Web Authorization Protocol
Internet-Draft
Intended status: Standards Track
Expires: 3 September 2026

A. Schwenkschuster
P. Kasselmann
Defakto Security
S. Rose
NIST
S. Thorgersen
IBM

2 March 2026

OAuth SPIFFE Client Authentication
draft-ietf-oauth-spiffe-client-auth-01

Abstract

This specification profiles the Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants [RFC7521], the JWT Profile for OAuth 2.0 Client Authentication and Authorization Grants [RFC7523], and OAuth 2.0 Attestation-Based Client Authentication [I-D.draft-ietf-oauth-attestation-based-client-auth] to enable the use of SPIFFE Verifiable Identity Documents (SVIDs) as client credentials in OAuth 2.0. It defines how OAuth clients with SPIFFE credentials can authenticate to OAuth authorization servers using their JWT-SVIDs, WIT-SVIDs, or X.509-SVIDs without the need for client secrets. This approach enhances security by enabling seamless integration between SPIFFE-enabled workloads and OAuth authorization servers while eliminating the need to distribute and manage shared secrets such as static client secrets.

About This Document

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

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-oauth-spiffe-client-auth/.

Discussion of this document takes place on the Web Authorization Protocol Working Group mailing list (mailto:oauth@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/oauth/. Subscribe at https://www.ietf.org/mailman/listinfo/oauth/.

Source for this draft and an issue tracker can be found at https://github.com/arndt-s/oauth-spiffe-client-authentication.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Traditional OAuth client authentication typically relies on client secrets or private key JWT authentication, both require an out of band distribution of secret material to the OAuth client. In modern cloud-native architectures where identity is managed by SPIFFE (Secure Production Identity Framework for Everyone), there is a need to provision additional secret material for OAuth clients when attested identifiers and credentials such as SVIDs are already available.

This specification profiles the Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants [RFC7521] to allow SPIFFE-enabled workloads to use their SPIFFE Verifiable Identity Documents (SVIDs), either X.509 certificates or JSON Web Tokens (JWT-SVID & WIT-SVID), as client credentials for OAuth 2.0 client authentication. JWT-SVIDs make use of a profiled version of the JWT Profile for OAuth 2.0 Client Authentication and Authorization Grants [RFC7523]. WIT-SVIDs make use of the OAuth 2.0 Attestation-Based Client Authentication [I-D.draft-ietf-oauth-attestation-based-client-auth].

This profile focuses on using SPIFFE credentials for OAuth client authentication.

The SPIFFE profile for client authentication enables seamless integration between SPIFFE-based and OAuth-based systems, allowing applications to leverage both ecosystems without requiring additional credential management. It also enables a more secure authentication method by leveraging cryptographically verifiable credentials rather than shared secrets.

2. Conventions and Definitions

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 specification uses the terms defined in OAuth 2.0 [RFC6749], the Assertion Framework for OAuth 2.0 [RFC7521], the JWT profile of it [RFC7523], and the SPIFFE specifications. In particular, the following terms are particularly relevant:

Trust Domain: As defined in SPIFFE; A trust domain represents a single trust root. All SVIDs issued within a trust domain are verifiable via the trust domain’s keys.

SPIFFE ID: A unified resource identifier that uniquely and specifically identifies a workload using the spiffe scheme. See [SPIFFE_ID] for details.

SVID: A SPIFFE Verifiable Identity Document. This document specifies the use of two types of SVIDs:

* *X.509-SVID*: An X.509 certificate that contains a SPIFFE ID in the URI SAN extension. See [SPIFFE_X509] for details.

* *JWT-SVID*: A JSON Web Token (JWT) that contains a SPIFFE ID in the sub claim. See [SPIFFE_JWT] for details.

SPIFFE Bundle: A collection of public keys and associated metadata that allow validation of SVIDs issued by a trust domain.

SPIFFE Bundle Endpoint: A URL that serves a SPIFFE bundle for a trust domain.

3. OAuth Client Authentication Using SPIFFE

This section describes how SPIFFE identity documents can be used for OAuth 2.0 client authentication, following the patterns established in [RFC7521] and, in case of JWT-SVID [RFC7523].

OAuth 2.0 client authentication is used to authenticate the client to the authorization server when making requests to the token endpoint. When using SPIFFE for client authentication, the client presents its SVID (JWT-SVID, WIT-SVID, or X.509-SVID) to prove its identity.

3.1. Client Authentication with JWT-SVIDs

JWT-SVID based authentication naturally follows the JWT Profile for OAuth 2.0 Client Authentication [RFC7523], with specific adaptations for SPIFFE JWT-SVIDs. [RFC7521] remains valid.

