

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.

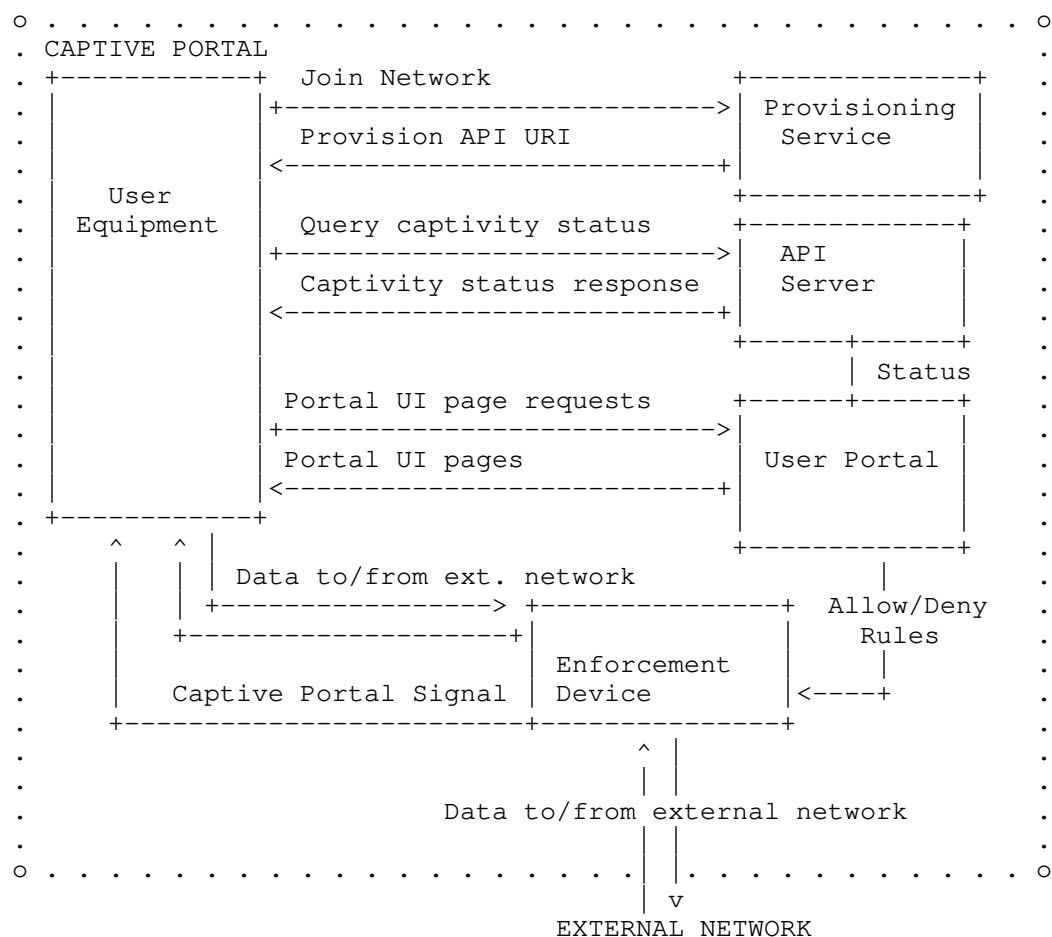


Figure 1: Captive Portal Architecture Component Diagram

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 as 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

David Dolson

Email: ddolson@acm.org

Heng Liu
Google

Email: liucougar@google.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 3, 2020

P. Pfister
E. Vyncke
Cisco
T. Pauly
Apple Inc.
D. Schinazi
Google LLC
W. Shao
Cisco
January 31, 2020

Discovering Provisioning Domain Names and Data
draft-ietf-intarea-provisioning-domains-11

Abstract

Provisioning Domains (PvDs) are defined as consistent sets of network configuration information. This allows hosts to manage connections to multiple networks and interfaces simultaneously, such as when a home router provides connectivity through both a broadband and cellular network provider.

This document defines a mechanism for explicitly identifying PvDs through a Router Advertisement (RA) option. This RA option announces a PvD identifier, which hosts can compare to differentiate between PvDs. The option can directly carry some information about a PvD and can optionally point to additional PvD information that can be retrieved using HTTP over TLS.

