

Captive Portal Interaction  
Internet-Draft  
Intended status: Standards Track  
Expires: 20 December 2020

T. Pauly, Ed.  
Apple Inc.  
D. Thakore, Ed.  
CableLabs  
18 June 2020

Captive Portal API  
draft-ietf-capport-api-08

Abstract

This document describes an HTTP API that allows clients to interact with a Captive Portal system. With this API, clients can discover how to get out of captivity and fetch state about their Captive Portal sessions.

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 20 December 2020.

Copyright Notice

Copyright (c) 2020 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	2
3. Workflow . . . . .	3
4. API Connection Details . . . . .	3
4.1. Server Authentication . . . . .	4
5. API State Structure . . . . .	5
6. Example Interaction . . . . .	6
7. Security Considerations . . . . .	8
7.1. Privacy Considerations . . . . .	8
8. IANA Considerations . . . . .	8
8.1. Captive Portal API JSON Media Type Registration . . . . .	9
8.2. Captive Portal API Keys Registry . . . . .	9
9. Acknowledgments . . . . .	10
10. References . . . . .	10
10.1. Normative References . . . . .	10
10.2. Informative References . . . . .	12
Authors' Addresses . . . . .	12

## 1. Introduction

This document describes a HyperText Transfer Protocol (HTTP) Application Program Interface (API) that allows clients to interact with a Captive Portal system. The API defined in this document has been designed to meet the requirements in the Captive Portal Architecture [I-D.ietf-capport-architecture]. Specifically, the API provides:

- \* The state of captivity (whether or not the client has access to the Internet)
- \* A URI of a user-facing web portal that can be used to get out of captivity
- \* Authenticated and encrypted connections, using TLS for connections to both the API and user-facing web portal

## 2. Terminology

This document leverages the terminology and components described in [I-D.ietf-capport-architecture] and additionally defines the following terms:

- \* **Captive Portal Client:** The client that interacts with the Captive Portal API is typically some application running on the User Equipment that is connected to the Captive Network. This is also referred to as the "client" in this document.

- \* Captive Portal API Server: The server exposing the APIs defined in this document to the client. This is also referred to as the "API server" 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. Workflow

The Captive Portal Architecture defines several categories of interaction between clients and Captive Portal systems:

1. Provisioning, in which a client discovers that a network has a captive portal, and learns the URI of the API server.
2. API Server interaction, in which a client queries the state of captivity and retrieves the necessary information to get out of captivity.
3. Enforcement, in which the enforcement device in the network blocks disallowed traffic.

This document defines the mechanisms used in the second category. It is assumed that the location of the Captive Portal API server has been discovered by the client as part of Provisioning. A set of mechanisms for discovering the API Server endpoint is defined in [I-D.ietf-capport-rfc7710bis].

### 4. API Connection Details

The API server endpoint MUST be accessed over HTTP using an https URI [RFC2818], and SHOULD use the default https port. For example, if the Captive Portal API server is hosted at "example.org", the URI of the API could be "https://example.org/captive-portal/api"

The client SHOULD NOT assume that the URI of the API server for a given network will stay the same, and SHOULD rely on the discovery or provisioning process each time it joins the network.

As described in Section 3 of [I-D.ietf-capport-architecture], the identity of the client needs to be visible to the Captive Portal API server in order for the server to correctly reply with the client's portal state. If the identifier used by the Captive Portal system is the client's set of IP addresses, the system needs to ensure that the same IP addresses are visible to both the API server and the enforcement device.

If the API server needs information about the client identity that is not otherwise visible to it, the URI provided to the client during provisioning SHOULD be distinct per client. Thus, depending on how the Captive Portal system is configured, the URI will be unique for each client host and between sessions for the same client host.

For example, a Captive Portal system that uses per-client session URIs could use "https://example.org/captive-portal/api/X54PD39JV" as its API URI.

#### 4.1. Server Authentication

The purpose of accessing the Captive Portal API over an HTTPS connection is twofold: first, the encrypted connection protects the integrity and confidentiality of the API exchange from other parties on the local network; and second, it provides the client of the API an opportunity to authenticate the server that is hosting the API. This authentication allows the client to ensure that the entity providing the Captive Portal API has a valid certificate for the hostname provisioned by the network using the mechanisms defined in [I-D.ietf-capport-rfc7710bis], by validating that a DNS-ID [RFC6125] on the certificate is equal to the provisioned hostname.

Clients performing revocation checking will need some means of accessing revocation information for certificates presented by the API server. Online Certificate Status Protocol [RFC6960] (OCSP) stapling, using the TLS Certificate Status Request extension [RFC6066] SHOULD be used. OCSP stapling allows a client to perform revocation checks without initiating new connections. To allow for other forms of revocation checking, especially for clients that do not support OCSP stapling, a captive network SHOULD permit connections to OCSP responders or Certificate Revocation Lists (CRLs) that are referenced by certificates provided by the API server. For more discussion on certificate revocation checks, see Section 6.5 of BCP 195 [RFC7525]. In addition to connections to OCSP responders and CRLs, a captive network SHOULD also permit connections to Network Time Protocol (NTP) [RFC5905] servers or other time-sync mechanisms to allow clients to accurately validate certificates.

Certificates with missing intermediate certificates that rely on clients validating the certificate chain using the URI specified in the Authority Information Access (AIA) extension [RFC5280] SHOULD NOT be used by the Captive Portal API server. If the certificates do require the use of AIA, the captive network MUST allow client access to the host specified in the URI.

If the client is unable to validate the certificate presented by the API server, it MUST NOT proceed with any of the behavior for API interaction described in this document. The client will proceed to interact with the captive network as if the API capabilities were not present. It may still be possible for the user to access the network if the network redirects a cleartext webpage to a web portal.

## 5. API State Structure

The Captive Portal API data structures are specified in JavaScript Object Notation (JSON) [RFC8259]. Requests and responses for the Captive Portal API use the "application/captive+json" media type. Clients SHOULD include this media type as an Accept header in their GET requests, and servers MUST mark this media type as their Content-Type header in responses.

The following key MUST be included in the top-level of the JSON structure returned by the API server:

- \* "captive" (boolean): indicates whether the client is in a state of captivity, i.e it has not satisfied the conditions to access the external network. If the client is captive (i.e. captive=true), it will still be allowed enough access for it to perform server authentication (Section 4.1).

The following keys can be optionally included in the top-level of the JSON structure returned by the API server:

- \* "user-portal-url" (string): provides the URL of a web portal that MUST be accessed over TLS with which a user can interact.
- \* "venue-info-url" (string): provides the URL of a webpage or site that SHOULD be accessed over TLS on which the operator of the network has information that it wishes to share with the user (e.g., store info, maps, flight status, or entertainment).
- \* "can-extend-session" (boolean): indicates that the URL specified as "user-portal-url" allows the user to extend a session once the client is no longer in a state of captivity. This provides a hint that a client system can suggest accessing the portal URL to the user when the session is near its limit in terms of time or bytes.

- \* "seconds-remaining" (number): an integer that indicates the number of seconds remaining, after which the client will be placed into a captive state. The API server SHOULD include this value if the client is not captive (i.e. captive=false) and the client session is time-limited, and SHOULD omit this value for captive clients (i.e. captive=true) or when the session is not time-limited.
- \* "bytes-remaining" (number): an integer that indicates the number of bytes remaining, after which the client will be placed into a captive state. The byte count represents the sum of the total number of IP packet (layer 3) bytes sent and received by the client, including IP headers. Captive portal systems might not count traffic to whitelisted servers, such as the API server, but clients cannot rely on such behavior. The API server SHOULD include this value if the client is not captive (i.e. captive=false) and the client session is byte-limited, and SHOULD omit this value for captive clients (i.e. captive=true) or when the session is not byte-limited.

The valid JSON keys can be extended by adding entries to the Captive Portal API Keys Registry (Section 8). If a client receives a key that it does not recognize, it MUST ignore the key and any associated values. All keys other than the ones defined in this document as "required" will be considered optional.

Captive Portal JSON content can contain per-client data that is not appropriate to store in an intermediary cache. Captive Portal API servers SHOULD set the Cache-Control header field in any responses to "private", or a more restrictive value such as "no-store" [RFC7234].

Client behavior for issuing requests for updated JSON content is implementation-specific, and can be based on user interaction or the indications of seconds and bytes remaining in a given session. If at any point the client does not receive valid JSON content from the API server, either due to an error or due to receiving no response, the client SHOULD continue to apply the most recent valid content it had received; or, if no content had been received previously, proceed to interact with the captive network as if the API capabilities were not present.

## 6. Example Interaction

A client connected to a captive network upon discovering the URI of the API server 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 [I-D.ietf-capport-rfc7710bis].

