any further AWS service.

12. References

12.1. Normative References

[OIDC.Core]
Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0 incorporating errata set 2", 2023, <https://openid.net/specs/openid-connect-core-1_0.html>.

[OIDC.Discovery]
Sakimura, N., Bradley, J., Jones, M., and E. Jay, "OpenID Connect Discovery 1.0 incorporating errata set 2", 2023, <https://openid.net/specs/openid-connect-discovery-1_0.html>.

[OIDC.Dynamic]
Sakimura, N., Bradley, J., and M. Jones, "OpenID Connect Dynamic Client Registration 1.0 incorporating errata set 2", 2023, <https://openid.net/specs/openid-connect-registration-1_0.html>.

[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>.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <https://www.rfc-editor.org/rfc/rfc5785>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/rfc/rfc6749>.

[RFC7521] Campbell, B., Mortimore, C., Jones, M., and Y. Goland, "Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants", RFC 7521, DOI 10.17487/RFC7521, May 2015, <https://www.rfc-editor.org/rfc/rfc7521>.

[RFC7523] Jones, M., Campbell, B., and C. Mortimore, "JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants", RFC 7523, DOI 10.17487/RFC7523, May 2015, <https://www.rfc-editor.org/rfc/rfc7523>.

[RFC7591] Richer, J., Ed., Jones, M., Bradley, J., Machulak, M., and P. Hunt, "OAuth 2.0 Dynamic Client Registration Protocol", RFC 7591, DOI 10.17487/RFC7591, July 2015, <https://www.rfc-editor.org/rfc/rfc7591>.

[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>.

12.2. Informative References

[GitHub] "About security hardening with OpenID Connect", 2025, <https://docs.github.com/en/actions/security-for-github-actions/security-hardening-your-deployments/about-security-hardening-with-openid-connect>.

[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-04, 2 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-wimse-arch-04>.

[I-D.ietf-wimse-s2s-protocol] Campbell, B., Salowey, J., Schwenkschuster, A., and Y. Sheffer, "WIMSE Workload to Workload Authentication", Work in Progress, Internet-Draft, draft-ietf-wimse-s2s-protocol-05, 19 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-wimse-s2s-protocol-05>.

[OIPD] "OpenID Provider Commands 1.0", 2025, <https://openid.net/specs/openid-provider-commands-1_0.html>.

Acknowledgments

The authors would like to thank the following people for contributions to this document: Evan Gilman, Pieter Kasselman, Darin McAdams, and Arndt Schwenkschuster.

Authors' Addresses

Marcel Levy
SPIRL
Email: heymarcel@gmail.com

Hirsch Singhal
GitHub
Email: hpsin@github.com