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 August 3, 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	3
1.1. Specification of Requirements	4
2. Terminology	4
3. Provisioning Domain Identification using Router Advertisements	5
3.1. PvD ID Option for Router Advertisements	5
3.2. Router Behavior	8
3.3. Non-PvD-aware Host Behavior	9
3.4. PvD-aware Host Behavior	9
3.4.1. DHCPv6 configuration association	10
3.4.2. DHCPv4 configuration association	11
3.4.3. Connection Sharing by the Host	11
3.4.4. Usage of DNS Servers	12
4. Provisioning Domain Additional Information	13
4.1. Retrieving the PvD Additional Information	13
4.2. Operational Consideration to Providing the PvD Additional Information	16
4.3. PvD Additional Information Format	16
4.3.1. Example	18
4.4. Detecting misconfiguration and misuse	18
5. Operational Considerations	19
5.1. Exposing Extra RA Options to PvD-Aware Hosts	19
5.2. Different RAs for PvD-Aware and Non-PvD-Aware Hosts	19
5.3. Enabling Multi-homing for PvD-Aware Hosts	21
5.4. Providing Additional Information to PvD-Aware Hosts	22
6. Security Considerations	23
7. Privacy Considerations	24
8. IANA Considerations	25
8.1. New entry in the Well-Known URIs Registry	26
8.2. Additional Information PvD Keys Registry	26
8.3. PvD Option Flags Registry	26

8.4. PvD JSON Media Type Registration	27
9. Acknowledgments	28
10. References	28
10.1. Normative References	28
10.2. Informative References	30
Authors' Addresses	32

1. Introduction

Provisioning Domains (PvDs) are defined in [RFC7556] as consistent sets of network configuration information. This information includes properties that are traditionally associated with a single networking interface, such as source addresses, DNS configuration, proxy configuration, and gateway addresses.

Clients that are aware of PvDs can take advantage of multiple network interfaces simultaneously. This enables using two PvDs in parallel for separate connections or for multi-path transports.

While most PvDs today are discovered implicitly (such as by receiving information via Router Advertisements from a router on a network that a client host directly connects to), [RFC7556] also defines the notion of Explicit PvDs. IPsec Virtual Private Networks are considered Explicit PvDs, but Explicit PvDs can also be discovered via the local network router. Discovering Explicit PvDs allows two key advancements in managing multiple PvDs:

1. The ability to discover and use multiple PvDs on a single interface, such as when a local router can provide connectivity to two different Internet Service Providers.
2. The ability to associate additional information about PvDs to describe the properties of the network.

While [RFC7556] defines the concept of Explicit PvDs, it does not define the mechanism for discovering multiple Explicit PvDs on a single network and their additional information.

This document specifies a way to identify PvDs with Fully Qualified Domain Names (FQDN), called PvD IDs. Those identifiers are advertised in a new Router Advertisement (RA) [RFC4861] option called the PvD ID Router Advertisement option which, when present, associates the PvD ID with all the information present in the Router Advertisement as well as any configuration object, such as addresses, derived from it. The PVD ID Router Advertisement option may also contain a set of other RA options, along with an optional inner Router Advertisement message header. These options and optional

inner header are only visible to 'PvD-aware' hosts, allowing such hosts to have a specialized view of the network configuration.

Since PvD IDs are used to identify different ways to access the internet, multiple PvDs (with different PvD IDs) can be provisioned on a single host interface. Similarly, the same PvD ID could be used on different interfaces of a host in order to inform that those PvDs ultimately provide equivalent services.

This document also introduces a mechanism for hosts to retrieve optional additional information related to a specific PvD by means of an HTTP over TLS query using a URI derived from the PvD ID. The retrieved JSON object contains additional information that would typically be considered too large to be directly included in the Router Advertisement, but might be considered useful to the applications, or even sometimes users, when choosing which PvD should be used.

For example, if Alice has both a cellular network provider and a broadband provider in her home, her PvD-aware devices and applications would be aware of both available uplinks. These applications could fail-over between these networks, or run connections over both (potentially using multi-path transports). Applications could also select specific uplinks based on the properties of the network; for example, if the cellular network provides free high-quality video streaming, a video-streaming application could select that network while most of the other traffic on Alice's device uses the broadband provider.

1.1. Specification of Requirements

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. Terminology

This document uses the following terminology:

Provisioning Domain (PvD): A set of network configuration information; for more information, see [RFC7556].

PvD ID: A Fully Qualified Domain Name (FQDN) used to identify a PvD.

Explicit PvD: A PvD uniquely identified with a PvD ID. For more information, see [RFC7556].

Implicit PvD: A PvD that, in the absence of a PvD ID, is identified by the host interface to which it is attached and the address of the advertising router. See also [RFC7556].

PvD-aware host: A host that supports the association of network configuration information into PvDs and the use of these PvDs as described in this document. Also named PvD-aware node in [RFC7556].

