Web Authorization Protocol
Internet-Draft
Intended status: Informational
Expires: 26 December 2025

P. Kasselman
I. Kazzouzi
SPIRL

24 June 2025

OAuth 2.0 Dynamic Client Registration with Trusted Issuer Credentials
draft-kasselman-oauth-dcr-trusted-issuer-token-01

Abstract

An OAuth 2.0 client requires specific information to interact with an authorization server, including a client identifier registered with the authorization server. The OAuth 2.0 Dynamic Client Registration Protocol [RFC7591] defines a mechansim for dynamic client registration that removes the need for manual registration. This provides a more scalable mechanism that can be used by clients that do not have a pre-existing relationship with an authorization server, or where manually configuring such relationships is prohibitive from a cost and scale perspective. Examples of deployments that benefit from dynamic client registration includes modern cloud architectures where microservices are created on demand to meet scale requirements or for use with emerging protocols like the Model Context Protocol (MCP) [MCP] which requires clients to register with an authorization server even if they do not have a predefined relationship with the authorization server. Similar to modern cloud native workloads, MCP service integrations may be ephemeral in nature. The OAuth 2.0 Dynamic Client Registration Protocol [RFC7591] defines a software statement that includes metadata about the client and is signed by a developer or trusted third party. This specification describes the use of specific third party credentials issued to workloads and applications as software statements. The specification describes two types of credentials that may be used as software statements. The first is Secure Production Identity Framework For Everyone (SPIFFE) credentials. The second is the use of JWT representation of a Verifiable Credentials [VC-JWT]

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://PieterKas.github.io/OAuth-2.0-DCR-with-Trusted-Issuer-Credentials/draft-kasselman-oauth-dcr-trusted-issuer-token.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-kasselman-oauth-dcr-trusted-issuer-token/.

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/PieterKas/OAuth-2.0-DCR-with-Trusted-Issuer-Credentials.

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 26 December 2025.

Copyright Notice

Table of Contents

1. Introduction

The OAuth framework [RFC6749] is a widely deployed authorization protocol standard that enables applications to obtain limited access to user resources. OAuth clients must be registered with the OAuth authorization server, which poses significant operational challenges in dynamically scaling environments. Manual registration and subsequent lifecycle management is common, but is costly, difficult to scale and hard to maintain throughout the lifecycle of a client. These challenges are exacerbated by the increasingly dynamic nature of modern cloud native workloads that are instantiated as needed. In addition, emerging protocols like Model Context Protocol (MCP) [MCP] requires clients to register with an authorization server, but these clients may not always have a predefined relationship with the authorization server, and similar to modern cloud native workloads, may be ephemeral in nature.

[RFC7591] defines a mechansim for dynamic client registration that removes the need for manual registration. This provides a more scalable mechanism that can be used by clients that do not have a pre-existing relationship with an authorization server, or where manually configuring such relationships is prohibitive from a cost and scale perspective. It defines the concept of a software statement, which is a JSON Web Token (JWT) [RFC7519] that asserts metadata values about the client that is signed by a developer or trusted third party.

This specification describes the use of SPIFFE JWT-SVIDs and Verifiable Credentials as software statements that can be submitted as software statements when using OAuth 2.0 Dynamic Client Registration Protocol [RFC7591].

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.

3. Overview

This specification describes the use of two credential types as software statements, namely Secure Production Identity Framework For Everyone (SPIFFE) credentials and Verifiable Credentials (VC).