To identify the assertion content as a JWT-SVID this specification establishes the following client assertion type as an OAuth URI according to [RFC6755]:

urn:ietf:params:oauth:client-assertion-type:jwt-spiffe

Based on [RFC7523] the following request parameters MUST be present to perform client authentication in the context of this specification:

* client_assertion_type: MUST be set to urn:ietf:params:oauth:client-assertion-type:jwt-spiffe.

* client_assertion: MUST be a single SPIFFE JWT-SVID.

To validate JWT-SVID client authentication requests the authorization server MUST:

1. Verify that the JWT is well-formed and contains all required claims (SPIFFE ID in sub, aud, and exp).

2. Verify that the JWT has not expired (check the exp claim).

3. Verify that the aud claim contains only the issuer identifier of the authorization server as its sole value. See [I-D.draft-ietf-oauth-rfc7523bis] for details.

4. Verify the JWT signature using the signing keys of the trust domains according to Section 6.

5. Verify that the SPIFFE ID in the sub claim matches or is associated with a recognized client identifier. If the client authentication is presented with a client_id that is a URL described in [I-D.ietf-oauth-client-id-metadata-document], verify that the SPIFFE ID in the sub claim matches the spiffe_id value in the Client ID Metadata Document as described in Section 5.1.

3.1.1. JWT-SVID example

The following examples illustrates an authorization_code request to the token endpoint of an OAuth 2.0 authorization server leveraging a SPIFFE JWT-SVID to authenticate the client.

```
POST /token HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&
code=n0esc3NRze7LTCu7iYzS6a5acc3f0ogp4&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3A
client-assertion-type=jwt-spiffe&
client_assertion=eyJhbGciOiJFUzI1NiIsImtpZCI6IjR2QzhhZ3ljSHU2cm5rRUVKWUFINlZ1Q2U0Sm9Ta1BW
IiwidHlwIjoiSldUIn0.eyJhdWQiOlsiaHR0cHM6Ly9hcy5leGFtcGxlLmNvbS8iXSwiZXhwIjoxNzQ3MTI0NTQzL
CJpYXQiOjE3NDcxMjQyNDMsInN1YiI6InNwaWZmZTovL2V4YW1wbGUub3JnL215LW9hdXRoLWNsaWVudCJ9.Xlv5l
W4cbxDsQk4l0paewG4nXOR7MxF_FMn_c27DX45Bxr2HUZf9a6Untfq5S47xpwbw495HBL6_1Lc6TMJxmw
```

For clarify, the SPIFFE-JWT header and body decoded:

```
{
  "alg": "ES256",
  "kid": "4vC8agycHu6rnkEEJYAH6VuCe4JoSkPV",
  "typ": "JWT"
}.
{
  "aud": [
    "https://as.example.com/"
  ],
  "exp": 1747124543,
  "iat": 1747124243,
  "sub": "spiffe://example.org/my-oauth-client"
}
```

3.1.2. JWT-SVID example with Client ID Metadata Document

The following example illustrates a client_credentials request to the token endpoint of an OAuth 2.0 authorization server leveraging a SPIFFE JWT-SVID to authenticate the client.

```
POST /token HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=client_credentials&
client_id=https%3A%2F%2Fexample.org%2Fclient%2Fmetadata.json&
client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3A
client-assertion-type=jwt-spiffe&
client_assertion=eyJhbGciOiJFUzI1NiIsImtpZCI6IjR2QzhhZ3ljSHU2cm5rRUVKWUFINlZ1Q2U0Sm9Ta1BW
IiwidHlwIjoiSldUIn0.eyJhdWQiOlsiaHR0cHM6Ly9hcy5leGFtcGxlLmNvbS8iXSwiZXhwIjoxNzQ3MTI0NTQzL
CJpYXQiOjE3NDcxMjQyNDMsInN1YiI6InNwaWZmZTovL2V4YW1wbGUub3JnL2NsaWVudC8xMjM0In0.Xlv5lW4cbx
DsQk4l0paewG4nXOR7MxF_FMn_c27DX45Bxr2HUZf9a6Untfq5S47xpwbw495HBL6_1Lc6TMJxmw
```

For clarity, the SPIFFE-JWT header and body decoded:

```
{
  "alg": "ES256",
  "kid": "4vC8agycHu6rnkEEJYAH6VuCe4JoSkPV",
  "typ": "JWT"
}.
{
  "aud": [
    "https://as.example.com/"
  ],
  "exp": 1747124543,
  "iat": 1747124243,
  "sub": "spiffe://example.org/client/1234"
}
```

The client_id points to a Client ID Metadata Document ([I-D.ietf-oauth-client-id-metadata-document]) with the following contents:

```
{
    "client_id": "https://example.org/client/metadata.json",
    "client_name": "Example Client",
    "spiffe_id": "spiffe://example.org/client/*",
    "spiffe_bundle_endpoint": "https://example.org/client/bundle.json"
}
```

3.2. Client Authentication using X509-SVID

X.509-SVID based authentication uses mutual TLS as defined in OAuth 2.0 Mutual-TLS Client Authentication [RFC8705], with specific adaptations for SPIFFE X.509-SVIDs.