3. Provisioning Domain Identification using Router Advertisements

Explicit PvDs are identified by a PvD ID. The PvD ID is a Fully Qualified Domain Name (FQDN) that identifies the network operator. Network operators **MUST** use names that they own or manage to avoid naming conflicts. The same PvD ID **MAY** be used in several access networks when they ultimately provide identical services (e.g., in all home networks subscribed to the same service); else, the PvD ID **MUST** be different to follow Section 2.4 of [RFC7556].

3.1. PvD ID Option for Router Advertisements

This document introduces a Router Advertisement (RA) option called the PvD Option. It is used to convey the FQDN identifying a given PvD (see Figure 1), bind the PvD ID with configuration information received over DHCPv4 (see Section 3.4.2), enable the use of HTTP over TLS to retrieve the PvD Additional Information JSON object (see Section 4), as well as contain any other RA options which would otherwise be valid in the RA.

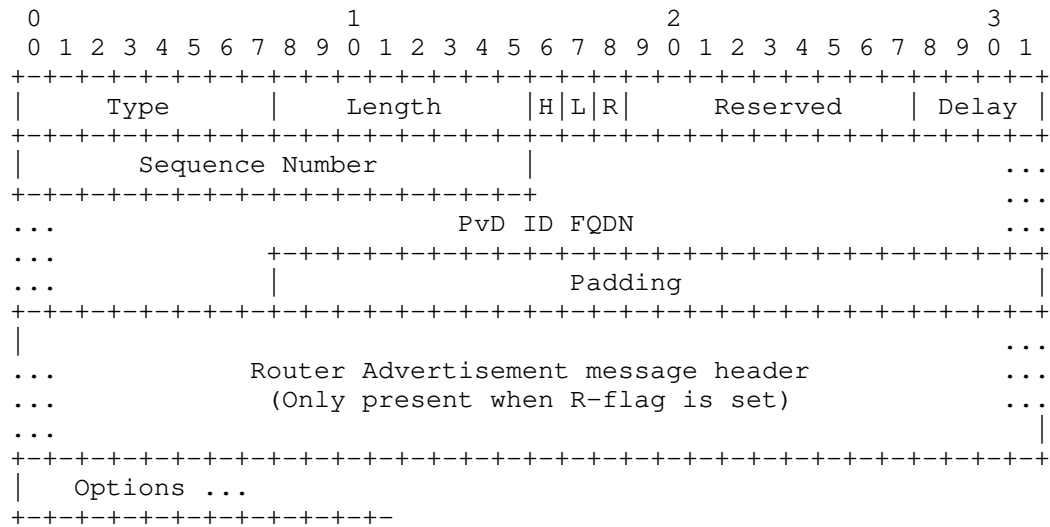


Figure 1: PvD ID Router Advertisements Option Format

Type: (8 bits) Set to 21.

Length: (8 bits) The length of the option in units of 8 octets, including the Type and Length fields, the Router Advertisement message header, if any, as well as the RA options that are included within the PvD Option.

H-flag: (1 bit) 'HTTP' flag stating whether some PvD Additional Information is made available through HTTP over TLS, as described in Section 4.

L-flag: (1 bit) 'Legacy' flag stating whether the PvD is associated with IPv4 information assigned using DHCPv4 (see Section 3.4.2).

R-flag: (1 bit) 'Router Advertisement' flag stating whether the PvD Option header is followed (right after padding to the next 64 bits boundary) by a Router Advertisement message header (see section 4.2 of [RFC4861]). The usage of the inner message header is described in Section 3.4.

Reserved: (13 bits) Reserved for later use. It MUST be set to zero by the sender and ignored by the receiver.

Delay: (4 bits) Unsigned integer used to delay HTTP GET queries from hosts by a randomized backoff (see Section 4.1). If the H-flag is not set, senders SHOULD set the delay to zero, and receivers SHOULD ignore the value.

Sequence Number: (16 bits) Sequence number for the PvD Additional Information, as described in Section 4. If the H-flag is not set, senders SHOULD set the Sequence Number to zero, and receivers SHOULD ignore the value.

PvD ID FQDN: The FQDN used as PvD ID encoded in DNS format, as described in Section 3.1 of [RFC1035]. Domain name compression described in Section 4.1.4 of [RFC1035] MUST NOT be used.

Padding: Zero or more padding octets to the next 8 octet boundary (see Section 4.6 of [RFC4861]). It MUST be set to zero by the sender, and ignored by the receiver.