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, 02 Mar 2020 05:07:35 GMT
Content-Type: application/captive+json
```

```
{
  "captive": true,
  "user-portal-url": "https://example.org/portal.html"
}
```

Upon receiving this information the client will use this information to direct the user to the web portal (as specified by the user-portal-url value) to enable access to the external network. Once the user satisfies the requirements for external network access, 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, 02 Mar 2020 05:08:13 GMT
Content-Type: application/captive+json
```

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

## 7. Security Considerations

One of the goals of this protocol is to improve the security of the communication between client hosts and Captive Portal systems. Client traffic is protected from passive listeners on the local network by requiring TLS-encrypted connections between the client and the Captive Portal API server, as described in Section 4. All communication between the clients and the API server MUST be encrypted.

In addition to encrypting communications between clients and Captive Portal systems, this protocol requires a basic level of authentication from the API server, as described in Section 4.1. Specifically, the API server MUST present a valid certificate on which the client can perform revocation checks. This allows the client to ensure that the API server has authority for the hostname that was provisioned by the network using [I-D.ietf-capport-rfc7710bis]. Note that this validation only confirms that the API server matches what the network's provisioning mechanism (such as DHCP or IPv6 Router Advertisements) provided, and not validating the security of those provisioning mechanisms or the user's trust relationship to the network.

### 7.1. Privacy Considerations

Information passed between a client and the user-facing web portal may include a user's personal information, such as a full name and credit card details. Therefore, it is important that both the user-facing web portal and the API server that points a client to the web portal are only accessed over encrypted connections.

It is important to note that although communication to the user-facing web portal requires using TLS, the authentication only validates that the web portal server matches the name in the URI provided by the API server. Since this is not a name that a user typed in, the hostname of the web site that would be presented to the user may include "confusable characters" that can mislead the user. See Section 12.5 of [RFC8264] for a discussion of confusable characters.

## 8. IANA Considerations

IANA is requested to create a registration for an "application/captive+json" media type (Section 8.1) and a registry for fields in that format (Section 8.2).



### 8.1. Captive Portal API JSON Media Type Registration

This document registers the media type for Captive Portal API JSON text, "application/captive+json".

Type name: application

Subtype name: captive+json

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: Encoding considerations are identical to those specified for the "application/json" media type.

Security considerations: See Section 7

Interoperability considerations: This document specifies format of conforming messages and the interpretation thereof.

Published specification: This document

Applications that use this media type: This media type is intended to be used by servers presenting the Captive Portal API, and clients connecting to such captive networks.

Fragment identifier considerations: N/A

Additional information: N/A

Person and email address to contact for further information: See Authors' Addresses section

Intended usage: COMMON

Restrictions on usage: N/A

Author: CAPPOR IETF WG

Change controller: IETF

### 8.2. Captive Portal API Keys Registry

IANA is asked to create and maintain a new registry called "Captive Portal API Keys", which will reserve JSON keys for use in Captive Portal API data structures. The initial contents of this registry are provided in Section 5.

Each entry in the registry contains the following fields:

Key: The JSON key being registered, in string format.

Type: The type of the JSON value to be stored, as one of the value types defined in [RFC8259].

Description: A brief description explaining the meaning of the value, how it might be used, and/or how it should be interpreted by clients.

Specification: A reference to a specification that defines the key and explains its usage.

New assignments for Captive Portal API Keys Registry will be administered by IANA using the Specification Required policy [RFC8126]. The Designated Expert is expected to validate the existence of documentation describing new keys in a permanent publicly available specification, such as an Internet Draft or RFC. The expert is expected to validate that new keys have a clear meaning and do not create unnecessary confusion or overlap with existing keys. Keys that are specific to non-generic use cases, particularly ones that are not specified as part of an IETF document, are encouraged to use a domain-specific prefix.

## 9. Acknowledgments

This work in this document was started by Mark Donnelly and Margaret Cullen. Thanks to everyone in the CAPPOR Working Group who has given input.

## 10. References

### 10.1. 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>>.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <<https://www.rfc-editor.org/info/rfc2818>>.

- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.
- [RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <<https://www.rfc-editor.org/info/rfc6066>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <<https://www.rfc-editor.org/info/rfc6960>>.
- [RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <<https://www.rfc-editor.org/info/rfc7234>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [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>>.

## 10.2. Informative References

- [I-D.ietf-capport-architecture]  
Larose, K., Dolson, D., and H. Liu, "CAPPORT Architecture", Work in Progress, Internet-Draft, draft-ietf-capport-architecture-08, 11 May 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-capport-architecture-08.txt>>.
- [I-D.ietf-capport-rfc7710bis]  
Kumari, W. and E. Kline, "Captive-Portal Identification in DHCP / RA", Work in Progress, Internet-Draft, draft-ietf-capport-rfc7710bis-07, 23 May 2020, <<http://www.ietf.org/internet-drafts/draft-ietf-capport-rfc7710bis-07.txt>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.
- [RFC8264] Saint-Andre, P. and M. Blanchet, "PRECIS Framework: Preparation, Enforcement, and Comparison of Internationalized Strings in Application Protocols", RFC 8264, DOI 10.17487/RFC8264, October 2017, <<https://www.rfc-editor.org/info/rfc8264>>.

## Authors' Addresses

Tommy Pauly (editor)  
Apple Inc.  
One Apple Park Way  
Cupertino, California 95014,  
United States of America

Email: [tpauly@apple.com](mailto:tpauly@apple.com)

Darshak Thakore (editor)  
CableLabs  
858 Coal Creek Circle  
Louisville, CO 80027,  
United States of America

Email: [d.thakore@cablelabs.com](mailto:d.thakore@cablelabs.com)

Internet Engineering Task Force  
Internet-Draft  
Intended status: Informational  
Expires: 27 March 2021

K. Larose  
Agilicus  
D. Dolson

H. Liu  
Google  
23 September 2020

Captive Portal Architecture  
draft-ietf-capport-architecture-10

Abstract

This document describes a captive portal architecture. Network provisioning protocols such as DHCP or Router Advertisements (RAs), an optional signaling protocol, and an HTTP API are used to provide the solution.

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 27 March 2021.

Copyright Notice

Copyright (c) 2020 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	3
1.1. Requirements Language . . . . .	5
1.2. Terminology . . . . .	5
2. Components . . . . .	6
2.1. User Equipment . . . . .	6
2.2. Provisioning Service . . . . .	7
2.2.1. DHCP or Router Advertisements . . . . .	8
2.2.2. Provisioning Domains . . . . .	8
2.3. Captive Portal API Server . . . . .	8
2.4. Captive Portal Enforcement Device . . . . .	9
2.5. Captive Portal Signal . . . . .	10
2.6. Component Diagram . . . . .	10
3. User Equipment Identity . . . . .	12
3.1. Identifiers . . . . .	12
3.2. Recommended Properties . . . . .	12
3.2.1. Uniquely Identify User Equipment . . . . .	13
3.2.2. Hard to Spoof . . . . .	13
3.2.3. Visible to the API Server . . . . .	13
3.2.4. Visible to the Enforcement Device . . . . .	14
3.3. Evaluating Types of Identifiers . . . . .	14
3.4. Example Identifier Types . . . . .	14
3.4.1. Physical Interface . . . . .	14
3.4.2. IP Address . . . . .	15
3.4.3. Media Access Control (MAC) Address . . . . .	16
3.5. Context-free URI . . . . .	16
4. Solution Workflow . . . . .	17
4.1. Initial Connection . . . . .	17
4.2. Conditions About to Expire . . . . .	17
4.3. Handling of Changes in Portal URI . . . . .	18
5. Acknowledgments . . . . .	18
6. IANA Considerations . . . . .	19
7. Security Considerations . . . . .	19
7.1. Trusting the Network . . . . .	19
7.2. Authenticated APIs . . . . .	19
7.3. Secure APIs . . . . .	20
7.4. Risks Associated with the Signaling Protocol . . . . .	20
7.5. User Options . . . . .	21

7.6. Privacy . . . . .	21
8. References . . . . .	21
8.1. Normative References . . . . .	21
8.2. Informative References . . . . .	22
Appendix A. Existing Captive Portal Detection Implementations .	23
Authors' Addresses . . . . .	23

## 1. Introduction

In this document, "Captive Portal" is used to describe a network to which a device may be voluntarily attached, such that network access is limited until some requirements have been fulfilled. Typically a user is required to use a web browser to fulfill requirements imposed by the network operator, such as reading advertisements, accepting an acceptable-use policy, or providing some form of credentials.

Implementations of captive portals generally require a web server, some method to allow/block traffic, and some method to alert the user. Common methods of alerting the user in implementations prior to this work involve modifying HTTP or DNS traffic.