To authenticate using an X.509-SVID, the client establishes a mutual TLS connection with the authorization server using its X.509-SVID as the client certificate. The authorization server validates the client certificate as an X.509-SVID and extracts the SPIFFE ID from the URI SAN. The server certificate MUST be validated by the client using its system trust store, and NOT the SPIFFE trust bundle.

The request MUST include the client_id parameter containing the SPIFFE-ID of the client. It MUST match the URI SAN of the presented X509-SVID client credential.

The server validates the client certificates according the following rules

1. Perform standard X.509 path validation against the trust anchors according to Section 6.

2. Verify that the certificate contains exactly one URI SAN with a valid SPIFFE ID.

3. Verify that the certificate is a leaf certificate (Basic Constraints extension has CA=FALSE).

4. Verify that the certificate has the digitalSignature key usage bit set.

5. Verify that the SPIFFE ID in the URI SAN matches a registered client identifier or is associated with a registered client identifier.

3.2.1. X509-SVID Example

The following request uses a refresh token to obtain a new access token. The client is spiffe://example.org/my-oauth-client and is authenticated by performing this request over a mutual TLS connection.

```
POST /token HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=refresh_token&
refresh_token=tGzv3JOkF0XG5Qx2TlKWIA&
client_id=spiffe://example.org/my-oauth-client
```

For clarity, the presented X509-SVID client certificate to the server decoded via openssl x509 -text is:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            dd:48:ec:d4:a4:c6:b2:ea:8e:9b:54:35:e8:30:65:7b
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C=US, O=SPIFFE, serialNumber=69687291928591476146956383703880290008
        Validity
            Not Before: May 16 11:26:11 2025 GMT
            Not After : May 16 12:26:21 2025 GMT
        Subject: C=US, O=SPIRE
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:c2:0b:b6:8e:47:9a:20:ab:33:f1:a9:a5:77:97:
                    fa:a0:95:7d:2c:9f:e9:94:3d:e9:ed:e6:35:52:7f:
                    ff:82:34:74:20:97:5a:1b:4e:87:5f:32:3e:9d:da:
                    60:6a:05:8b:86:9d:0b:59:5f:67:be:93:3b:26:de:
                    ea:1e:18:98:96
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment, Key Agreement
            X509v3 Extended Key Usage:
                TLS Web Server Authentication, TLS Web Client Authentication
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Subject Key Identifier:
                D8:7A:2F:8B:E3:CF:08:83:EA:DD:5E:0A:59:33:6E:4C:E0:CC:6B:AD
            X509v3 Authority Key Identifier:
                C2:41:49:B0:ED:E0:94:7B:FA:7D:C2:F1:02:24:20:B9:1E:3D:56:FA
            X509v3 Subject Alternative Name:
                URI:spiffe://example.org/my-oauth-client
    Signature Algorithm: ecdsa-with-SHA256
    Signature Value:
        30:44:02:20:48:c3:5f:68:b2:c5:5d:96:c4:96:32:37:1f:af:
        b8:1c:1c:45:ad:41:26:dd:e2:92:b5:73:62:83:34:c6:16:2a:
        02:20:0f:48:02:8e:6b:1d:09:01:80:d8:85:2b:ca:25:c6:2c:
        9e:f2:27:c2:3c:e4:03:58:a8:47:21:f6:3c:5e:7a:c8
```

3.3. Client Authentication with WIT-SVIDs

WIT-SVIDs are the SPIFFE variant of the WIMSE Workload Identity Token (WIT) as defined in [I-D.draft-ietf-wimse-workload-creds] and make use of concepts defined in OAuth 2.0 Attestation-Based Client Authentication [I-D.draft-ietf-oauth-attestation-based-client-auth].

A WIT-SVID as issued by a SPIFFE implementation binds a key held by the client via the cnf claim. This key is used as attestation proof during client authentication. The attestation proof is in the form of a "Client Attestation PoP JWT" as defined in [I-D.draft-ietf-oauth-attestation-based-client-auth] that is issued by the client and signed with the private part of the key bound in the WIT-SVID.

The WIT-SVID and the corresponding Client Attestation PoP JWT are sent together to the authorization server as a means of client authentication using the HTTP header-based syntax defined in Section 6.1 of [I-D.draft-ietf-oauth-attestation-based-client-auth].

3.3.1. Authorization Server Validation

To validate a WIT-SVID client authentication request, the authorization server MUST apply the following rules:

* The authorization server MUST accept wit+jwt as a valid value of the typ JWT header parameter in the OAuth-Client-Attestation header, in addition to oauth-client-attestation+jwt.

* The WIT-SVID MUST be validated as a Workload Identity Token according to Section 3.1 of [I-D.draft-ietf-wimse-workload-creds].

* The WIT-SVID signature MUST be verified using the signing keys of the trust domain according to Section 6.

* The Client Attestation PoP JWT MUST be validated according to Section 5.2 of [I-D.draft-ietf-oauth-attestation-based-client-auth].

  TODO: What to do with attest_jwt_client_auth method in AS metadata?

3.3.2. WIT-SVID Example

The following example illustrates a token request using a WIT-SVID and a Client Attestation PoP JWT for client authentication.

The WIT-SVID (OAuth-Client-Attestation) header and body decoded:

```
{
  "typ": "wit+jwt",
  "alg": "ES256",
  "kid": "4vC8agycHu6rnkEEJYAH6VuCe4JoSkPV"
}.
{
  "iss": "spiffe://example.org/my-spiffe-workload-api",
  "sub": "spiffe://example.org/my-oauth-client",
  "exp": 1747128000,
  "iat": 1747124400,
  "cnf": {
    "jwk": {
      "kty": "EC",
      "crv": "P-256",
      "x": "18wHLeIgW9wVN6VD1Txgpqy2LszYkMf6J8njVAibvhM",
      "y": "-V4dS4UaLMgP_4fY4j8ir7cl1TXlFdAgcx55o7TkcSA"
    }
  }
}
```

The Client Attestation PoP JWT (OAuth-Client-Attestation-PoP) header and body decoded:

```
{
  "typ": "oauth-client-attestation-pop+jwt",
  "alg": "ES256"
}.
{
  "iss": "spiffe://example.org/my-oauth-client",
  "aud": "https://as.example.com",
  "jti": "d25d00ab-552b-46fc-ae19-98f440f25064",
  "iat": 1747124400
}
```

The token endpoint request:

```
POST /token HTTP/1.1
Host: as.example.com
Content-Type: application/x-www-form-urlencoded
OAuth-Client-Attestation: eyJ0eXAiOiJ3aXQrand0IiwiYWxnIjoiRVMyNTYiL...
OAuth-Client-Attestation-PoP: eyJhbGciOiJFUzI1NiIsInR5cCI6Im9hdXRoLWN...

grant_type=authorization_code&
code=n0esc3NRze7LTCu7iYzS6a5acc3f0ogp4&
client_id=spiffe://example.org/my-oauth-client
```

4. Interoperability

In order to achieve interoperability between the authorization server and clients the authorization server MUST advertise what client authentication methods are supported.

Authorization servers MUST support at least one of JWT-SVID or X509-SVID. The methods supported MUST be advertised in the authorization servers metadata [RFC8414] by including spiffe_jwt, spiffe_wit and/or spiffe_x509 in the token_endpoint_auth_methods_supported list. Additionally, the same should be included in revocation_endpoint_auth_methods_supported and introspection_endpoint_auth_methods_supported when applicable.

Clients MUST support at least one of JWT-SVID, WIT-SVID or X509-SVID. To guarantee interoperability a client SHOULD support all.

It is the responsibility of the client to select the authentication method supported by the authorization server and its deployment.

5. SPIFFE Trust Establishment and Client Registration

This specification requires previously established trust between the OAuth 2.0 Authorization Server and the SPIFFE Trust Domain. This needs to happen out of band and is not in scope of this specification. However, the mechanisms of key distribution is in scope and described in Section 6.

Similar to the trust establishment, corresponding OAuth clients need to be established prior of using SPIFFE as client authentication. This is also out of scope, implementors may for example choose to leverage OAuth 2.0 dynamic client registration according to [RFC7591] or configure them out of band.

5.1. Client Registration Metadata

This specification defines the following client metadata parameters for use in Client ID Metadata Documents [I-D.ietf-oauth-client-id-metadata-document]:

spiffe_id REQUIRED. The SPIFFE ID of the client, e.g. spiffe://example.org/my-oauth-client. The value MAY include a trailing /* character sequence (e.g. spiffe://example.org/workloads/*) indicating that a prefix match against the SPIFFE ID in the sub claim of the JWT-SVID is acceptable rather than requiring an exact match.

spiffe_bundle_endpoint OPTIONAL. The URL of the SPIFFE Bundle

Endpoint for the client's trust domain, which the authorization server can use to retrieve the signing keys for validating the client's SVIDs. If not provided, the authorization server MUST obtain the signing keys for the client's trust domain through some other established mechanism, such as a pre-configured SPIFFE bundle.

If the spiffe_id value uses a wildcard, it MUST end with the two-character sequence "/*". In this case, the authorization server MUST perform a path-segment prefix match: the sub claim value MUST begin with the spiffe_id value excluding the trailing "*" (i.e., up to and including the final "/"), and this prefix MUST correspond to complete path segment(s) of the SPIFFE ID (for example, spiffe://example.org/client/* matches spiffe://example.org/client/123 but does not match spiffe://example.org/client123). Otherwise, the sub claim MUST be an exact match of the spiffe_id value.

6. SPIFFE Key Distribution and Validation

This section describes how an authorization server verifies the signature of an X509 or JWT-SVID. It recommends two SPIFFE-native approaches.

Trust bundles in general MUST be keyed by the trust domain identifier to prevent mix up between trust domain and their corresponding bundles. The 2 approaches can be used in conjunction, for instance:

```
Trust domain "example.org": Workload API at unix:///var/secrets/spiffe/agent.sock
Trust domain "production": SPIFFE Bundle Endpoint at https://example.com/auth/spiffe/bund
le.json
```

6.1. SPIFFE Bundle Endpoint

The SPIFFE Bundle Endpoint exposes the signing keys for X509-SVIDs and JWT-SVIDs over HTTP via a JSON Web Key Set according to [RFC7517].

Server authentication on this endpoint is available in two flavors. For the sake of interoperability, in the context of this specification the WebPKI flavor MUST be used. This effectively means that the server certificate of the bundle endpoint is trusted by the authorization server accessing it. See Sec 5.2.1 of [SPIFFE_FEDERATION] for details.

The authorization server SHOULD periodically poll the bundle endpoint to retrieve updated trust bundles, following the refresh hint and period provided in the bundle. See [SPIFFE_FEDERATION] for details.

The SPIFFE bundle endpoint cannot be derived from the JWT-SVID and X509-SVID and MUST be configured manually out of band. Bundle endpoints MUST be keyed by the trust domain identifier.

6.1.1. Example

The following examples showcase how the Authorization Server can perform key discovery for the trust domain example.org. Important to note is the difference between example.org trust domain and example.com location for the SPIFFE Bundle Endpoint. This highlights the importance of explicit configuration and undermines the fact that the SPIFFE Bundle Endpoint cannot be derived or discovered from the X509-SVID without explicit configuration.

Example configuration at the OAuth Authorization Server in the JSON format

```
{
  "example.org": {
    "spiffe_bundle_endpoint": {
      "url": "https://example.com/bundle.json"
    }
  }
}
```

Note difference between example.org and example.com

Example SPIFFE Bundle Endpoint request, response:

```
GET /bundle.json HTTP/1.1
Host: example.com

{
  "keys": [
    {
      "use": "x509-svid",
      "kty": "EC",
      "crv": "P-384",
      "x": "9XBzty8W_ex4Xr0RdzUBgie_okdaUTheSF0PQvVAaTsXaP1J7yv0Dhlaw45I7Cv9",
      "y": "HP21HOmMxIlZ0XeqsOl9sM5H57HBQWu0bINXfw4jdeHdB5vk1XyNyBQQxeUpSxhn",
      "x5c": [
        "MIIB2DCCAV6gAwIBAgIURJ20yIzal3ZT9NXkdwrsm0selwwwCgYIKoZIzj0EAwQwHjELMAkGA1UEBhMC
VVMxDzANBgNVBAoMBlNQSUZGRTAeFw0yMzA1MTUwMjA1MDZaFw0yODA1MTMwMjA1MDZaMB4xCzAJBgNVBAYTAlVTM
Q8wDQYDVQQKDAZTUElGRkUwdjAQBgcqhkjOPQIBBgUrgQQAIgNiAAT1cHO3Lxb97HhevRF3NQGCJ7+iR1pROF5IXQ
9C9UBpOxdo/UnvK/QOGVrDjkjsK/0c/bUc6YzEiVnRd6qw6X2wzkfnscFBa7Rsg1d/DiN14d0Hm+TVfI3IFBDF5Sl
LGGejXTBbMB0GA1UdDgQWBBSSiuNgxqqnz2r/jRcWsARqphwQ/zAPBgNVHRMBAf8EBTADAQH/MA4GA1UdDwEB/wQE
AwIBBjAZBgNVHREEEjAQhg5zcGlmZmU6Ly9sb2NhbDAKBggqhkjOPQQDBANoADBlAjEA54Q8hfhEd4qVycwbLNzOm
/HQrp1n1+a2xc88iU036FMPancR1PLqgsODPfWyttdRAjAKIodUi4eYiMa9+I2rVbj8gOxJAFn0hLLEF3QDmXtGPp
ARs9qC+KbiklTu5Fpik2Q="
      ]
    },
    {
      "use": "jwt-svid",
      "kty": "EC",
      "kid": "6d02Vc2oU62mXVH5nlggHGLmfIhrlnNW",
      "crv": "P-256",
      "x": "S2V42XlFjNp30CFmOidbWQT9IpZHqJ8JuuJgDBvkdZA",
      "y": "vN0y5TK36VRxZo_E3Gc7S5c0jIRIaHZ53f2UiJ1NFto"
    },
    {
      "use": "wit-svid",
      "kty": "EC",
      "kid": "b7f3K2oPz8nQvT1xYl9gHGLmfIhrlnXY",
      "crv": "P-256",
      "x": "18wHLeIgW9wVN6VD1Txgpqy2LszYkMf6J8njVAibvhM",
      "y": "-V4dS4UaLMgP_4fY4j8ir7cl1TXlFdAgcx55o7TkcSA"
    }
  ],
  "spiffe_sequence": 10,
  "spiffe_refresh_hint": 300
}
```

The use parameter in the JSON Web Key indicates the credential format the key is indended for. Multiple keys of the same use can be present.

The X509-SVID signing certificate (.keys[0].x5c[0] from response above) in text form:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            5c:4b:d5:2d:f9:c1:6e:78:2c:32:a6:bb:6c:73:f0:b8:f4:be:13:09
        Signature Algorithm: ecdsa-with-SHA512
        Issuer: C=US, O=SPIFFE
        Validity
            Not Before: May 16 11:23:19 2025 GMT
            Not After : May 15 11:23:19 2030 GMT
        Subject: C=US, O=SPIFFE
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (384 bit)
                pub:
                    04:ef:3f:db:67:2b:e8:5c:a1:64:23:e7:f2:fd:f0:
                    3b:16:55:68:17:55:17:d4:bd:cd:6d:04:fd:cc:8f:
                    99:31:f7:8c:ac:b0:1e:31:60:18:45:32:8b:a1:17:
                    4b:2f:01:75:27:6c:3f:c3:a5:b9:da:56:fb:29:54:
                    63:cb:08:96:81:35:0e:96:04:03:40:fe:51:0d:26:
                    da:d5:99:6c:8f:c2:45:43:cb:2c:b4:8d:9b:68:78:
                    9f:c0:2d:68:36:b8:5e
                ASN1 OID: secp384r1
                NIST CURVE: P-384
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                8D:79:D2:26:5E:4C:83:30:40:C7:E9:1D:E1:35:12:F6:60:CF:0B:DB
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Subject Alternative Name:
                URI:spiffe://example.org
    Signature Algorithm: ecdsa-with-SHA512
    Signature Value:
        30:64:02:30:0a:e9:fd:d4:cd:99:52:90:cb:14:86:93:4e:f8:
        02:52:d6:17:12:9f:2e:65:99:0e:38:b6:b9:a6:fe:43:0f:60:
        30:04:87:ec:24:20:80:a4:75:ee:3c:ad:9d:a2:72:0d:02:30:
        55:93:0e:14:8c:47:47:3b:74:7c:a7:2a:2a:96:1d:a4:85:46:
        4f:3f:95:a4:c2:ab:3c:2e:04:b3:1b:cf:02:0f:33:fc:dd:dc:
        d5:2f:44:c8:2a:dc:ce:3f:c5:c6:89:d0
```

TODO: Bundle doesn’t match X509-SVID. This needs to be fixed.

6.2. Alternative methods to avoid

The following key distribution mechanisms are alternatives and SHOULD be avoided for interopability reasons.

6.2.1. SPIFFE Workload API

The SPIFFE Workload API allows workloads to retrieve a trust bundle. It requires the authorization server to be part of a SPIFFE trust domain and be considered a workload within it. The SPIFFE Workload API is build in a way that the workload proactively retrieves trust bundles updates and does not need to poll them, which reduces the time to distribute them. In addition to the trust bundle of the trust domain the workload resides in, the SPIFFE Workload API also allows to retrieve trust bundles from federated trust domains.