RA message header: (16 octets) When the R-flag is set, a full Router Advertisement message header as specified in [RFC4861]. The sender MUST set the 'Type' to 134, the value for "Router Advertisement", and set the 'Code' to 0. Receivers MUST ignore both of these fields. The 'Checksum' MUST be set to 0 by the sender; non-zero checksums MUST be ignored by the receiver without causing the processing of the message to fail. All other fields are to be set and parsed as specified in [RFC4861] or any updating documents.

Options: Zero or more RA options that would otherwise be valid as part of the Router Advertisement main body, but are instead included in the PvD Option so as to be ignored by hosts that are not PvD-aware.

Figure 2 shows an example of a PvD Option with "example.org" as the PvD ID FQDN and including both a Recursive DNS Server (RDNSS) option and a prefix information option. It has a Sequence Number of 123, and indicates the presence of additional information that is expected to be fetched with a delay factor of 1.

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								
Type: 21										Length: 12										1 0 0 Reserved										Delay:1									
Seq number: 123										7										e																			
x										a										m										p									
l										e										3										o									
r										g										0										0 (padding)									
0 (padding)										0 (padding)										0 (padding)										0 (padding)									
RDNSS option (RFC 8106) length: 5																														...									
...																														...									
...																																							
Prefix Information Option (RFC 4861) length: 4																														...									
...																																							
...																																							

Figure 2

3.2. Router Behavior

A router MAY send RAs containing one PvD Option, but MUST NOT include more than one PvD Option in each RA. The PvD Option MUST NOT contain further PvD Options.

The PvD Option MAY contain zero, one, or more RA options which would otherwise be valid as part of the same RA. Such options are processed by PvD-aware hosts, while ignored by other hosts as per section 4.2 of [RFC4861].

In order to provide multiple different PvDs, a router MUST send multiple RAs. RAs sent from different link-local source addresses establish distinct implicit PvDs, in the absence of a PvD Option. Explicit PvDs MAY share link-local source addresses with an Implicit PvD and any number of other Explicit PvDs.

In other words, different Explicit PvDs MAY be advertised with RAs using the same link-local source address; but different Implicit PvDs, advertised by different RAs, MUST use different link-local addresses because these Implicit PvDs are identified by the source addresses of the RAs. If a link-local address on the router is

changed, then any new RA will be interpreted as a different Implicit PvD by PvD-aware hosts.

As specified in [RFC4861] and [RFC6980], when the set of options causes the size of an advertisement to exceed the link MTU, multiple router advertisements MUST be sent to avoid fragmentation, each containing a subset of the options. In such cases, the PvD Option header (i.e., all fields except the 'Options' field) MUST be repeated in all the transmitted RAs. The options within the 'Options' field, MAY be transmitted only once, included in one of the transmitted PvD Options.

3.3. Non-PvD-aware Host Behavior

As the PvD Option has a new option code, non-PvD-aware hosts will simply ignore the PvD Option and all the options it contains (see section 4.2 of [RFC4861]. This ensures the backward compatibility required in Section 3.3 of [RFC7556]. This behavior allows for a mixed-mode network where a mix of PvD-aware and non-PvD-aware hosts coexist.

3.4. PvD-aware Host Behavior

Hosts MUST associate received RAs and included configuration information (e.g., Router Valid Lifetime, Prefix Information [RFC4861], Recursive DNS Server [RFC8106], Routing Information [RFC4191] options) with the Explicit PvD identified by the first PvD Option present in the received RA, if any, or with the Implicit PvD identified by the host interface and the source address of the received RA otherwise. If an RA message header is present both within the PvD Option and outside it, the header within the PvD Option takes precedence.

In case multiple PvD Options are found in a given RA, hosts MUST ignore all but the first PvD Option.

If a host receives PvD Options flags that it does not recognize (currently in the Reserved field), it MUST ignore these flags.

Similarly, hosts MUST associate all network configuration objects (e.g., default routers, addresses, more specific routes, DNS Recursive Resolvers) with the PvD associated with the RA that provisioned the object. For example, addresses that are generated using a received Prefix Information option (PIO) are associated with the PvD of the last received RA which included the given PIO.

PvD IDs MUST be compared in a case-insensitive manner as defined by [RFC4343]. For example, "pvd.example.com." or "PvD.Example.coM." would refer to the same PvD.

While performing PvD-specific operations such as resolving names, executing the default address selection algorithm [RFC6724] or executing the default router selection algorithm when forwarding packets ([RFC4861], [RFC4191] and [RFC8028]), hosts and applications MAY consider only the configuration associated with any non-empty subset of PvDs. For example, a host MAY associate a given process with a specific PvD, or a specific set of PvDs, while associating another process with another PvD. A PvD-aware application might also be able to select, on a per-connection basis, which PvDs should be used. In particular, constrained devices such as small battery operated devices (e.g., IoT), or devices with limited CPU or memory resources may purposefully use a single PvD while ignoring some received RAs containing different PvD IDs.