3.1. SPIFFE Credentials as Software Statements

The Secure Production Identity Framework For Everyone (SPIFFE) is a graduated Cloud Native Compute Foundation project designed to dynamically attest and verify workload identity (see [SPIFFE]. SPIFFE is commonly deployed to support large scale deployment of workloads in enterprise and cloud native compute environments. It ensures that every workload is attested and issued with X.509-SVID [SPIFFE_X509] and JWT-SVID [SPIFFE_JWT] credentials. JWT-SVIDs contain a minimal set of claims, but may be extended to include additional claims, including those defined in the Dynamic Client Registration specification (see Section 2 of [RFC7591]). A SPIFFE issuer may attest a workload, add additional metadata based on the attestation information and issue a JWT-SVID. Workloads that are provisioned with JWT-SVIDs present these credentials as software statements when using Dynamic Client Registration. The registration endpoint is configured to trust the SPIFFE issuer and can verify the JWT-SVID, as well as retrieve additional claims that represents metadata attested to by the SPIFFE issuer. Upon succesful validation of the software statement, the registration endpoint completes the client registration. When SPIFFE JWT-SVIDs are used, the authorization server MUST NOT provision additional client secrets, as the workload can use its JWT-SVID or X.509 SVID to authenticate when initiating any OAuth flow. This removes the risk of additional secrets proliferation, reduces the overheads of securing and managing secrets and improves the overall risk posture.

3.2. Verifiable Credentials as Software Statements

A Verifiable Credential (VC) is a tamper-evident digital assertion made by an issuer about a subject, which can be cryptographically verified by a third party. VCs follow an issuer寰塗older寰砺erifier model where the issuer creates and signs the credential, the holder stores and presents it, and the verifier checks its authenticity and validity. VCs can be represented as a JSON Web Token (JWT) [VC-JWT]. A VC encoded as a JWT can serve as a software statement in dynamic client registration by including metadata as defined in Section 2 of [RFC7591] about the client as additional claims in the VC. Workloads that are provisioned with VCs present these credentials as software statements when using Dynamic Client Registration. The registration endpoint is configured to trust the VC issuer and can verify the VC and use the additional metadata claims included by the issuer. Upon succesful validation of the software statement, the registration endpoint completes the client registration. The use of Verifiable Credentials for client authentication is currently undefined and it is up to the authorization server to make a risk based decision on which authentication methods to use.

3.3. SPIFFE and VC Deployment Models

SPIFFE and VC address different deployment models. SPIFFE is more commonly deployed within large enterprise compute environments and provides high levels of assurance about a workload identity (in this case an OAuth client) along with a highly robust, low latency provisioning pipeline with built in credential lifecycle management to maximise availability. It is commonly deployed as an alternative to managing secrets.

VCs provide an option for clients that are deployed outside an enterprise where SPIFFE is not deployed. These clients use VC issuance protocols to obtain VCs. The use of these protocols or any attestation that needs to be provided is out of scope for this specification.

