Workgroup:	Network Working Group
Internet-Draft:	draft-sawant-capport-api-state-enhancement-02
Published:	22 August 2024
Intended Status:	Informational
Expires:	23 February 2025
Author:	P. Sawant
	Apple Inc.

Captive Portal API State Structure Enhancement

Abstract

This document specifies a new key in Captive Portal API State data structure. The purpose of the new key is to allow clients to perform the client authentication without user interaction.

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 23 February 2025.

Copyright Notice

Table of Contents

1. Introduction

As described in [RFC8908], the Captive Portal API data structure is specified in JavaScript Object Notation (JSON) [RFC8259]. Requests and responses for the Captive Portal API use the "application/captive+json" media type. The original specification specifies key "user-portal-url" to convey the web portal URL to the client. Although in most cases client devices are capable of presenting the web portal to the user, there are types of devices that are not built to support the user interaction with the web portal. This document specifies a new key that allows client to perform the authentication without user interaction.

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. API State Structure Enhancement

Table 1 shows the new key that can be optionally included in the top-level of the JSON structure returned by the API server.

Key	Type	Description
client-authentication-url	string	Provides the URL of the authentication server that **MUST** be accessed over TLS with which the client is authenticated without user interaction. Authentication Server authenticates clients using the HTTP authentication framework specified in [RFC9110]. The server **MUST NOT** require user interaction on the client device. The client **MUST** have a credential to perform the authentication without user interaction.

Table 1

4. Use Cases

- Devices that don't support user interaction or web interface, can join captive networks using "client-authentication-url" key. For example, IoT (Internet of Things) devices or speakers can join captive networks using the credentials provisioned to them.
- Section 2 of [RFC9577] specifies "PrivateToken" HTTP authentication scheme. Use of "PrivateToken" HTTP authentication scheme can attest that the client behind the user agent is likely not a bot attempting to perform some form of automated attack such as credential stuffing.
- User experience improvements by not requiring user's interaction with the Captive Portal system every time client device connects the captive network. For example, many Captive Portal systems require users to accept the same terms of service or solve CAPTCHA or watch some commercial on every connection to captive network.

5. Example Interaction

Upon discovering the URI of the API server, a client connected to a captive network will query the API server to retrieve information about its captive state and conditions to escape captivity. In this example, the client discovered the URI "https://example.org/captive-portal/api/X54PD39JV" using one of the mechanisms defined in [RFC8910].

This example illustrates the use of "client-authentication-url" key, and therefore, excludes "user-portal-url" key from the JSON content sent by the API server. An API server may choose to send both the keys based on its policy and configuration. And the client can choose one of them or both to perform the authentication, based on its configuration and capability.

To request the Captive Portal JSON content, a client sends an HTTP GET request:

```
GET /captive-portal/api/X54PD39JV HTTP/1.1
Host: example.org
Accept: application/captive+json
```

The server then responds with the JSON content for that client:

```
HTTP/1.1 200 OK
Cache-Control: private
Date: Mon, 19 Aug 2024 05:07:35 GMT
Content-Type: application/captive+json

{
    "captive": true,
    "client-authentication-url": "https://server.example.org/auth"
}
```

Upon receiving this information, the client will use it to start an authentication session with the server (as specified by "client- authentication-url" key) to enable access to the external network. The client sends following HTTP request to begin the authentication:

```
GET /auth HTTP/1.1
Host: server.example.org
```

Upon receiving the HTTP request, the server selects appropriate authentication scheme to authenticate the client. This example shows "Bearer" authentication scheme defined in [RFC6750]. Section 16.4.1 of [RFC9110] specifies the list of authentication schemes. The server sends HTTP response message with 401 (Unauthorized) status code along with WWW-Authenticate header:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer realm="example"
```

In response to the received challenge, the client sends an access token in the "Authorization" request header using "Bearer" authentication scheme:

```
GET /auth HTTP/1.1
Host: server.example.org
Authorization: Bearer mF_9.B5f-4.1JqM
```

If the access token is found valid, the server sends a response to the client. An example of such response is:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
{
    "access_token":"mF_9.B5f-4.1JqM",
    "token_type":"Bearer",
    "expires_in":3600,
    "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA"
}
```

On a successful HTTP authentication, the client **SHOULD** query the API server again to verify that it is no longer captive.

When the client requests the Captive Portal JSON content after gaining external network access, the server responds with updated JSON content:

```
HTTP/1.1 200 OK
Cache-Control: private
Date: Mon, 19 Aug 2024 05:08:13 GMT
Content-Type: application/captive+json

{
    "captive": false,
    "venue-info-url": "https://flight.example.com/entertainment",
    "seconds-remaining": 326,
    "can-extend-session": true
}
```

6. Security Considerations

This document recommends security considerations specified in Section 7 of [RFC8908].

7. Privacy Considerations

This document recommends privacy consideration specified in Section 7.1 of [RFC8908].

8. IANA Considerations

IANA is requested to add the new key specified in Table 1.

9. References

9.1. 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/info/rfc2119>.

[RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, DOI 10.17487/RFC6750, October 2012, <https://www.rfc-editor.org/info/rfc6750>.

[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>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC8908] Pauly, T., Ed. and D. Thakore, Ed., "Captive Portal API", RFC 8908, DOI 10.17487/RFC8908, September 2020, <https://www.rfc-editor.org/info/rfc8908>.

[RFC8910] Kumari, W. and E. Kline, "Captive-Portal Identification in DHCP and Router Advertisements (RAs)", RFC 8910, DOI 10.17487/RFC8910, September 2020, <https://www.rfc-editor.org/info/rfc8910>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

9.2. Informative References

[RFC9577] Pauly, T., Valdez, S., and C. A. Wood, "The Privacy Pass HTTP Authentication Scheme", RFC 9577, DOI 10.17487/RFC9577, June 2024, <https://www.rfc-editor.org/info/rfc9577>.

Author's Address

Paresh Sawant
Apple Inc.
Email: paresh_sawant@apple.com