The way an application expresses its desire to use a given PvD, or a set of PvDs, or the way this selection is enforced, is out of the scope of this document. Useful insights about these considerations can be found in [I-D.kline-mif-mpvd-api-reqs].

3.4.1. DHCPv6 configuration association

When a host retrieves stateless configuration elements using DHCPv6 (e.g., DNS recursive resolvers or DNS domain search lists [RFC3646]), they MUST be associated with all the explicit and implicit PvDs received on the same interface and contained in a RA with the O-flag set [RFC4861].

When a host retrieves stateful assignments using DHCPv6, such assignments MUST be associated with the received PvD which was received with RAs with the M-flag set and including a matching PIO. A PIO is considered to match a DHCPv6 assignment when the IPv6 prefix from the PIO includes the assignment from DHCPv6. For example, if a PvD's associated PIO defines the prefix 2001:db8:cafe::/64, a DHCPv6 IA_NA message that assigns the address 2001:db8:cafe::1234:4567 would be considered to match.

In cases where an address would be assigned by DHCPv6 and no matching PvD could be found, hosts MAY associate the assigned address with any implicit PvD received on the same interface or to multiple implicit PvDs received on the same interface. This is intended to resolve backward compatibility issues with rare deployments choosing to assign addresses with DHCPv6 while not sending any matching PIO. Implementations are suggested to flag or log such scenarios as errors to help detect misconfigurations.

3.4.2. DHCPv4 configuration association

Associating DHCPv4 [RFC2131] configuration elements with Explicit PvDs allows hosts to treat a set of IPv4 and IPv6 configurations as a single PvD with shared properties. For example, consider a router that provides two different uplinks. One could be a broadband network that has data rate and streaming properties described in PvD additional information and that provides both IPv4 and IPv6 network access. The other could be a cellular network that provides only IPv6 network access, and uses NAT64 [RFC6146]. The broadband network can be represented by an Explicit PvD that points to the additional information, and also marks association with DHCPv4 information. The cellular network can be represented by a different Explicit PvD that is not associated with DHCPv4.

When a PvD-aware host retrieves configuration elements from DHCPv4, the information is associated either with a single Explicit PvD on that interface, or else with all Implicit PvDs on the same interface.

An Explicit PvD indicates its association with DHCPv4 information by setting the L-flag in the PvD RA Option. If there is exactly one Explicit PvD that sets this flag, hosts MUST associate the DHCPv4 information with that PvD. Multiple Explicit PvDs on the same interface marking this flag is a misconfiguration, and hosts SHOULD NOT associate the DHCPv4 information with any Explicit PvD in this case.

If no single Explicit PvD claims association with DHCPv4, the configuration elements coming from DHCPv4 MUST be associated with all Implicit PvDs identified by the interface on which the DHCPv4 transaction happened. This maintains existing host behavior.

3.4.3. Connection Sharing by the Host

The situation when a host shares connectivity from an upstream interface (e.g., cellular) to a downstream interface (e.g., Wi-Fi) is known as 'tethering'. Techniques such as ND-proxy [RFC4389], 64share [RFC7278] or prefix delegation (e.g., using DHCPv6-PD [RFC8415]) may be used for that purpose.

Whenever the RAs received from the upstream interface contain a PVD RA option, hosts that are sharing connectivity SHOULD include a PVD option within the RAs sent downstream with:

- o The same PVD-ID FQDN
- o The same H-flag, Delay and Sequence Number values

- o The L bit set whenever the host is sharing IPv4 connectivity received from the same upstream interface
- o The bits from the Reserved field set to 0

The values of the R-flag, Router Advertisement message header and Options field depend on whether the connectivity should be shared only with PvD-aware hosts or not (see Section 3.2). In particular, all options received within the upstream PvD Option and included in the downstream RA SHOULD be included in the downstream PvD Option.

3.4.4. Usage of DNS Servers

PvD-aware hosts can be provisioned with recursive DNS servers via RA options passed within an Explicit PvD, via RA options associated with an Implicit PvD, via DHCPv6 or DHCPv4, or from some other provisioning mechanism that creates an Explicit PvD (such as a VPN). In all of these cases, the recursive DNS server addresses SHOULD be associated with the corresponding PvD. Specifically, queries sent to a configured recursive DNS server SHOULD be sent from a local IP address that was provisioned for the PvD via RA or DHCP. Answers received from the DNS server SHOULD only be used on the same PvD.