3.4. Protocol Flow

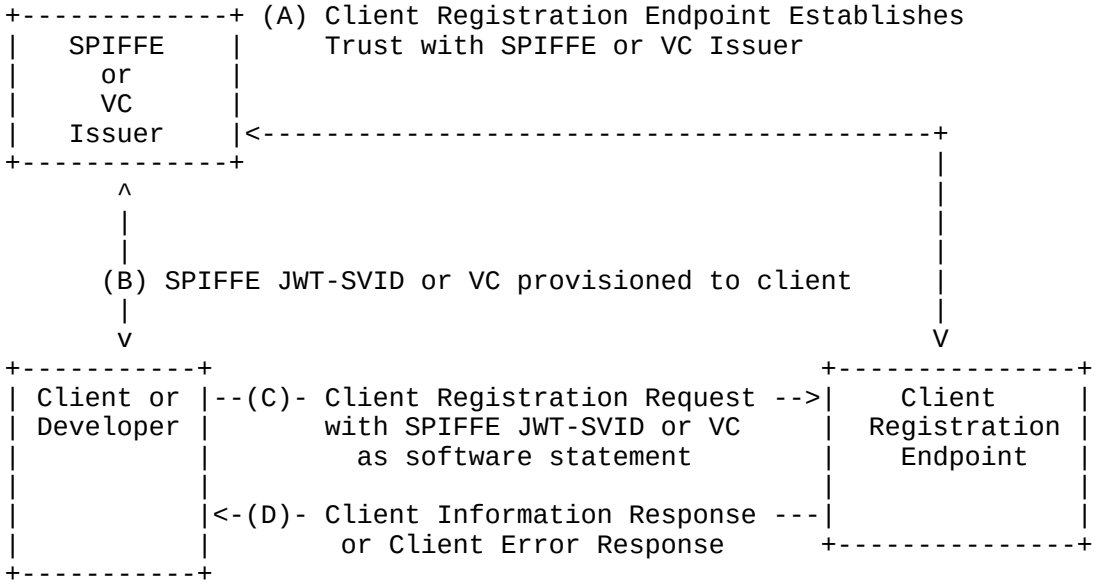

Figure 1: Abstracted Dynamic Client Registration Flow with Software Statement

The OAuth 2.0 Client Dynamic Registration flow illustrated in Figure 1 describes the interaction between the SPIFFE or VC issuers, the client and the client registration endpoint defined in [RFC7591]. It includes the following steps:

* (A) The client registration endpoint establishes a trust relationship with the SPIFFE or VC Issuer. Once trust is established, the client registration endpoint can verify credentials issued by the SPIFFE or VC Issuer when presented as software statements.

* (B) The client is attested and credentialed by the SPIFFE Issuer [SPIFFE], after which it assigns a SPIFFE ID [SPIFFE_ID], along with SPIFFE credentials (e.g. JWT-SVID [SPIFFE_JWT] and X.509-SVID [SPIFFE_X509]). Alternatively a VC issuer issues and provisions a VC. Issuers may include additional claims containing metadata (e.g the request URI).

* (C) The client calls the client registration endpoint with the client’s desired registration metadata, and the software statement (either a SPIFFE JWT-SVID or a VC).

* (D) The client registration endpoint verifies the SPIFFE JWT-SVID or VC that was presented as a software statement. It extracts any additional claims that should be used as metadata before registering the client. It may return additional metadata, client identifiers and optionally credentials if a VC is used. Clients with SPIFFE credentials MUST use the SPIFFE credentials when authenticating as part of an OAuth flow.

4. Issuer and Client Registration Endpoint Trust Relationship

4.1. SPIFFE Issuer and Client Registration Endpoint Trust Relationship

SPIFFE makes provision for multiple Trust Domains, which are represented in the workload identifier. Trust Domains offers additional segmentation within a SPIFFE deployment and each Trust Domain has its own keys for signing credentials. The OAuth authorization server may choose to trust one or more trust domains as defined in [SPIFFE-OAUTH-CLIENT-AUTH].

4.2. Verifiable Credentials and Client Registration Endpoint Trust Relationship

To validate Verifiable Credentials (VCs) during dynamic client registration, the client registration endpoint MUST be configured to trust the VC issuer寰洰 public keys or a certificate chain anchored in a trusted root. This trust may be established statically or through a dynamic trust discovery mechanism. The registration endpoint SHOULD periodically refresh or revalidate the issuer寰洰 key material against a trusted source to ensure ongoing integrity. When a client presents a VC, the registration endpoint verifies its signature using the trusted key material. It MAY also perform additional validation steps, such as checking the issuer寰洰 identity, validating the credential寰洰 expiration, and evaluating its revocation status using mechanisms such as decentralized revocation registries or issuer-specific status endpoints.

5. Client Registration Endpoint Processing

The client registration endpoint MUST process software statements (e.g., SPIFFE JWT-SVIDs or Verifiable Credentials) as follows: 1. Verify Authenticity and Integrity: Validate the software statement’s signature using a trusted issuer key (e.g., SPIFFE Trust Bundle or VC issuer metadata). 2. Validate Credential Expiry: Check that the credential is not expired and is within its valid time window. 3. Evaluate Revocation Status (if applicable): Optionally verify revocation status via decentralized registries or issuer-specific status endpoints (for VCs). 4. Extract and Validate Metadata Claims: Parse the credential and validate required claims needed for OAuth

client registration (e.g., client_name, redirect_uris, grant_types). 5. Register the Client: If all validations succeed, proceed with client registration using the verified metadata. 6. Return Response: Send an appropriate OAuth client registration response.