This approach is NOT RECOMMENDED for OAuth SPIFFE Client Authentication for several reasons:

* OAuth Authorization Server needs to be a workload within a SPIFFE trust domain, which is a significant limitation for deployment scenarios.

* Federated trust domain bundles create ambiguity about how they are handled. When distributed via the SPIFFE Workload API the trust relationship and points where they are established become ambiguous.

6.2.2. Manual configuration

In small, static environments the authorization server MAY be configured with the SPIFFE bundles manually. This approach requires human interaction to set up, rotate and manage keying material and is thus generally NOT RECOMMENDED.

6.2.3. Using the system trust store

X509-SVIDs MUST NOT be validated using the system trust store. The SPIFFE ID carried in the URI SAN is rarely a verifiable attribute in the broader X.509 ecosystem. Using the system trust store as trust anchor would allow ANY certificate authority in it to issue a trusted X509-SVID for ANY SPIFFE ID. In comparison: using SPIFFE-native validation methods restricts the signing of SPIFFE-IDs to the corresponding trust domain signing keys.

6.2.4. Using the JWT-SVID or WIT-SVID iss claim

JSON Web Token-based credentials carrying iss claims could technically be validated by retrieving the signing keys via OpenID Connect Discovery or OAuth 2.0 Authorization Server Metadata. This approach only applies for the JWT-SVID and WIT-SVID and only works when the iss claim is present, which is optional.

Because of its narrow scope and interoperability considerations, this approach is not a general alternative to the SPIFFE Bundle Endpoint. Implementations SHOULD use the mechanisms defined in this specification when available. However, when those mechanisms are not supported by a peer deployment, implementations MAY use iss-based discovery and key retrieval for JWT-SVID or WIT-SVID validation as a compatibility mechanism, subject to local policy, appropriate trust configuration and risk mitigations described in Section 8.1.

7. 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".

Keycloak

* Organization: CNCF / IBM
* Maturity: preview
* Coverage: JWT-SVID client authentication using SPIFFE Trust Bundle Endpoint
* Contact: Keycloak community & maintainers (https://www.keycloak.org/community)

8. Security Considerations

Client authentication using JWT-SVIDs has the same security considerations as described in [RFC6749] and [RFC7521].

Client authentication using X509-SVIDs has the same security considerations as described in [RFC8705]. The validation rules in section 3.2 protect against an OAuth2 token being issued (or being issued incorrectly) to a client that did not present an appropriate X509-SVID.

The issues described in Section 5.2 above include the threat that an authorization server may have the incorrect trust stores configured to validate the client SVID. This could result in an incorrectly issued token to an attacker if the attacker is able to obtain a certificate that can be validated by one of the misconfigured trust anchors in the trust store.

8.1. JWT-SVID and WIT-SVID iss claim

As described in Section 6.2.4, iss-based key discovery is not a general alternative to the SPIFFE Bundle Endpoint and SHOULD be avoided.

However, if it is used, the authorization server MUST NOT perform issuer-based discovery solely based on the iss value from the token unless the iss value is already known to the authorization server, and that iss value is explicitly associated with the configured SPIFFE Trust Domain being validated.

In addition implementations MUST NOT assume that keys advertised via OpenID Connect Discovery or OAuth 2.0 Authorization Server Metadata are dedicated to JWT-SVID issuance. The same provider infrastructure may issue multiple JWT types (e.g., OAuth/OIDC tokens, JWT-SVIDs, WIT-SVIDs). A token MUST NOT be accepted as a JWT-SVID or WIT-SVID solely because it is a JWT, its signature validates under keys discovered from iss, and its sub resembles a SPIFFE ID. Doing so can enable token confusion (e.g., presenting an OAuth/OIDC token issued for another purpose as a JWT-SVID). Implementations MUST validate SVIDs according to SVID-specific requirements and MUST use a trust anchor or key source explicitly bound to the configured SPIFFE Trust Domain, rather than generic issuer-based discovery from untrusted token input.