This document describes an architecture for implementing captive portals while addressing most of the problems arising for current captive portal mechanisms. The architecture is guided by these requirements:

- \* Current captive portal solutions typically implement some variations of forging DNS or HTTP responses. Some attempt man-in-the-middle (MITM) proxy of HTTPS in order to forge responses. Captive Portal Solutions should not have to break any protocols or otherwise act in the manner of an attacker. Therefore, solutions MUST NOT require the forging of responses from DNS or HTTP servers, or any other protocol.
- \* Solutions MUST permit clients to perform DNSSEC validation, which rules out solutions that forge DNS responses. Solutions SHOULD permit clients to detect and avoid TLS man-in-the-middle attacks without requiring a human to perform any kind of "exception" processing.
- \* To maximize universality and adoption, solutions MUST operate at the layer of Internet Protocol (IP) or above, not being specific to any particular access technology such as Cable, WiFi or mobile telecom.
- \* Solutions SHOULD allow a device to query the network to determine whether the device is captive, without the solution being coupled to forging intercepted protocols or requiring the device to make

sacrificial queries to "canary" URIs to check for response tampering (see Appendix A). Current captive portal solutions that work by affecting DNS or HTTP generally only function as intended with browsers, breaking other applications using those protocols; applications using other protocols are not alerted that the network is a captive portal.

- \* The state of captivity SHOULD be explicitly available to devices via a standard protocol, rather than having to infer the state indirectly.
- \* The architecture MUST provide a path of incremental migration, acknowledging the existence of a huge variety of pre-existing portals and end-user device implementations and software versions. This requirement is not to recommend or standardize existing approaches, rather to provide device and portal implementors a path to new standard.

A side-benefit of the architecture described in this document is that devices without user interfaces are able to identify parameters of captivity. However, this document does not describe a mechanism for such devices to negotiate for unrestricted network access. A future document could provide a solution to devices without user interfaces. This document focuses on devices with user interfaces.

The architecture uses the following mechanisms:

- \* Network provisioning protocols provide end-user devices with a Uniform Resource Identifier [RFC3986] (URI) for the API that end-user devices query for information about what is required to escape captivity. DHCP, DHCPv6, and Router-Advertisement options for this purpose are available in [RFC7710bis]. Other protocols (such as RADIUS), Provisioning Domains [I-D.pfister-capport-pvd], or static configuration may also be used to convey this Captive Portal API URI. A device MAY query this API at any time to determine whether the network is holding the device in a captive state.
- \* A Captive Portal can signal User Equipment in response to transmissions by the User Equipment. This signal works in response to any Internet protocol, and is not done by modifying protocols in-band. This signal does not carry the Captive Portal API URI; rather it provides a signal to the User Equipment that it is in a captive state.



- \* Receipt of a Captive Portal Signal provides a hint that User Equipment could be captive. In response, the device MAY query the provisioned API to obtain information about the network state. The device can take immediate action to satisfy the portal (according to its configuration/policy).

The architecture attempts to provide confidentiality, authentication, and safety mechanisms to the extent possible.

### 1.1. Requirements Language

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.

### 1.2. Terminology

**Captive Portal:** A network which limits communication of attached devices to restricted hosts until the user has satisfied Captive Portal Conditions, after which access is permitted to a wider set of hosts (typically the Internet).

**Captive Portal Conditions:** site-specific requirements that a user or device must satisfy in order to gain access to the wider network.

**Captive Portal Enforcement Device:** The network equipment which enforces the traffic restriction. Also known as Enforcement Device.

**Captive Portal User Equipment:** Also known as User Equipment. A device which has voluntarily joined a network for purposes of communicating beyond the constraints of the Captive Portal.

**User Portal:** The web server providing a user interface for assisting the user in satisfying the conditions to escape captivity.

**Captive Portal API:** Also known as API. An HTTP API allowing User Equipment to query information about its state of captivity within the Captive Portal. This information might include how to obtain full network access (e.g. by visiting a URI).

**Captive Portal API Server:** Also known as API Server. A server hosting the Captive Portal API.

**Captive Portal Signal:** A notification from the network used to signal to the User Equipment that the state of its captivity could have changed.

Captive Portal Signaling Protocol: Also known as Signaling Protocol. The protocol for communicating Captive Portal Signals.

Captive Portal Session: Also referred to simply as the "session", a Captive Portal Session is the association for a particular User Equipment that starts when it interacts with the Captive Portal and gains open access to the network, and ends when the User Equipment moves back into the original captive state. The Captive Network maintains the state of each active Session, and can limit Sessions based on a length of time or a number of bytes used. The Session is associated with a particular User Equipment using the User Equipment's identifier (see Section 3).

## 2. Components

### 2.1. User Equipment

The User Equipment is the device that a user desires to be attached to a network with full access to all hosts on the network (e.g., to have Internet access). The User Equipment communication is typically restricted by the Enforcement Device, described in Section 2.4, until site-specific requirements have been met.

This document only considers devices with web browsers, with web applications being the means of satisfying Captive Portal Conditions. An example of such User Equipment is a smart phone.

The User Equipment:

- \* SHOULD support provisioning of the URI for the Captive Portal API (e.g., by DHCP)
- \* SHOULD distinguish Captive Portal API access per network interface, in the manner of Provisioning Domain Architecture [RFC7556].
- \* SHOULD have a non-spoofable mechanism for notifying the user of the Captive Portal
- \* SHOULD have a web browser so that the user may navigate to the User Portal.
- \* SHOULD support updates to the Captive Portal API URI from the network provisioning service.
- \* MAY prevent applications from using networks that do not grant full network access. E.g., a device connected to a mobile network may be connecting to a captive WiFi network; the operating system

could avoid updating the default route to a device on captive WiFi network until network access restrictions have been lifted (excepting access to the User Portal) in the new network. This has been termed "make before break".

None of the above requirements are mandatory because (a) we do not wish to say users or devices must seek full access to the Captive Portal, (b) the requirements may be fulfilled by manually visiting the captive portal web application, and (c) legacy devices must continue to be supported.

If User Equipment supports the Captive Portal API, it MUST validate the API server's TLS certificate (see [RFC2818]) according to the procedures in [RFC6125]. The API server's URI is obtained via a network provisioning protocol, which will typically provide a hostname to be used in TLS server certificate validation, against a DNS-ID in the server certificate. If the API server is identified by IP address, the `iPAddress.subjectAltName` is used to validate the server certificate. An Enforcement Device SHOULD allow access to any services that User Equipment could need to contact to perform certificate validation, such as OCSP responders, CRLs, and NTP servers; see Section 4.1 of [I-D.ietf-capport-api] for more information. If certificate validation fails, User Equipment MUST NOT make any calls to the API server.

The User Equipment can store the last response it received from the Captive Portal API as a cached view of its state within the Captive Portal. This state can be used to determine whether its Captive Portal Session is near expiry. For example, the User Equipment might compare a timestamp indicating when the session expires to the current time. Storing state in this way can reduce the need for communication with the Captive Portal API. However, it could lead to the state becoming stale if the User Equipment's view of the relevant conditions (byte quota, for example) is not consistent with the Captive Portal API's.

## 2.2. Provisioning Service

The Provisioning Service is primarily responsible for providing a Captive Portal API URI to the User Equipment when it connects to the network, and later if the URI changes. The provisioning service could also be the same service which is responsible for provisioning the User Equipment for access to the Captive Portal (e.g., by providing it with an IP address). This section discusses two mechanisms which may be used to provide the Captive Portal API URI to the User Equipment.

### 2.2.1. DHCP or Router Advertisements

A standard for providing a Captive Portal API URI using DHCP or Router Advertisements is described in [RFC7710bis]. The captive portal architecture expects this URI to indicate the API described in Section 2.3.

### 2.2.2. Provisioning Domains

Although still a work in progress, [I-D.pfister-capport-pvd] proposes a mechanism for User Equipment to be provided with PvD Bootstrap Information containing the URI for the API described in Section 2.3.

## 2.3. Captive Portal API Server

The purpose of a Captive Portal API is to permit a query of Captive Portal state without interrupting the user. This API thereby removes the need for User Equipment to perform clear-text "canary" (see Appendix A) queries to check for response tampering.

The URI of this API will have been provisioned to the User Equipment. (Refer to Section 2.2).

This architecture expects the User Equipment to query the API when the User Equipment attaches to the network and multiple times thereafter. Therefore the API MUST support multiple repeated queries from the same User Equipment and return the state of captivity for the equipment.

At minimum, the API MUST provide the state of captivity. Further the API MUST be able to provide a URI for the User Portal. The scheme for the URI MUST be https so that the User Equipment communicates with the User Portal over TLS.

