Stream: Internet Engineering Task Force (IETF)
RFC: 9749
Category: Standards Track
Published: March 2025
ISSN: 2070-1721
Author: D. Gultsch

RFC 9749
Use of Voluntary Application Server Identification (VAPID) in JSON Meta Application Protocol (JMAP) Web Push

Abstract

This document defines a method for JSON Meta Application Protocol (JMAP) servers to advertise their capability to authenticate Web Push notifications using the Voluntary Application Server Identification (VAPID) protocol.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9749.

Table of Contents

1. Introduction

JMAP [RFC8620] specifies how clients can subscribe to events using a protocol that is compatible with Web Push [RFC8030]. Some push services require that the application server authenticate all push messages using the VAPID protocol [RFC8292]. To facilitate that, the client (or user agent in Web Push terminology) needs the VAPID public key of the application server to pass along to the push service when retrieving a new endpoint.

2. Conventions Used in This Document

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. Discovering Support for VAPID

The JMAP capabilities object is returned as part of the standard JMAP session object (see Section 2 of [RFC8620]). Servers supporting this specification **MUST** add a property called `urn:ietf:params:jmap:webpush-vapid` to the capabilities object. The value of this property is an object that **MUST** contain the following information:

`applicationServerKey`: "String"
: The Elliptic Curve Digital Signature Algorithm (ECDSA) public key that the push service will use to authenticate the application server, in its uncompressed form (as described in Section 2.3.3 of [SEC1]) and encoded using base64url encoding [RFC7515]. Current systems use the P-256 curve [FIPS186].

> Informative Note: The format of the application server key was chosen to ensure compatibility with the browser API (Section 7.2 of [PUSH-API]), allowing the key to be directly copied and used without additional transformation. Additionally, as noted in Section 3.2 of [RFC8292], the X9.62 encoding (which is compatible with SEC1 encoding) simplifies key comparisons and is more compact than alternative formats.

4. Issuing Push Notifications

Every time the server sends a push message to a `PushSubscription` URL, it **MUST** authenticate the POST request using the protocol outlined in [RFC8292]. This includes both `StateChange` events and `PushVerification` notifications. To authenticate the request, the server **MUST** use a JSON Web Token (JWT) signed by the private key corresponding to the application server key. This application server key **MUST** be the one that was advertised in the capabilities object at the time the `PushSubscription` was created.

5. Key Rotation

When a server needs to replace its VAPID key, it **MUST** update the `sessionState` per [RFC8620]. The client **MUST** monitor the JMAP session object for changes to the VAPID key and **MUST** recreate its push subscription when it detects such a change.

After key rotation, the server **MAY** continue to send push notifications for existing push subscriptions using the old application server key for a transitional period. This allows clients time to recreate their respective push subscriptions. At the end of the transitional period (or immediately for implementations that do not have one), the server **MUST** destroy push subscriptions that use the old key.

When destroying push subscriptions that include the data type `PushSubscription`, the server **MAY** issue one final `StateChange` push notification using the old URL and application server key to notify the client of changes to the `PushSubscription` data type. This prompts the client to make a `PushSubscription/changes` method call. The response to this call will contain an updated `sessionState`, which refers to a session object that contains the new VAPID key.

A race condition can occur when the server updates its VAPID key after the client has refreshed the session object but before calling the `PushSubscription/set` method. This situation causes the server to send a `PushVerification` object to a push resource URL that is now associated with an outdated VAPID key. Consequently, the push service will reject the `PushVerification` with a 403 (Forbidden) status code, as specified in Section 4.2 of [RFC8292].

To alleviate this problem, the client **MUST** check if the `sessionState` in the response from the `PushSubscription/set` method points to a session object with an `applicationServerKey` that matches their expectations. If there is a mismatch, the client **MAY** retry creating the `PushSubscription`. Additionally, the client **MAY** destroy the `PushSubscription` from the earlier, failed attempt.

6. Security Considerations

During the key rotation process, synchronization issues between the client and server may arise. Specifically, a client might restrict a push subscription with the push service to an outdated key, while the server sends the `PushVerification` object authenticated with the newly rotated key. This mismatch leads to the push service rejecting the `PushVerification` request with a 403 (Forbidden) status code, as specified in Section 4.2 of [RFC8292].

Per the requirements of Section 7.2 of [RFC8620], the server **MUST NOT** retry the rejected `PushVerification` request. Consequently, the `PushVerification` object will not be delivered to the client.

To mitigate such issues, the client is responsible for detecting and resolving any synchronization discrepancies, as outlined in Section 5 of this document.

The inclusion of the `urn:ietf:params:jmap:webpush-vapid` property in the JMAP capabilities object is limited to providing information about the server's support for VAPID. This property does not reveal sensitive information, nor does it introduce new security or privacy risks beyond those inherent to JMAP and Web Push. The security considerations for JMAP [RFC8620] (especially Sections 8.6 and 8.7), Web Push [RFC8030], and VAPID [RFC8292] apply to this document.

7. IANA Considerations

7.1. Registration of the JMAP Capability for VAPID

IANA has registered the following new capability in the "JMAP Capabilities" registry:

Capability Name: `urn:ietf:params:jmap:webpush-vapid`
Intended Use: common
Change Controller: IETF
Security and Privacy Considerations: RFC 9749, Section 6
Reference: RFC 9749

8. References

8.1. Normative References

[FIPS186] NIST, "Digital Signature Standard (DSS)", NIST FIPS 186-5, DOI 10.6028/NIST.FIPS.186-5, February 2023, <https://doi.org/10.6028/NIST.FIPS.186-5>.

[SEC1] Standards for Efficient Cryptography Group, "SEC 1: Elliptic Curve Cryptography", Version 2.0, May 2009, <http://www.secg.org/sec1-v2.pdf>.

[RFC8620] Jenkins, N. and C. Newman, "The JSON Meta Application Protocol (JMAP)", RFC 8620, DOI 10.17487/RFC8620, July 2019, <https://www.rfc-editor.org/info/rfc8620>.

[RFC8030] Thomson, M., Damaggio, E., and B. Raymor, Ed., "Generic Event Delivery Using HTTP Push", RFC 8030, DOI 10.17487/RFC8030, December 2016, <https://www.rfc-editor.org/info/rfc8030>.

[RFC8292] Thomson, M. and P. Beverloo, "Voluntary Application Server Identification (VAPID) for Web Push", RFC 8292, DOI 10.17487/RFC8292, November 2017, <https://www.rfc-editor.org/info/rfc8292>.

[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/info/rfc2119>.

[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/info/rfc8174>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

8.2. Informative References

[PUSH-API] Beverloo, P., Ed., Thomson, M., Ed., and M. Caceres, Ed., "Push API", W3C Working Draft, September 2024, <https://www.w3.org/TR/push-api/>.

Author's Address

Daniel Gultsch
Email: daniel@gultsch.de