Web Authorization Protocol
Internet-Draft
Intended status: Standards Track
Expires: 24 March 2026

A. Parecki
Okta
B. Campbell
Ping Identity

20 September 2025

DPoP for the OAuth 2.0 Device Authorization Grant
draft-parecki-oauth-dpop-device-flow-00

Abstract

The OAuth 2.0 Device Authorization Grant [RFC8628] is an authorization flow for devices with limited input capabilities. Demonstrating Proof of Possession (DPoP) [RFC9449] is a mechanism to sender-constrain OAuth 2.0 tokens. This document describes how to use DPoP with the Device Authorization Grant to provide a higher level of security for clients. It binds the DPoP key to the entire transaction, from the initial device authorization request through the lifetime of the issued tokens.

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://drafts.aaronpk.com/oauth-dpop-device-flow/draft-parecki-oauth-dpop-device-flow.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-parecki-oauth-dpop-device-flow/.

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/aaronpk/oauth-dpop-device-flow.

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/.

Copyright Notice

Table of Contents

1. Introduction

The OAuth 2.0 Device Authorization Grant [RFC8628] provides a mechanism for devices that lack a browser or have constrained input capabilities to obtain an OAuth [RFC6749] access token. The flow involves the device polling the token endpoint while the user authorizes the request on a separate, more capable device.

OAuth 2.0 Demonstrating Proof of Possession (DPoP) [RFC9449] introduces a mechanism for sender-constraining access and refresh tokens. It works by requiring the client to prove possession of a cryptographic key with every request, binding the tokens to that key. [RFC9449] explicitly details its application with Pushed Authorization Requests (PAR) [RFC9126], a flow that, like the Device Authorization Grant, begins with a direct back-channel request from the client to the authorization server.

This specification formally defines the mechanism of using DPoP with the Device Authorization Grant. By requiring a DPoP proof at the beginning of the flow, the authorization server can bind the device_code to a specific public key. This ensures that only the client possessing the corresponding private key can complete the flow by polling the token endpoint, thereby preventing a stolen device_code from being redeemed by a malicious actor. The resulting access and refresh tokens are also DPoP-bound, mitigating the risk of token leakage.

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. Protocol Flow

The overall Device Authorization Grant flow remains as defined in [RFC8628], with the addition of the DPoP header and associated validation logic at two key steps: the device authorization request and the device access token (polling) request.

3.1. Device Authorization Request

To initiate the flow, the client makes a POST request to the device authorization endpoint as specified in Section 3.1 of [RFC8628]. When using DPoP, this request MUST include a DPoP header field containing a valid DPoP proof JWT as defined in Section 4 of [RFC9449].

The authorization server MUST validate the DPoP proof according to the rules in Section 4.3 of [RFC9449]. If the DPoP proof is invalid, the authorization server MUST return an error response, and it is RECOMMENDED that this be a 400 Bad Request with an error code of invalid_dpop_proof.

If the DPoP proof is valid, the authorization server proceeds as defined in Section 3.2 of [RFC8628] by generating a device_code, user_code, etc. In addition, the authorization server MUST associate the public key from the jwk header of the DPoP proof with the generated device_code and store this association for later verification.

For example:

```
POST /device_authorization HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
DPoP: eyJ0eXAiOiJkcG9wK2p3dCIsImFsZyI6IkVTMjU2IiwiandrIjp7Imt0eSI6Ik...

client_id=1406020730&scope=example_scope
```

3.2. Device Access Token Request

After the user authorizes the request, the client begins polling the token endpoint as described in Section 3.4 of [RFC8628]. Each access token request (polling request) using the urn:ietf:params:oauth:grant-type:device_code grant type MUST include a DPoP header with DPoP proof JWT.

Upon receiving a token request, the authorization server MUST perform the following steps in addition to the processing described in [RFC8628]:

* Look up the data associated with the received device_code.

* Retrieve the public key that was associated with the device_code during the device authorization request.

* Validate the DPoP proof JWT in the DPoP header of the current polling request per Section 4.3 of [RFC9449].

* Verify that the public key in the jwk header of the DPoP proof matches the public key associated with the device_code.

If any of these checks fail, the authorization server MUST reject the request with an invalid_grant error.

If all checks are successful and the user has approved the grant, the authorization server issues an access token and an optional refresh token. The issued access token MUST be bound to the DPoP public key, and the access token response MUST include "token_type": "DPoP"‘, as specified in Section 5 of [RFC9449].

Any issued refresh token MUST also be bound to the same DPoP public key.

4. Security Considerations

4.1. Device Code Binding

The primary security benefit of this specification is the binding of the DPoP key to the device_code at the start of the authorization flow. In the standard Device Authorization Grant, an attacker who obtains a valid device_code (e.g., through log inspection or a compromised device) can start polling the token endpoint. If the attacker completes the user authorization step (e.g., via a phishing attack that tricks the user into entering the user_code), they can obtain the access token.

By binding the device_code to the client's DPoP key, this attack is prevented. The attacker's polling requests to the token endpoint will fail because they cannot produce a valid DPoP proof signed with the private key corresponding to the public key bound to the device_code. Only the legitimate client can successfully redeem the device_code.

4.2. Client Impersonation

This specification does not prevent a malicious application on a device from initiating a device flow and using DPoP correctly. It protects the integrity of a single authorization flow by ensuring the same cryptographic identity (the DPoP key pair) is used throughout. It is not a client authentication mechanism. As such, the security considerations for public clients in Section 5 of [RFC8628] and [RFC9700] remain relevant.

4.3. General DPoP Considerations

All security considerations from [RFC9449] apply, including those regarding DPoP proof replay, nonce usage, signature algorithms, and the need to protect the private key on the client device.

5. IANA Considerations

This document has no IANA actions.

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[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>.

[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>.

[RFC8628] Denniss, W., Bradley, J., Jones, M., and H. Tschofenig, "OAuth 2.0 Device Authorization Grant", RFC 8628, DOI 10.17487/RFC8628, August 2019, <https://www.rfc-editor.org/rfc/rfc8628>.

[RFC9449] Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., and D. Waite, "OAuth 2.0 Demonstrating Proof of Possession (DPoP)", RFC 9449, DOI 10.17487/RFC9449, September 2023, <https://www.rfc-editor.org/rfc/rfc9449>.

[RFC9700] Lodderstedt, T., Bradley, J., Labunets, A., and D. Fett, "Best Current Practice for OAuth 2.0 Security", BCP 240, RFC 9700, DOI 10.17487/RFC9700, January 2025, <https://www.rfc-editor.org/rfc/rfc9700>.

6.2. Informative References

[RFC9126] Lodderstedt, T., Campbell, B., Sakimura, N., Tonge, D., and F. Skokan, "OAuth 2.0 Pushed Authorization Requests", RFC 9126, DOI 10.17487/RFC9126, September 2021, <https://www.rfc-editor.org/rfc/rfc9126>.

Acknowledgments

The authors would like to thank Emelia Smith for her reply on social media suggesting that this document should exist.

Document History

[[To be removed from the final specification]]

-00

* Initial draft

Authors' Addresses

Aaron Parecki
Okta
Email: aaron@parecki.com

Brian Campbell
Ping Identity
Email: bcampbell@pingidentity.com