If the API receives a request for state that does not correspond to the requesting User Equipment, the API SHOULD deny access. Given that the API might use the User Equipment's identifier for authentication, this requirement motivates Section 3.2.2.

A caller to the API needs to be presented with evidence that the content it is receiving is for a version of the API that it supports. For an HTTP-based interaction, such as in [I-D.ietf-capport-api] this might be achieved by using a content type that is unique to the protocol.

When User Equipment receives Captive Portal Signals, the User Equipment MAY query the API to check its state of captivity. The User Equipment SHOULD rate-limit these API queries in the event of the signal being flooded. (See Section 7.)

The API MUST be extensible to support future use-cases by allowing extensible information elements.

The API MUST use TLS to ensure server authentication. The implementation of the API MUST ensure both confidentiality and integrity of any information provided by or required by it.

This document does not specify the details of the API.

#### 2.4. Captive Portal Enforcement Device

The Enforcement Device component restricts the network access of User Equipment according to site-specific policy. Typically User Equipment is permitted access to a small number of services (according to the policies of the network provider) and is denied general network access until it satisfies the Captive Portal Conditions.

The Enforcement Device component:

- \* Allows traffic to pass for User Equipment that is permitted to use the network and has satisfied the Captive Portal Conditions.
- \* Blocks (discards) traffic according to the site-specific policy for User Equipment that has not yet satisfied the Captive Portal Conditions.
- \* Optionally signals User Equipment using the Captive Portal Signaling protocol if certain traffic is blocked.
- \* Permits User Equipment that has not satisfied the Captive Portal Conditions to access necessary APIs and web pages to fulfill requirements for escaping captivity.
- \* Updates allow/block rules per User Equipment in response to operations from the User Portal.

## 2.5. Captive Portal Signal

When User Equipment first connects to a network, or when there are changes in status, the Enforcement Device could generate a signal toward the User Equipment. This signal indicates that the User Equipment might need to contact the API Server to receive updated information. For instance, this signal might be generated when the end of a session is imminent, or when network access was denied. For simplicity, and to reduce the attack surface, all signals SHOULD be considered equivalent by the User Equipment: as a hint to contact the API. If future solutions have multiple signal types, each type SHOULD be rate-limited independently.

An Enforcement Device MUST rate-limit any signal generated in response to these conditions. See Section 7.4 for a discussion of risks related to a Captive Portal Signal.

## 2.6. Component Diagram

The following diagram shows the communication between each component in the case where the Captive Portal has a User Portal, and the User Equipment chooses to visit the User Portal in response to discovering and interacting with the API Server.

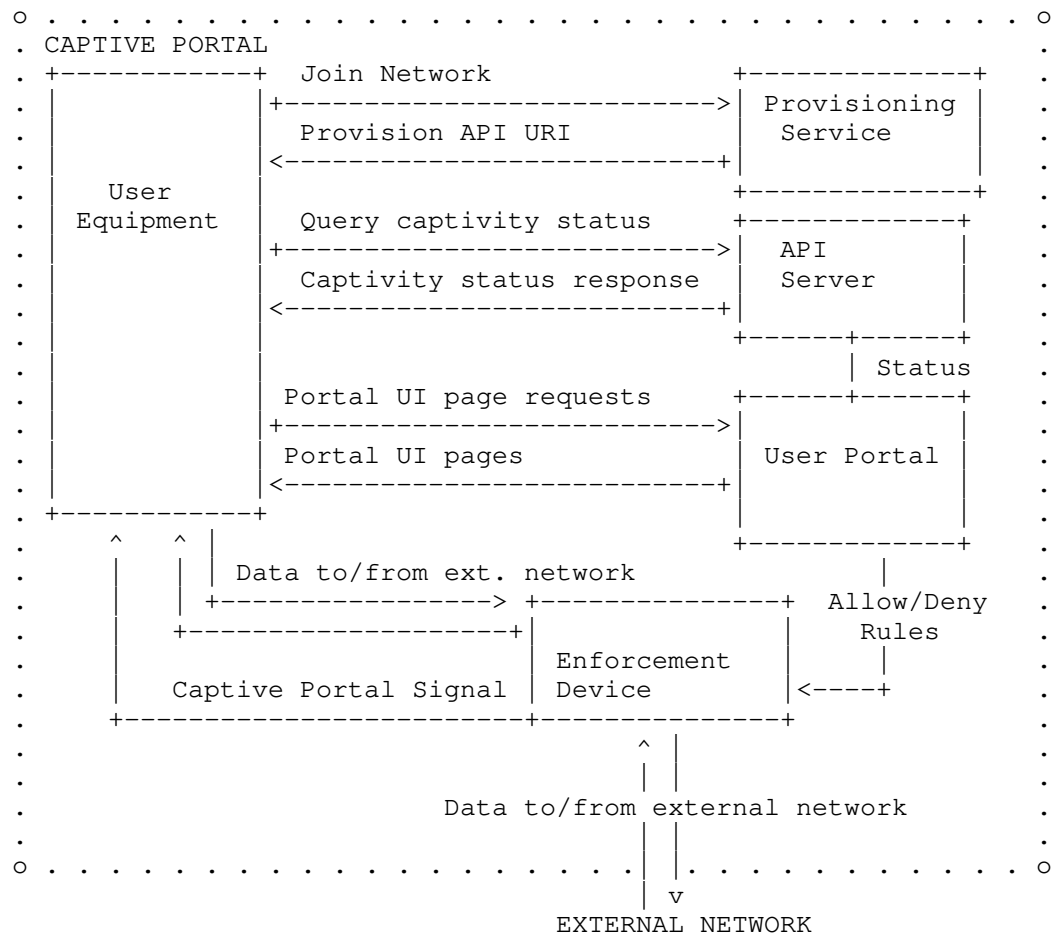


Figure 1: Captive Portal Architecture Component Diagram

In the diagram:

- \* During provisioning (e.g., DHCP), and possibly later, the User Equipment acquires the Captive Portal API URI.
- \* The User Equipment queries the API to learn of its state of captivity. If captive, the User Equipment presents the portal user interface from the User Portal to the user.
- \* Based on user interaction, the User Portal directs the Enforcement Device to either allow or deny external network access for the User Equipment.

- \* The User Equipment attempts to communicate to the external network through the Enforcement Device.
- \* The Enforcement Device either allows the User Equipment's packets to the external network, or blocks the packets. If blocking traffic and a signal has been implemented, it may respond with a Captive Portal Signal.

The Provisioning Service, API Server, and User Portal are described as discrete functions. An implementation might provide the multiple functions within a single entity. Furthermore, these functions, combined or not, as well as the Enforcement Device, could be replicated for redundancy or scale.

### 3. User Equipment Identity

Multiple components in the architecture interact with both the User Equipment and each other. Since the User Equipment is the focus of these interactions, the components must be able to both identify the User Equipment from their interactions with it, and to agree on the identity of the User Equipment when interacting with each other.

The methods by which the components interact restrict the type of information that may be used as an identifying characteristics. This section discusses the identifying characteristics.

#### 3.1. Identifiers

An Identifier is a characteristic of the User Equipment used by the components of a Captive Portal to uniquely determine which specific User Equipment is interacting with them. An Identifier can be a field contained in packets sent by the User Equipment to the External Network. Or, an Identifier can be an ephemeral property not contained in packets destined for the External Network, but instead correlated with such information through knowledge available to the different components.

#### 3.2. Recommended Properties

The set of possible identifiers is quite large. However, in order to be considered a good identifier, an identifier SHOULD meet the following criteria. Note that the optimal identifier will likely change depending on the position of the components in the network as well as the information available to them. An identifier SHOULD:

- \* uniquely identify the User Equipment
- \* be hard to spoof



- \* be visible to the API Server
- \* be visible to the Enforcement Device

An identifier might only apply to the current point of network attachment. If the device moves to a different network location its identity could change.

#### 3.2.1. Uniquely Identify User Equipment

The Captive Portal MUST associate the User Equipment with an identifier that is unique among the User Equipment that are interacting with the Captive Portal at that time.

Over time, the User Equipment assigned to an identifier value MAY change. Allowing the identified device to change over time ensures that the space of possible identifying values need not be overly large.

Independent Captive Portals MAY use the same identifying value to identify different User Equipment. Allowing independent captive portals to reuse identifying values allows the identifier to be a property of the local network, expanding the space of possible identifiers.

#### 3.2.2. Hard to Spoof

A good identifier does not lend itself to being easily spoofed. At no time should it be simple or straightforward for one User Equipment to pretend to be another User Equipment, regardless of whether both are active at the same time. This property is particularly important when the User Equipment identifier is referenced externally by devices such as billing systems, or where the identity of the User Equipment could imply liability.