PvD-aware applications will be able to select which PvD(s) to use for DNS resolution and connections, which allows them to effectively use multiple Explicit PvDs. In order to support non-PvD-aware applications, however, PvD-aware hosts SHOULD ensure that non-PvD-aware name resolution APIs like "getaddrinfo" only use resolvers from a single PvD for a given query. Handling DNS across PvDs is discussed in Section 5.2.1 of [RFC7556], and PvD APIs are discussed in Section 6 of [RFC7556].

Maintaining the correct usage of DNS within PvDs avoids various practical errors, such as:

- o A PvD associated with a VPN or otherwise private network may provide DNS answers that contain addresses inaccessible over another PvD. This includes the DNS queries to retrieve PvD additional information, which could otherwise send identifying information to the recursive DNS system (see Section 4.1).
- o A PvD that uses a NAT64 [RFC6146] and DNS64 [RFC6147] will synthesize IPv6 addresses in DNS answers that are not globally routable, and would be invalid on other PvDs. Conversely, an IPv4 address resolved via DNS on another PvD cannot be directly used on a NAT64 network.

4. Provisioning Domain Additional Information

Additional information about the network characteristics can be retrieved based on the PvD ID. This set of information is called PvD Additional Information, and is encoded as a JSON object [RFC8259]. This JSON object is restricted to the I-JSON profile, as defined in [RFC7493].

The purpose of this JSON object is to provide additional information to applications on a client host about the connectivity that is provided using a given interface and source address. It typically includes data that would be considered too large, or not critical enough, to be provided within an RA option. The information contained in this object MAY be used by the operating system, network libraries, applications, or users, in order to decide which set of PvDs should be used for which connection, as described in Section 3.4.

The additional information related to a PvD is specifically intended to be optional, and is targeted at optimizing or informing the behavior of user-facing hosts. This information can be extended to provide hints for host system behavior (such as captive portal or walled-garden PvD detection) or application behavior (describing application-specific services offered on a given PvD). This content may not be appropriate for light-weight Internet of Things (IoT) devices. IoT devices might need only a subset of the information, and would in some cases prefer a smaller representation like CBOR ([RFC7049]). Delivering a reduced version of the PvD Additional Information designed for such devices is not defined in this document.

4.1. Retrieving the PvD Additional Information

When the H-flag of the PvD Option is set, hosts MAY attempt to retrieve the PvD Additional Information associated with a given PvD by performing an HTTP over TLS [RFC2818] GET query to `https://<PvD-ID>/well-known/pvd`. Inversely, hosts MUST NOT do so whenever the H-flag is not set.

Recommendations for how to use TLS securely can be found in [RFC7525].

When a host retrieves the PvD Additional Information, it MUST verify that the TLS server certificate is valid for the performed request; specifically, that a DNS-ID [RFC6125] on the certificate is equal to the PvD ID expressed as an FQDN. This validation indicates that the owner of the FQDN authorizes its use with the prefix advertised by

the router. If this validation fails, hosts MUST close the connection and treat the PvD as if it has no Additional Information.

HTTP requests and responses for PvD additional information use the "application/pvd+json" media type (see Section 8). Clients SHOULD include this media type as an Accept header field in their GET requests, and servers MUST mark this media type as their Content-Type header field in responses.

Note that the DNS name resolution of the PvD ID, any connections made for certificate validation (such as OCSP [RFC6960]), and the HTTP request itself MUST be performed using the considered PvD. In other words, the name resolution, PKI checks, source address selection, as well as the next-hop router selection MUST be performed while using exclusively the set of configuration information attached with the PvD, as defined in Section 3.4. In some cases, it may therefore be necessary to wait for an address to be available for use (e.g., once the Duplicate Address Detection or DHCPv6 processes are complete) before initiating the HTTP over TLS query. In order to address privacy concerns around linkability of the PvD HTTP connection with future user-initiated connections, if the host has a temporary address per [RFC4941] in this PvD, then it SHOULD use a temporary address to fetch the PvD Additional Information and MAY deprecate the used temporary address and generate a new temporary address afterward.

If the HTTP status of the answer is greater than or equal to 400 the host MUST close its connection and consider that there is no additional PvD information. If the HTTP status of the answer is between 300 and 399, inclusive, it MUST follow the redirection(s). If the HTTP status of the answer is between 200 and 299, inclusive, the response is expected to be a single JSON object.

After retrieval of the PvD Additional Information, hosts MUST remember the last Sequence Number value received in an RA including the same PvD ID. Whenever a new RA for the same PvD is received with a different Sequence Number value, or whenever the expiry date for the additional information is reached, hosts MUST deprecate the additional information and stop using it.