9. IANA Considerations

This document requests a new entry to be added to the Oauth URI registry found at https://www.iana.org/assignments/oauth-parameters/oauth-parameters.xhtml#uri (https://www.iana.org/assignments/oauth-parameters/oauth-parameters.xhtml#uri). The registration process is defined in [RFC6755]. This document requests the following entry to be added to the registry:

* URN: urn:ietf:params:oauth:client-assertion-type:jwt-spiffe
* Common Name: SPIFFE JWT-SVID Profile for OAuth 2.0 Client Authentication
* Change Controller: IETF
* Reference: This Document

9.1. OAuth Dynamic Client Registration Metadata

This document requests the following entries to be added to the "OAuth Dynamic Client Registration Metadata" registry defined in [RFC7591]:

* Client Metadata Name: spiffe_id
* Client Metadata Description: SPIFFE ID of the client
* Change Controller: IETF
* Reference: This Document
* Client Metadata Name: spiffe_bundle_endpoint
* Client Metadata Description: URL of the SPIFFE Bundle Endpoint for the client’s trust domain
* Change Controller: IETF
* Reference: This Document

10. References

10.1. Normative References

[I-D.draft-ietf-oauth-rfc7523bis]
Jones, M. B., Campbell, B., Mortimore, C., and F. Skokan, "Updates to OAuth 2.0 JSON Web Token (JWT) Client

Authentication and Assertion-Based Authorization Grants", Work in Progress, Internet-Draft, draft-ietf-oauth-rfc7523bis-05, 12 January 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-rfc7523bis-05>.

[I-D.ietf-oauth-client-id-metadata-document] Parecki, A. and E. Smith, "OAuth Client ID Metadata Document", Work in Progress, Internet-Draft, draft-ietf-oauth-client-id-metadata-document-01, 1 March 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-client-id-metadata-document-01>.

[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>.

[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>.

[RFC6755] Campbell, B. and H. Tschofenig, "An IETF URN Sub-Namespace for OAuth", RFC 6755, DOI 10.17487/RFC6755, October 2012, <https://www.rfc-editor.org/rfc/rfc6755>.

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

[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>.

[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>.

[RFC8705] Campbell, B., Bradley, J., Sakimura, N., and T. Lodderstedt, "OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens", RFC 8705, DOI 10.17487/RFC8705, February 2020, <https://www.rfc-editor.org/rfc/rfc8705>.

[SPIFFE_BUNDLE] "SPIFFE Bundle", n.d., <https://github.com/spiffe/spiffe/blob/main/standards/SPIFFE_Trust_Domain_and_Bundle.md#4-spiffe-bundle-format>.

[SPIFFE_FEDERATION] "SPIFFE Federation", n.d., <https://github.com/spiffe/spiffe/blob/main/standards/SPIFFE_Federation.md>.

[SPIFFE_ID] "SPIFFE-ID", n.d., <https://github.com/spiffe/spiffe/blob/main/standards/SPIFFE-ID.md>.

[SPIFFE_JWT] "JWT-SVID", n.d., <https://github.com/spiffe/spiffe/blob/main/standards/JWT-SVID.md>.

[SPIFFE_X509] "X509-SVID", n.d., <https://github.com/spiffe/spiffe/blob/main/standards/X509-SVID.md>.

10.2. Informative References

[I-D.draft-ietf-oauth-attestation-based-client-auth]
Looker, T., Bastian, P., and C. Bormann, "OAuth 2.0 Attestation-Based Client Authentication", Work in Progress, Internet-Draft, draft-ietf-oauth-attestation-based-client-auth-07, 15 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-attestation-based-client-auth-07>.

[I-D.draft-ietf-wimse-workload-creds]
Campbell, B., Salowey, J. A., Schwenkschuster, A., Sheffer, Y., and Y. Rosomakho, "WIMSE Workload Credentials", Work in Progress, Internet-Draft, draft-ietf-wimse-workload-creds-00, 3 November 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-wimse-workload-creds-00>.

Appendix A. Document History

// RFC Editor: please remove before publication.

A.1. draft-ietf-oauth-spiffe-client-auth-01

* Add mechanism to use Client ID Metadata Document for SPIFFE client metadata discovery.
* Add WIT-SVID variant using Workload Identity Tokens and Attestation-Based Client Authentication.
* Add interoperability section.
* Add more guidance and security considerations when using the iss claim.
* Add Stian Thorgersen as co-author.

A.2. draft-ietf-oauth-spiffe-client-auth-00

* Document name update to reflect adoption into the OAuth working group

A.3. draft-schwenkschuster-oauth-spiffe-client-auth-01

* Rephrase introduction to make the focus on client authentication more clear.
* Add implementation section.

* Add audience restrictions from RFC7523bis adopted WG document.

* Add security and IANA considerations section.

* Add Scott Rose as co-author.

A.4. draft-schwenkschuster-oauth-spiffe-client-auth-00

* Initial document

Acknowledgments

TODO acknowledge.

Authors' Addresses

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

Pieter Kasselmann
Defakto Security
Email: pieter@defakto.security

Scott Rose
NIST
Email: scott.rose@nist.gov

Stian Thorgersen
IBM
Email: sthorger@ibm.com