5.1. SPIFFE JWT-SVIDs Processing by Client Registration Endpoint

When a client presents a SPIFFE JWT-SVID as a software statement, the client registration endpoint MUST:

1. Verify the Signature: Validate the JWT signature using the trusted key material from the SPIFFE Trust Bundle corresponding to the issuer宴洯 trust domain.

2. Validate Standard Claims: Ensure that the JWT includes and satisfies the following claims:

   * iss: Must match the expected SPIFFE trust domain.

   * sub: Must be a valid SPIFFE ID URI identifying the client.

   * aud: Must include the registration endpoint宴洯 audience value.

   * exp and iat: Must be present and within acceptable bounds.

3. Extract and Validate Metadata Claims: Parse and validate any additional claims used to supply OAuth client metadata (e.g., client_name, redirect_uris, scope).

4. Reject Invalid Credentials: Reject the registration if the JWT is expired, not correctly signed, or missing required claims.

5. Register the Client (if all checks succeed): Upon successful validation, register the client using the extracted metadata and return a client identifier. The server MUST NOT issue a client secret, as the SPIFFE credential is used for subsequent authentication.

5.2. VCs Processing by Client Registration Endpoint

When a client presents a Verifiable Credential (VC) as a software statement, the client registration endpoint MUST: 1. Verify the Credential Signature: Validate the cryptographic signature of the VC using the issuer宴洯 trusted public key or certificate chain. 2. Check Credential Validity: Ensure the VC has not expired and is not revoked, using appropriate status mechanisms such as decentralized revocation registries or issuer-hosted status endpoints. 3. Extract and Validate Metadata Claims: Extract required client metadata claims

(e.g., client_name, redirect_uris, scope) embedded in the VC, and validate their presence and format according to the authorization server寰冭 policy. 4. Reject Invalid Credentials: Reject the registration request if the VC fails any verification step, is malformed, or lacks required claims.

Optionally, upon successful validation, the registration endpoint MAY provision additional credentials but SHOULD avoid provisioning traditional client secrets to minimize risk.

6. Security Considerations

6.1. Client Secrets

Dynamic client registration is designed to increase the ease with which clients are registered in large scale deployments. The increased use of client secrets amplifies the risks of secret theft and information compromise. This risk is further amplified if those secrets are long lived or infrequently rotated. Clients that are provisioned with SPIFFE JWT-SVIDs or X.509-SVIDs, and use this specification MUST use them not only as software statements, but also when authenticating to the authorization server in subsequent OAuth flows.

The use of VCs as client authentication mechansims in OAuth is undefine. Consequently, when using VCs as software statements, the authorization server MAY provision additional credentials, but SHOULD avoid provisioning client secrets to limit the risks of secret proliferation and the consequences of secret theft.

7. IANA Considerations

This document has no IANA actions.

8. Normative References

[MCP] "Model Context Protocol", n.d..

[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>.

[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>.

[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>.

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

[SPIFFE-OAUTH-CLIENT-AUTH] "OAuth SPIFFE Client Authentication", n.d., <foo>.

[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>.

[VC-JWT] Zundel, B., Steele, O., and K. Yasuda, "Securing Verifiable Credentials using JSON Web Tokens", 14 June 2023, <https://www.w3.org/TR/vc-jwt/>.

Appendix A. Acknowledgments

TODO acknowledge.

Appendix B. Document History

[[To be removed from the final specification]] -latest * Editorial updates * Corrected markup to show all authors

-00 * Initial draft submitted to Datatracker

Authors' Addresses

Pieter Kasselman
SPIRL
Email: pieter@spirl.com

Ismael Kazzouzi
SPIRL
Email: ismael@spirl.com