#### 3.2.3. Visible to the API Server

Since the API Server will need to perform operations which rely on the identity of the User Equipment, such as answering a query about whether the User Equipment is captive, the API Server needs to be able to relate a request to the User Equipment making the request.

#### 3.2.4. Visible to the Enforcement Device

The Enforcement Device will decide on a per-packet basis whether the packet should be forwarded to the external network. Since this decision depends on which User Equipment sent the packet, the Enforcement Device requires that it be able to map the packet to its concept of the User Equipment.

#### 3.3. Evaluating Types of Identifiers

To evaluate whether a type of identifier is appropriate, one should consider every recommended property from the perspective of interactions among the components in the architecture. When comparing identifier types, choose the one which best satisfies all of the recommended properties. The architecture does not provide an exact measure of how well an identifier type satisfies a given property; care should be taken in performing the evaluation.

#### 3.4. Example Identifier Types

This section provides some example identifier types, along with some evaluation of whether they are suitable types. The list of identifier types is not exhaustive. Other types may be used. An important point to note is that whether a given identifier type is suitable depends heavily on the capabilities of the components and where in the network the components exist.

##### 3.4.1. Physical Interface

The physical interface by which the User Equipment is attached to the network can be used to identify the User Equipment. This identifier type has the property of being extremely difficult to spoof: the User Equipment is unaware of the property; one User Equipment cannot manipulate its interactions to appear as though it is another.

Further, if only a single User Equipment is attached to a given physical interface, then the identifier will be unique. If multiple User Equipment is attached to the network on the same physical interface, then this type is not appropriate.

Another consideration related to uniqueness of the User Equipment is that if the attached User Equipment changes, both the API Server and the Enforcement Device MUST invalidate their state related to the User Equipment.

The Enforcement Device needs to be aware of the physical interface, which constrains the environment: it must either be part of the device providing physical access (e.g., implemented in firmware), or packets traversing the network must be extended to include information about the source physical interface (e.g. a tunnel).

The API Server faces a similar problem, implying that it should co-exist with the Enforcement Device, or that the Enforcement Device should extend requests to it with the identifying information.

#### 3.4.2. IP Address

A natural identifier type to consider is the IP address of the User Equipment. At any given time, no device on the network can have the same IP address without causing the network to malfunction, so it is appropriate from the perspective of uniqueness.

However, it may be possible to spoof the IP address, particularly for malicious reasons where proper functioning of the network is not necessary for the malicious actor. Consequently, any solution using the IP address SHOULD proactively try to prevent spoofing of the IP address. Similarly, if the mapping of IP address to User Equipment is changed, the components of the architecture MUST remove or update their mapping to prevent spoofing. Demonstrations of return routeability, such as that required for TCP connection establishment, might be sufficient defense against spoofing, though this might not be sufficient in networks that use broadcast media (such as some wireless networks).

Since the IP address may traverse multiple segments of the network, more flexibility is afforded to the Enforcement Device and the API Server: they simply must exist on a segment of the network where the IP address is still unique. However, consider that a NAT may be deployed between the User Equipment and the Enforcement Device. In such cases, it is possible for the components to still uniquely identify the device if they are aware of the port mapping.

In some situations, the User Equipment may have multiple IP addresses (either IPv4, IPv6 or a dual-stack [RFC4213] combination), while still satisfying all of the recommended properties. This raises some challenges to the components of the network. For example, if the User Equipment tries to access the network with multiple IP addresses, should the Enforcement Device and API Server treat each IP address as a unique User Equipment, or should it tie the multiple addresses together into one view of the subscriber? An implementation MAY do either. Attention should be paid to IPv6 and the fact that it is expected for a device to have multiple IPv6 addresses on a single link. In such cases, identification could be performed by subnet, such as the /64 to which the IP belongs.

#### 3.4.3. Media Access Control (MAC) Address

The MAC address of a device is often used as an identifier in existing implementations. This document does not discuss the use of MAC addresses within a captive portal system, but they can be used as an identifier type, subject to the criteria in Section 3.2.

#### 3.5. Context-free URI

A Captive Portal API needs to present information to clients that is unique to that client. To do this, some systems use information from the context of a request, such as the source address, to identify the User Equipment.

Using information from context rather than information from the URI allows the same URI to be used for different clients. However, it also means that the resource is unable to provide relevant information if the User Equipment makes a request using a different network path. This might happen when User Equipment has multiple network interfaces. It might also happen if the address of the API provided by DNS depends on where the query originates (as in split DNS [RFC8499]).

Accessing the API MAY depend on contextual information. However, the URIs provided in the API SHOULD be unique to the User Equipment and not dependent on contextual information to function correctly.

Though a URI might still correctly resolve when the User Equipment makes the request from a different network, it is possible that some functions could be limited to when the User Equipment makes requests using the Captive Portal. For example, payment options could be absent or a warning could be displayed to indicate the payment is not for the current connection.

URIs could include some means of identifying the User Equipment in the URIs. However, including unauthenticated User Equipment identifiers in the URI may expose the service to spoofing or replay attacks.

#### 4. Solution Workflow

This section aims to improve understanding by describing a possible workflow of solutions adhering to the architecture. Note that the section is not normative: it describes only a subset of possible implementations.

##### 4.1. Initial Connection

This section describes a possible workflow when User Equipment initially joins a Captive Portal.

1. The User Equipment joins the Captive Portal by acquiring a DHCP lease, RA, or similar, acquiring provisioning information.
2. The User Equipment learns the URI for the Captive Portal API from the provisioning information (e.g., [RFC7710bis]).
3. The User Equipment accesses the Captive Portal API to receive parameters of the Captive Portal, including User Portal URI. (This step replaces the clear-text query to a canary URI.)
4. If necessary, the User navigates to the User Portal to gain access to the external network.
5. If the User interacted with the User Portal to gain access to the external network in the previous step, the User Portal indicates to the Enforcement Device that the User Equipment is allowed to access the external network.
6. The User Equipment attempts a connection outside the Captive Portal
7. If the requirements have been satisfied, the access is permitted; otherwise the "Expired" behavior occurs.
8. The User Equipment accesses the network until conditions expire.

##### 4.2. Conditions About to Expire

This section describes a possible workflow when access is about to expire.

1. Precondition: the API has provided the User Equipment with a duration over which its access is valid.
2. The User Equipment is communicating with the outside network.
3. The User Equipment detects that the length of time left for its access has fallen below a threshold by comparing its stored expiry time with the current time.
4. The User Equipment visits the API again to validate the expiry time.
5. If expiry is still imminent, the User Equipment prompts the user to access the User Portal URI again.
6. The User accepts the prompt displayed by the User Equipment.
7. The User extends their access through the User Portal via the User Equipment's user interface.
8. The User Equipment's access to the outside network continues uninterrupted.

#### 4.3. Handling of Changes in Portal URI

A different Captive Portal API URI could be returned in the following cases:

- \* If DHCP is used, a lease renewal/rebind may return a different Captive Portal API URI.
- \* If RA is used, a new Captive Portal API URI may be specified in a new RA message received by end User Equipment.

When the Network Provisioning Service updates the Captive Portal API URI, the User Equipment can retrieve updated state from the URI immediately, or it can wait as it normally would until the expiry conditions it retrieved from the old URI are about to expire.

#### 5. Acknowledgments

The authors thank Lorenzo Colitti for providing the majority of the content for the Captive Portal Signal requirements.

The authors thank Benjamin Kaduk for providing the content related to TLS certificate validation of the API server.

The authors thank Michael Richardson for providing wording requiring DNSSEC and TLS to operate without the user adding exceptions.

The authors thank various individuals for their feedback on the mailing list and during the IETF98 hackathon: David Bird, Erik Kline, Alexis La Goulette, Alex Roscoe, Darshak Thakore, and Vincent van Dam.

## 6. IANA Considerations

This memo includes no request to IANA.

## 7. Security Considerations

### 7.1. Trusting the Network

When joining a network, some trust is placed in the network operator. This is usually considered to be a decision by a user on the basis of the reputation of an organization. However, once a user makes such a decision, protocols can support authenticating that a network is operated by who claims to be operating it. The Provisioning Domain Architecture [RFC7556] provides some discussion on authenticating an operator.

The user makes an informed choice to visit and trust the Captive Portal URI. Since the network provides Captive Portal URI to the user equipment, the network SHOULD do so securely so that the user's trust in the network can extend to their trust of the Captive Portal URI. E.g., the DHCPv6 AUTH option can sign this information.