Hosts retrieving a new PvD Additional Information object MUST check for the presence and validity of the mandatory fields specified in Section 4.3. A retrieved object including an expiration time that is already past or missing a mandatory element MUST be ignored.

In order to avoid synchronized queries toward the server hosting the PvD Additional Information when an object expires, object updates are delayed by a randomized backoff time.

- o When a host performs a JSON object update after it detected a change in the PvD Option Sequence Number, it MUST add a delay before sending the query. The target time for the delay is calculated as a random time between zero and $2^{**}(10 + \text{Delay})$ milliseconds, where 'Delay' corresponds to the 4-bit unsigned integer in the last received PvD Option.
- o When a host last retrieved a JSON object at time A that includes a expiry time B using the "expires" key, and the host is configured to keep the PvD information up to date, it MUST add some randomness into its calculation of the time to fetch the update. The target time for fetching the updated object is calculated as a uniformly random time in the interval $[(B-A)/2, B]$.

In the example Figure 2, the delay field value is 1, this means that the host calculates its delay by choosing a uniformly random time between 0 and $2^{**}(10 + 1)$ milliseconds, i.e., between 0 and 2048 milliseconds.

Since the 'Delay' value is directly within the PvD Option rather than the object itself, an operator may perform a push-based update by incrementing the Sequence Number value while changing the Delay value depending on the criticality of the update and its PvD Additional Information servers capacity.

In addition to adding a random delay when fetching Additional Information, hosts MUST enforce a minimum time between requesting Additional Information for a given PvD on the same network. This minimum time is RECOMMENDED to be 10 seconds, in order to avoid hosts causing a denial-of-service on the PvD server. Hosts also MUST limit the number of requests that are made to different PvD Additional Information servers on the same network within a short period of time. A RECOMMENDED value is to issue no more than five PvD Additional Information requests in total on a given network within 10 seconds. For more discussion, see Section 6.

The PvD Additional Information object includes a set of IPv6 prefixes (under the key "prefixes") which MUST be checked against all the Prefix Information Options advertised in the RA. If any of the prefixes included in any associated PIO is not covered by at least one of the listed prefixes, the associated PvD information MUST be considered to be a misconfiguration, and MUST NOT be used by the host. See Section 4.4 for more discussion on handling such misconfigurations.

If the request for PvD Additional Information fails due to a TLS certificate validation error, an HTTP error, or because the retrieved file does not contain valid PvD JSON, hosts MUST close any connection

used to fetch the PvD Additional Information, and MUST NOT request the information for that PvD ID again for the duration of the local network attachment. If a host detects 10 or more such failures to fetch PvD Additional Information, the local network is assumed to be misconfigured or under attack, and the host MUST NOT make any further requests for any PvD Additional Information, belonging to any PvD ID, for the duration of the local network attachment. For more discussion, see Section 6.

4.2. Operational Consideration to Providing the PvD Additional Information

Whenever the H-flag is set in the PvD Option, a valid PvD Additional Information object MUST be made available to all hosts receiving the RA by the network operator. In particular, when a captive portal is present, hosts MUST still be allowed to perform DNS, certificate validation, and HTTP over TLS operations related to the retrieval of the object, even before logging into the captive portal.

Routers SHOULD increment the PVD Option Sequence Number by one whenever a new PvD Additional Information object is available and should be retrieved by hosts. If the value exceeds what can be stored in the Sequence Number field, it MUST wrap back to zero.

The server providing the JSON files SHOULD also check whether the client address is contained by the prefixes listed in the additional information, and SHOULD return a 403 response code if there is no match.

4.3. PvD Additional Information Format

The PvD Additional Information is a JSON object.

The following table presents the mandatory keys which MUST be included in the object:

JSON key	Description	Type	Example
identifier	PvD ID FQDN	String	"pvd.example.com."
expires	Date after which this object is no longer valid	[RFC3339] Date	"2020-05-23T06:00:00Z"
prefixes	Array of IPv6 prefixes valid for this PvD	Array of strings	["2001:db8:1::/48", "2001:db8:4::/48"]

A retrieved object which does not include all three of these keys at the root of the JSON object MUST be ignored. All three keys need to be validated, otherwise the object MUST be ignored. The value stored for "identifier" MUST be matched against the PvD ID FQDN presented in the PvD RA option using the comparison mechanism described in Section 3.4. The value stored for "expires" MUST be a valid date in the future. If the PIO of the received RA is not covered by at least one of the "prefixes" key, the retrieved object SHOULD be ignored.

