

Web Authorization Protocol
Internet-Draft
Intended status: Standards Track
Expires: 2 January 2026

A. Schwenkschuster
P. Kasselmann
SPIRL
1 July 2025

OAuth SPIFFE Client Authentication
draft-schwenkschuster-oauth-spiffe-client-auth-00

Abstract

This specification profiles the Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants [RFC7521] and JWT Profile for OAuth 2.0 Client Authentication and Authorization Grants [RFC7523] 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 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-schwenkschuster-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

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 2 January 2026.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	3
2. Conventions and Definitions	3
2.1. Terminology	4
3. OAuth Client Authentication Using SPIFFE	4
3.1. Client Authentication with JWT-SVIDs	4
3.1.1. JWT-SVID example	5
3.2. Client Authentication using X509-SVID	6
3.2.1. X509-SVID Example	7
4. SPIFFE Trust Establishment and Client Registration	9
5. SPIFFE Key Distribution and Validation	9
5.1. SPIFFE Bundle Endpoint	9
5.1.1. Example	10
5.2. Alternative methods to avoid	12
5.2.1. SPIFFE Workload API	13
5.2.2. Manual configuration	13
5.2.3. Using the system trust store	13
5.2.4. Using the JWT-SVID iss claim	13
6. Security Considerations	14
7. IANA Considerations	14
8. Normative References	14
Acknowledgments	15
Authors' Addresses	15

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 JWT tokens — as client credentials for OAuth 2.0 client authentication. JWT tokens make use of the profiled version of [RFC7523] - the JWT Profile for OAuth 2.0 Client Authentication and Authorization Grants [RFC7523].

This profile focuses specifically on client authentication rather than authorization grants. This focus is deliberate for several reasons:

1. In modern service-oriented architectures, services often need to authenticate as themselves to OAuth authorization servers.
2. Using SPIFFE as client authentication towards OAuth 2.0 authorization servers is a bridge between SPIFFE, which covers workload identity and OAuth, which covers human identity.
3. Using SPIFFE as authorization grants for authorization requests where the workload itself is the resource owner is covered by other specifications, such as [Headless_JWT].

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 (either JWT-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 equals the endpoint of the request without fragment and query parameters.
4. Verify the JWT signature using the signing keys of the trust domains according to Section 5.
5. Verify that the SPIFFE ID in the sub claim matches a registered client identifier or is associated with a registered client identifier.

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%3Ajwt-spiffe&
client_assertion=eyJhbGciOiJIUzI1NiIsImtpZCI6IjR2QzhhZ3l1jSHU2cm5rRUVKWUFINlZlQ2U0Sm9Ta1BW
IiwidHlwIjoisIldUIIn0.eyJhdWQiOiJlbnR0cHM6Ly9hcy5leGFtcGxlLmNvbS90b2t1biJdLCJleHAiOjE3NDcxM
jQ1NDMsImldCI6MTc0NzEyNDI0Mywic3ViIjoic3BpZmZlOi8vZXhhbXBsZS5vcmcvbXktb2F1dGgtY2xpZW50In
0.Xlv5lW4cbxDsQk4l0paewG4nXOR7MxF_FMn_c27DX45Bxr2HUZf9a6Untfq5S47xpwbw495HBL6_1Lc6TMJxmw
```

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

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

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 5.
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=6968729192859147614695638370388029008
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

4. 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 5.

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. 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/bundle.json

5.1. 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.

5.1.1. 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_okdaUTheSF0PQvVAaTsXaPlJ7yv0Dhlaw45I7Cv9",
      "y": "HP2lH0mMxIlZ0XeqsOl9sM5H57HBQWu0bINXfw4jdeHdB5vklXyNyBQQxeUpSxhn",
      "x5c": [
        "MIIB2DCCAV6gAwIBAgIUURJ20yIzal3ZT9NXkdwrsm0selwwwCgYIKoZIzj0EAwQwHjELMAkGA1UEBhMCVVMxZDZANBgNVBAoMBlNQSUZGRTAeFw0yMzA1MTUwMjA1MDZaFw0yODA1MTMwMjA1MDZaMB4xCzAJBgNVBAYTA1VTMQ8wDQYDVQQKDAZTUe1GRkUwdjAQBgcqhkJOPQIBBgUrgQQAIGNiAAT1cHO3Lxb97HhevRF3NQGCGJ7+iRlpROF5IXQ9C9UBpOxdo/UnvK/QOGVrDjkjsK/0c/bUc6YzEiVnRd6qw6X2wzkfnscFBa7Rsgld/DiN14d0Hm+TVfI3IFBDF5S1LGGejXTBbMB0GA1UdDgQWBBSSiuNgxqqnz2r/jRcWsARqphwQ/zAPBgNVHRMBAf8EBTADAQH/MA4GA1UdDwEB/wQEAwIBBjAZBgNVHREEEjAqhG5zcGlmZmU6Ly9sb2NhbnBDAKBggqhkJOPQDBANoADBlAjEA54Q8hfhEd4qVycwbLNzOm/HQrp1nl+a2xc88iU036FMPancR1PLqgsODPfWyttDRAjAKIodUi4eYiMa9+I2rVbj8gOxJAFn0hLLEF3QDmXtGPPARs9qC+KbiklTu5Fpik2Q="
      ]
    },
    {
      "use": "jwt-svid",
      "kty": "EC",
      "kid": "6d02Vc2oU62mXVH5nlggHGLmfIhrlnNW",
      "crv": "P-256",
      "x": "S2V42XlFjNp30CFmOidbWQT9IpZHqJ8JuuJgDBvkdZA",
      "y": "vN0y5TK36VRxZo_E3Gc7S5c0jIRIaHZ53f2UiJlNFto"
    }
  ],
  "spiffe_sequence": 10,
  "spiffe_refresh_hint": 300
}
```

The use parameter in the JSON Web Key indicates the credential format the key is intended 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
```

Arndt: Bundle doesn't match X509-SVID. This needs to be fixed.

5.2. Alternative methods to avoid

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

5.2.1. 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.

5.2.2. 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.

5.2.3. 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.

5.2.4. Using the JWT-SVID iss claim

JWT-SVIDs 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 JWT-SVIDs and only works when the iss claim is present, which is not guaranteed and not part of the JWT-SVID specification.

The narrow scope of applicability does not make it a viable alternative to the SPIFFE Bundle Endpoint. In combination with interoperability concerns, this approach is NOT RECOMMENDED.

6. Security Considerations

TODO Security

7. IANA Considerations

This document has no IANA actions.

8. Normative References

[Headless_JWT]

"Headless-JWT", n.d., <foo>.

[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>>.

- [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>>.
- [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>>.

Acknowledgments

TODO acknowledge.

Authors' Addresses

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

Pieter Kasselmann
SPIRL

Email: pieter@spirl.com