If a user decides to incorrectly trust an attacking network, they might be convinced to visit an attacking web page and unwittingly provide credentials to an attacker. Browsers can authenticate servers but cannot detect cleverly misspelled domains, for example.

Further, the possibility of an on-path attacker in an attacking network introduces some risks. The attacker could redirect traffic to arbitrary destinations. The attacker could analyze the user's traffic leading to loss of confidentiality. Or, the attacker could modify the traffic inline.

### 7.2. Authenticated APIs

The solution described here requires that when the User Equipment needs to access the API server, the User Equipment authenticates the server; see Section 2.1.

The Captive Portal API URI might change during the Captive Portal Session. The User Equipment can apply the same trust mechanisms to the new URI as it did to the URI it received initially from the network provisioning service.

### 7.3. Secure APIs

The solution described here requires that the API be secured using TLS. This is required to allow the User Equipment and API Server to exchange secrets which can be used to validate future interactions. The API MUST ensure the integrity of this information, as well as its confidentiality.

An attacker with access to this information might be able to masquerade as a specific User Equipment when interacting with the API, which could then allow them to masquerade as that User Equipment when interacting with the User Portal. This could give them the ability to determine whether the User Equipment has accessed the portal, or deny the User Equipment service by ending their session using mechanisms provided by the User Portal, or consume that User Equipment's quota. An attacker with the ability to modify the information could deny service to the User Equipment, or cause them to appear as a different User Equipment.

### 7.4. Risks Associated with the Signaling Protocol

If a Signaling Protocol is implemented, it may be possible for any user on the Internet to send signals in attempt to cause the receiving equipment to communicate with the Captive Portal API. This has been considered, and implementations may address it in the following ways:

- \* The signal only signals to the User Equipment to query the API. It does not carry any information which may mislead or misdirect the User Equipment.
- \* Even when responding to the signal, the User Equipment securely authenticates with API Servers.
- \* Accesses to the Captive Portal API are rate-limited, reducing the impact of an attack attempting to generate excessive load on either User Equipment or API. Note that because there is only one type of signal and one type of API request in response to the signal, this rate-limiting will not cause loss of signalling information.



### 7.5. User Options

The Captive Portal Signal could signal to the User Equipment that it is being held captive. There is no requirement that the User Equipment do something about this. Devices MAY permit users to disable automatic reaction to Captive Portal Signals indications for privacy reasons. However, there would be the trade-off that the user doesn't get notified when network access is restricted. Hence, end-user devices MAY allow users to manually control captive portal interactions, possibly on the granularity of Provisioning Domains.

### 7.6. Privacy

Section 3 describes a mechanism by which all components within the Captive Portal are designed to use the same identifier to uniquely identify the User Equipment. This identifier could be abused to track the user. Implementers and designers of Captive Portals should take care to ensure that identifiers, if stored, are stored securely. Likewise, if any component communicates the identifier over the network, it should ensure the confidentiality of the identifier on the wire by using encryption such as TLS.

There are benefits to choosing mutable anonymous identifiers. For example, User Equipment could cycle through multiple identifiers to help prevent long-term tracking. However, if the components of the network use an internal mapping to map the identity to a stable, long-term value in order to deal with changing identifiers, they need to treat that value as sensitive information: an attacker could use it to tie traffic back to the originating User Equipment, despite the User Equipment having changed identifiers.

## 8. References

### 8.1. 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>>.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <<https://www.rfc-editor.org/info/rfc2818>>.

- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <<https://www.rfc-editor.org/info/rfc7556>>.
- [RFC7710bis] Kumari, W. and E. Kline, "Captive-Portal Identification in DHCP / RA", Work in Progress, Internet-Draft, draft-ietf-capport-rfc7710bis-01, 12 January 2020, <<https://tools.ietf.org/html/draft-ietf-capport-rfc7710bis-01>>.
- [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>>.

## 8.2. Informative References

- [I-D.ietf-capport-api] Pauly, T. and D. Thakore, "Captive Portal API", Work in Progress, Internet-Draft, draft-ietf-capport-api-06, 31 March 2020, <<https://tools.ietf.org/html/draft-ietf-capport-api-06>>.
- [I-D.pfister-capport-pvd] Pfister, P. and T. Pauly, "Using Provisioning Domains for Captive Portal Discovery", Work in Progress, Internet-Draft, draft-pfister-capport-pvd-00, 30 June 2018, <<http://www.ietf.org/internet-drafts/draft-pfister-capport-pvd-00.txt>>.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, DOI 10.17487/RFC4213, October 2005, <<https://www.rfc-editor.org/info/rfc4213>>.

[RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <<https://www.rfc-editor.org/info/rfc8499>>.

## Appendix A. Existing Captive Portal Detection Implementations

Operating systems and user applications may perform various tests when network connectivity is established to determine if the device is attached to a network with a captive portal present. A common method is to attempt to make a HTTP request to a known, vendor-hosted endpoint with a fixed response. Any other response is interpreted as a signal that a captive portal is present. This check is typically not secured with TLS, as a network with a captive portal may intercept the connection, leading to a host name mismatch. This has been referred to as a "canary" request because, like the canary in the coal mine, it can be the first sign that something is wrong.

Another test that can be performed is a DNS lookup to a known address with an expected answer. If the answer differs from the expected answer, the equipment detects that a captive portal is present. DNS queries over TCP or HTTPS are less likely to be modified than DNS queries over UDP due to the complexity of implementation.

The different tests may produce different conclusions, varying by whether or not the implementation treats both TCP and UDP traffic, and by which types of DNS are intercepted.

Malicious or misconfigured networks with a captive portal present may not intercept these canary requests and choose to pass them through or decide to impersonate, leading to the device having a false negative.

## Authors' Addresses

Kyle Larose  
Agilicus

Email: [kyle@agilicus.com](mailto:kyle@agilicus.com)

David Dolson

Email: [ddolson@acm.org](mailto:ddolson@acm.org)

Heng Liu  
Google

Email: [liucougar@google.com](mailto:liucougar@google.com)

Network Working Group  
Internet-Draft  
Obsoletes: 7710 (if approved)  
Updates: 3679 (if approved)  
Intended status: Standards Track  
Expires: January 14, 2021

W. Kumari  
Google  
E. Kline  
Loon  
July 13, 2020

Captive-Portal Identification in DHCP / RA  
draft-ietf-capport-rfc7710bis-11

Abstract

In many environments offering short-term or temporary Internet access (such as coffee shops), it is common to start new connections in a captive portal mode. This highly restricts what the user can do until the user has satisfied the captive portal conditions.

This document describes a DHCPv4 and DHCPv6 option and a Router Advertisement (RA) option to inform clients that they are behind some sort of captive portal enforcement device, and that they will need to satisfy the Captive Portal conditions to get Internet access. It is not a full solution to address all of the issues that clients may have with captive portals; it is designed to be one component of a standardized approach for hosts to interact with such portals. While this document defines how the network operator may convey the captive portal API endpoint to hosts, the specific methods of satisfying and interacting with the captive portal are out of scope of this document.

This document replaces [RFC7710]. [RFC7710] used DHCP code point 160. Due to a conflict, this document specifies 114. Consequently, this document also updates [RFC3679].

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 January 14, 2021.

## Copyright Notice

Copyright (c) 2020 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 Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
1.1. Requirements Notation . . . . .	3
2. The Captive-Portal Option . . . . .	3
2.1. IPv4 DHCP Option . . . . .	4
2.2. IPv6 DHCP Option . . . . .	5
2.3. The Captive-Portal IPv6 RA Option . . . . .	5
3. Precedence of API URIs . . . . .	6
4. IANA Considerations . . . . .	6
4.1. Captive Portal Unrestricted Identifier . . . . .	7
4.2. BOOTP Vendor Extensions and DHCP Options Code Change . .	7
4.3. Update DHCPv6 and IPv6 ND Options Registries . . . . .	7
5. Security Considerations . . . . .	8
6. Acknowledgements . . . . .	9
7. References . . . . .	9
7.1. Normative References . . . . .	9
7.2. Informative References . . . . .	10
Appendix A. Changes / Author Notes. . . . .	11
Appendix B. Changes from RFC 7710 . . . . .	12
Appendix C. Observations From IETF 106 Network Experiment . . .	12
Authors' Addresses . . . . .	12

## 1. Introduction

In many environments, users need to connect to a captive portal device and agree to an Acceptable Use Policy (AUP) and / or provide billing information before they can access the Internet. Regardless of how that mechanism operates, this document provides functionality to allow the client to know when it is behind a captive portal and how to contact it.

In order to present users with the payment or AUP pages, presently a captive portal enforcement device has to intercept the user's connections and redirect the user to a captive portal server, using methods that are very similar to man-in-the-middle (MITM) attacks. As increasing focus is placed on security, and end nodes adopt a more secure stance, these interception techniques will become less effective and/or more intrusive.

This document describes a DHCPv4 [RFC2131] and DHCPv6 [RFC8415] option (Captive-Portal) and an IPv6 Router Advertisement (RA) [RFC4861] option that informs clients that they are behind a captive portal enforcement device and the API endpoint that the host can contact for more information.

This document replaces RFC 7710 [RFC7710].

### 1.1. Requirements Notation

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. The Captive-Portal Option

The Captive-Portal DHCP / RA Option informs the client that it may be behind a captive portal and provides the URI to access an API as defined by [draft-ietf-capport-api]. This is primarily intended to improve the user experience by showing the user the captive portal information faster and more reliably. Note that, for the foreseeable future, captive portals will still need to implement interception techniques to serve legacy clients, and clients will need to perform probing to detect captive portals"; nonetheless, the mechanism provided by this document provides a more reliable and performant way to do so, and is therefore the preferred mechanism for captive portal detection.

Clients that support the Captive Portal DHCP option SHOULD include the option in the Parameter Request List in DHCPREQUEST messages. DHCP servers MAY send the Captive Portal option without any explicit request.

In order to support multiple "classes" of clients (e.g. IPv4 only, IPv6 only with DHCPv6 ([RFC8415]), and IPv6 only with RA) the captive network can provision the client with the URI via multiple methods (IPv4 DHCP, IPv6 DHCP, and IPv6 RA). The captive portal operator SHOULD ensure that the URIs provisioned by each method are identical

to reduce the chance of operational problems. As the maximum length of the URI that can be carried in IPv4 DHCP is 255 bytes, URIs longer than this SHOULD NOT be provisioned by any of the IPv6 options described in this document. In IPv6-only environments this restriction can be relaxed.

In all variants of this option, the URI MUST be that of the captive portal API endpoint [draft-ietf-capport-api].

A captive portal MAY do content negotiation ([RFC7231] section 3.4) and attempt to redirect clients querying without an explicit indication of support for the captive portal API content type (i.e. without application/capport+json listed explicitly anywhere within an Accept header vis. [RFC7231] section 5.3). In so doing, the captive portal SHOULD redirect the client to the value associated with the "user-portal-url" API key. When performing such content negotiation ([RFC7231] Section 3.4), implementors of captive portals need to keep in mind that such responses might be cached, and therefore SHOULD include an appropriate Vary header field ([RFC7231] Section 7.1.4) or set the Cache-Control header field in any responses to "private", or a more restrictive value such as "no-store" [RFC7234] Section 5.2.2.3).

The URI SHOULD NOT contain an IP address literal. Exceptions to this might include networks with only one operational IP address family where DNS is either not available or not fully functional until the captive portal has been satisfied. Use of iPAddress certificates ([RFC3779]) adds considerations that are out of scope for this document.

Networks with no captive portals may explicitly indicate this condition by using this option with the IANA-assigned URI for this purpose. Clients observing the URI value "urn:ietf:params:capport:unrestricted" may forego time-consuming forms of captive portal detection.

## 2.1. IPv4 DHCP Option

The format of the IPv4 Captive-Portal DHCP option is shown below.

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-----+-----+-----+-----+-----+-----+-----+-----+
    | Code           | Len           | URI (variable length) ...    |
    +-----+-----+-----+-----+-----+-----+-----+-----+
    |                                     ...URI continued...      |
    |                                     ...                        |
    +-----+-----+-----+-----+-----+-----+-----+-----+

```



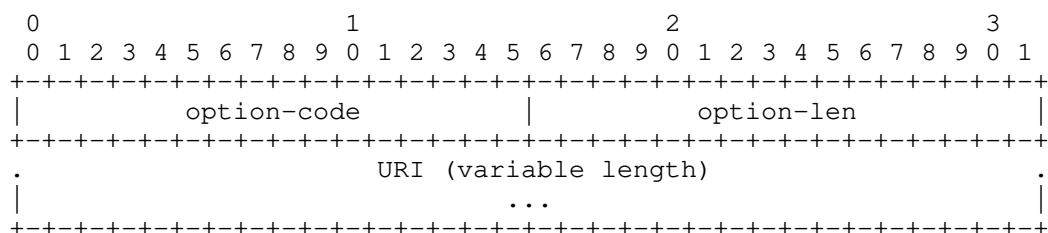
- o Code: The Captive-Portal DHCPv4 Option (114) (one octet)
- o Len: The length (one octet), in octets, of the URI.
- o URI: The URI for the captive portal API endpoint to which the user should connect (encoded following the rules in [RFC3986]).

See [RFC2132], Section 2 for more on the format of IPv4 DHCP options.

Note that the URI parameter is not null terminated.

## 2.2. IPv6 DHCP Option

The format of the IPv6 Captive-Portal DHCP option is shown below.



- o option-code: The Captive-Portal DHCPv6Option (103) (two octets)
- o option-len: The unsigned 16-bit length, in octets, of the URI.
- o URI: The URI for the captive portal API endpoint to which the user should connect (encoded following the rules in [RFC3986]).

See [RFC7227], Section 5.7 for more examples of DHCP Options with URIs. See [RFC8415], Section 21.1 for more on the format of IPv6 DHCP options.

Note that the URI parameter is not null terminated.

As the maximum length of the URI that can be carried in IPv4 DHCP is 255 bytes, URIs longer than this SHOULD NOT be provisioned via IPv6 DHCP options.

## 2.3. The Captive-Portal IPv6 RA Option

This section describes the Captive-Portal Router Advertisement option.

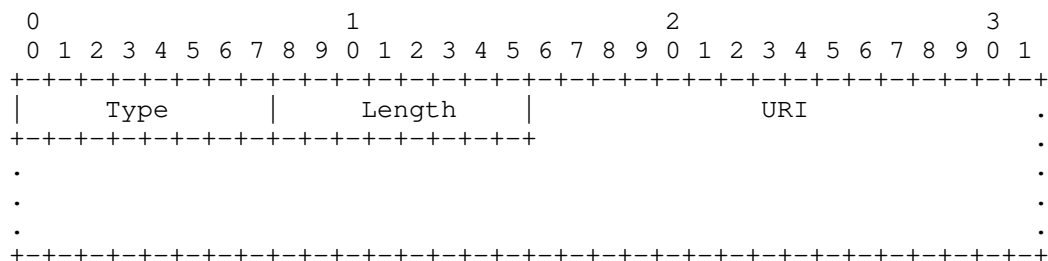


Figure 2: Captive-Portal RA Option Format

Type 37

Length 8-bit unsigned integer. The length of the option (including the Type and Length fields) in units of 8 bytes.

URI The URI for the captive portal API endpoint to which the user should connect. This MUST be padded with NUL (0x00) to make the total option length (including the Type and Length fields) a multiple of 8 bytes.

Note that the URI parameter is not guaranteed to be null terminated.

As the maximum length of the URI that can be carried in IPv4 DHCP is 255 bytes, URIs longer than this SHOULD NOT be provisioned via IPv6 RA options.

### 3. Precedence of API URIs

A device may learn about Captive Portal API URIs through more than one of (or indeed all of) the above options. Implementations can select their own precedence order (e.g., prefer one of the IPv6 options before the DHCPv4 option, or vice versa, et cetera).

If the URIs learned via more than one option described in Section 2 are not all identical, this condition should be logged for the device owner or administrator; it is a network configuration error if the learned URIs are not all identical.

### 4. IANA Considerations

This document requests one new IETF URN protocol parameter ([RFC3553]) entry. This document also requests a reallocation of DHCPv4 option codes (see Appendix C for background).

Thanks IANA!

#### 4.1. Captive Portal Unrestricted Identifier

This document registers a new entry under the IETF URN Sub-namespace for Registered Protocol Parameter Identifiers defined in [RFC3553]:

Registered Parameter Identifier: capport:unrestricted

Reference: RFC TBD (this document)

IANA Registry Reference: RFC TBD (this document)

Only one value is defined (see URN above). No hierarchy is defined and therefore no sub-namespace registrations are possible.

#### 4.2. BOOTP Vendor Extensions and DHCP Options Code Change

[ RFC Ed: Please remove before publication: RFC7710 uses DHCP Code 160 -- unfortunately, it was discovered that this option code is already widely used by Polycom (see appendix). Option 114 (URL) is currently assigned to Apple (RFC3679, Section 3.2.3 - Contact: Dieter Siegmund, dieter@apple.com - Reason to recover: Never published in an RFC) Tommy Pauly (Apple) and Dieter Siegmund confirm that this codepoint hasn't been used, and Apple is willing to relinquish it for use in CAPPORT. Please see thread: [https://mailarchive.ietf.org/arch/msg/captive-portals/TmqQz6Ma\\_fznD3XbhwkH9m2dB28](https://mailarchive.ietf.org/arch/msg/captive-portals/TmqQz6Ma_fznD3XbhwkH9m2dB28) for more background. ]

The IANA is requested to update the "BOOTP Vendor Extensions and DHCP Options" registry (<https://www.iana.org/assignments/bootp-dhcp-parameters/bootp-dhcp-parameters.xhtml>) as follows.

Tag: 114  
Name: DHCP Captive-Portal  
Data Length: N  
Meaning: DHCP Captive-Portal  
Reference: [THIS-RFC]

Tag: 160  
Name: Unassigned  
Data Length:  
Meaning: Previously assigned by RFC7710; known to also be used by Polycom.  
Reference: [THIS-RFC][RFC7710]

#### 4.3. Update DHCPv6 and IPv6 ND Options Registries

This document requests that the DHCPv6 and IPv6 ND options previously registered in [RFC7710] be updated to reference this document.

## 5. Security Considerations

By removing or reducing the need for captive portals to perform MITM hijacking, this mechanism improves security by making the portal and its actions visible, rather than hidden, and reduces the likelihood that users will disable useful security safeguards like DNSSEC validation, VPNs, etc in order to interact with the captive portal. In addition, because the system knows that it is behind a captive portal, it can know not to send cookies, credentials, etc. By handing out a URI which is protected with TLS, the captive portal operator can attempt to reassure the user that the captive portal is not malicious.

Clients processing these options SHOULD validate that the option's contents conform to the validation requirements for URIs, including [RFC3986].

Each of the options described in this document is presented to a node using the same protocols used to provision other information critical to the node's successful configuration on a network. The security considerations applicable to each of these provisioning mechanisms also apply when the node is attempting to learn the information conveyed in these options. In the absence of security measures like RA Guard ([RFC6105], [RFC7113]) or DHCP Shield [RFC7610], an attacker could inject, modify, or block DHCP messages or RAs.

An attacker with the ability to inject DHCP messages or RAs could include an option from this document to force users to contact an address of his choosing. As an attacker with this capability could simply list themselves as the default gateway (and so intercept all the victim's traffic); this does not provide them with significantly more capabilities, but because this document removes the need for interception, the attacker may have an easier time performing the attack.

However, as the operating systems and application(s) that make use of this information know that they are connecting to a captive portal device (as opposed to intercepted connections where the OS/application may not know that they are connecting to a captive portal or hostile device) they can render the page in a sandboxed environment and take other precautions, such as clearly labeling the page as untrusted. The means of sandboxing and user interface presenting this information is not covered in this document - by its nature it is implementation specific and best left to the application and user interface designers.

Devices and systems that automatically connect to an open network could potentially be tracked using the techniques described in this

document (forcing the user to continually re-satisfy the Captive Portal conditions, or exposing their browser fingerprint). However, similar tracking can already be performed with the presently common captive portal mechanisms, so this technique does not give the attackers more capabilities.

Captive portals are increasingly hijacking TLS connections to force browsers to talk to the portal. Providing the portal's URI via a DHCP or RA option is a cleaner technique, and reduces user expectations of being hijacked - this may improve security by making users more reluctant to accept TLS hijacking, which can be performed from beyond the network associated with the captive portal.

## 6. Acknowledgements

This document is a -bis of RFC7710. Thanks to all of the original authors (Warren Kumari, Olafur Gudmundsson, Paul Ebersman, Steve Sheng), and original contributors.

Also thanks to the CAPPORT WG for all of the discussion and improvements including contributions and review from Joe Clarke, Lorenzo Colitti, Dave Dolson, Hans Kuhn, Kyle Larose, Clemens Schimpe, Martin Thomson, Michael Richardson, Remi Nguyen Van, Subash Tirupachur Comerica, Bernie Volz, and Tommy Pauly.

## 7. References

### 7.1. 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>>.
- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <<https://www.rfc-editor.org/info/rfc2131>>.
- [RFC2132] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132, DOI 10.17487/RFC2132, March 1997, <<https://www.rfc-editor.org/info/rfc2132>>.
- [RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An IETF URN Sub-namespace for Registered Protocol Parameters", BCP 73, RFC 3553, DOI 10.17487/RFC3553, June 2003, <<https://www.rfc-editor.org/info/rfc3553>>.

- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [RFC7227] Hankins, D., Mrugalski, T., Siodelski, M., Jiang, S., and S. Krishnan, "Guidelines for Creating New DHCPv6 Options", BCP 187, RFC 7227, DOI 10.17487/RFC7227, May 2014, <<https://www.rfc-editor.org/info/rfc7227>>.
- [RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <<https://www.rfc-editor.org/info/rfc7231>>.
- [RFC7234] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Caching", RFC 7234, DOI 10.17487/RFC7234, June 2014, <<https://www.rfc-editor.org/info/rfc7234>>.
- [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>>.
- [RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <<https://www.rfc-editor.org/info/rfc8415>>.

## 7.2. Informative References

- [RFC3679] Droms, R., "Unused Dynamic Host Configuration Protocol (DHCP) Option Codes", RFC 3679, DOI 10.17487/RFC3679, January 2004, <<https://www.rfc-editor.org/info/rfc3679>>.
- [RFC3779] Lynn, C., Kent, S., and K. Seo, "X.509 Extensions for IP Addresses and AS Identifiers", RFC 3779, DOI 10.17487/RFC3779, June 2004, <<https://www.rfc-editor.org/info/rfc3779>>.

- [RFC6105] Levy-Abegnoli, E., Van de Velde, G., Popoviciu, C., and J. Mohacsi, "IPv6 Router Advertisement Guard", RFC 6105, DOI 10.17487/RFC6105, February 2011, <<https://www.rfc-editor.org/info/rfc6105>>.
- [RFC7113] Gont, F., "Implementation Advice for IPv6 Router Advertisement Guard (RA-Guard)", RFC 7113, DOI 10.17487/RFC7113, February 2014, <<https://www.rfc-editor.org/info/rfc7113>>.
- [RFC7610] Gont, F., Liu, W., and G. Van de Velde, "DHCPv6-Shield: Protecting against Rogue DHCPv6 Servers", BCP 199, RFC 7610, DOI 10.17487/RFC7610, August 2015, <<https://www.rfc-editor.org/info/rfc7610>>.
- [RFC7710] Kumari, W., Gudmundsson, O., Ebersman, P., and S. Sheng, "Captive-Portal Identification Using DHCP or Router Advertisements (RAs)", RFC 7710, DOI 10.17487/RFC7710, December 2015, <<https://www.rfc-editor.org/info/rfc7710>>.

### 7.3. URIs

- [1] <https://tickets.meeting.ietf.org/wiki/IETF106network#Experiments>
- [2] <https://tickets.meeting.ietf.org/wiki/CAPPORT>
- [3] <https://community.polycom.com/t5/VoIP-SIP-Phones/DHCP-Standardization-160-vs-66/td-p/72577>

### Appendix A. Changes / Author Notes.

[RFC Editor: Please remove this section before publication ]

From initial to -00.

- o Import of RFC7710.

From -00 to -01.

- o Remove link-relation text.
- o Clarify option should be in DHCPREQUEST parameter list.
- o Uppercase some SHOULDs.

## Appendix B. Changes from RFC 7710

This document incorporates the following changes from [RFC7710].

1. Clarify that IP string literals are NOT RECOMMENDED.
2. Clarify that the option URI MUST be that of the captive portal API endpoint.
3. Clarify that captive portals MAY do content negotiation.
4. Added text about Captive Portal API URI precedence in the event of a network configuration error.
5. Added urn:ietf:params:capport:unrestricted URN.
6. Notes that the DHCPv4 Option Code changed from 160 to 114.

## Appendix C. Observations From IETF 106 Network Experiment

During IETF 106 in Singapore an experiment [1] enabling Captive Portal API compatible clients to discover a venue-info-url (see experiment description [2] for more detail) revealed that some Polycom devices on the same network made use of DHCPv4 option code 160 for other purposes [3].

The presence of DHCPv4 Option code 160 holding a value indicating the Captive Portal API URL caused these devices to not function as desired. For this reason, this document requests IANA deprecate option code 160 and reallocate different value to be used for the Captive Portal API URL.

## Authors' Addresses

Warren Kumari  
Google  
1600 Amphitheatre Parkway  
Mountain View, CA 94043  
US

Email: warren@kumari.net



Erik Kline  
Loon  
1600 Amphitheatre Parkway  
Mountain View, CA 94043  
US

Email: [ek@loon.com](mailto:ek@loon.com)