The following table presents some optional keys which MAY be included in the object.

JSON key	Description	Type	Example
dnsZones	DNS zones searchable and accessible	Array of strings	["example.com", "sub.example.com"]
noInternet	No Internet, set to "true" when the PvD is restricted.	Boolean	true

It is worth noting that the JSON format allows for extensions. Whenever an unknown key is encountered, it MUST be ignored along with its associated elements.

Private-use or experimental keys MAY be used in the JSON dictionary. In order to avoid such keys colliding with IANA registry keys, implementers or vendors defining private-use or experimental keys

MUST create sub-dictionaries. If a set of PvD Additional Information keys are defined by an organization that has a Formal URN Namespace [URN], the URN namespace SHOULD be used as the top-level JSON key for the sub-dictionary. For other private uses, the sub-dictionary key SHOULD follow the format of "vendor-*", where the "*" is replaced by the implementer's or vendor's identifier. For example, keys specific to the FooBar organization could use "vendor-foobar". If a host receives a sub-dictionary with an unknown key, the host MUST ignore the contents of the sub-dictionary.

4.3.1. Example

The following two examples show how the JSON keys defined in this document can be used:

```
{
  "identifier": "cafe.example.com.",
  "expires": "2020-05-23T06:00:00Z",
  "prefixes": ["2001:db8:1::/48", "2001:db8:4::/48"],
}

{
  "identifier": "company.foo.example.com.",
  "expires": "2020-05-23T06:00:00Z",
  "prefixes": ["2001:db8:1::/48", "2001:db8:4::/48"],
  "vendor-foo":
    {
      "private-key": "private-value",
    },
}
```

4.4. Detecting misconfiguration and misuse

Hosts MUST validate the TLS server certificate when retrieving PvD Additional Information, as detailed in Section 4.1.

Hosts MUST verify that all prefixes in all the RA PIOs are covered by a prefix from the PvD Additional Information. An adversarial router attempting to spoof the definition of an Explicit PvD, without the ability to modify the PvD Additional Information, would need to perform NAT66 in order to circumvent this check. Thus, this check cannot prevent all spoofing, but it can detect misconfiguration or mismatched routers that are not adding a NAT.

If NAT66 is being added in order to spoof PvD ownership, the HTTPS server for additional information can detect this misconfiguration. The HTTPS server SHOULD validate the source addresses of incoming connections (see Section 4.1). This check gives reasonable assurance

that neither NPTv6 [RFC6296] nor NAT66 were used and restricts the information to the valid network users. If the PvD does not provision IPv4 (it does not include the 'L' bit in the RA), the server cannot validate the source addresses of connections using IPv4. Thus, the PvD ID FQDN for such PvDs SHOULD NOT have a DNS A record.

5. Operational Considerations

This section describes some example use cases of PvDs. For the sake of simplicity, the RA messages will not be described in the usual ASCII art but rather in an indented list. Values in the PvD Option header that are not included in the example are assumed to be zero or false (such as the H-flag, Sequence Number, and Delay fields).

5.1. Exposing Extra RA Options to PvD-Aware Hosts

In this example, there is one RA message sent by the router. This message contains some options applicable to all hosts on the network, and also a PvD Option that also contains other options only visible to PvD-aware hosts.

- o RA Header: router lifetime = 6000
- o Prefix Information Option: length = 4, prefix = 2001:db8:cafe::/64
- o PvD Option header: length = 3 + 5 + 4, PvD ID FQDN = example.org., R-flag = 0 (actual length of the header with padding 24 bytes = 3 * 8 bytes)
 - * Recursive DNS Server: length = 5, addresses = [2001:db8:cafe::53, 2001:db8:f00d::53]
 - * Prefix Information Option: length = 4, prefix = 2001:db8:f00d::/64

Note that a PvD-aware host will receive two different prefixes, 2001:db8:cafe::/64 and 2001:db8:f00d::/64, both associated with the same PvD (identified by "example.org."). A non-PvD-aware host will only receive one prefix, 2001:db8:cafe::/64.

5.2. Different RAs for PvD-Aware and Non-PvD-Aware Hosts

It is expected that for some years, networks will have a mixed environment of PvD-aware hosts and non-PvD-aware hosts. If there is a need to give specific information to PvD-aware hosts only, then it is RECOMMENDED to send two RA messages, one for each class of hosts. This approach allows for two distinct sets of configuration

information to be sent in a way that will not disrupt non-PvD-aware hosts. It also lowers the risk that a single RA message will approach its MTU limit due to duplicated information.

If two RA messages are sent for this reason, they MUST be sent from two different link-local source addresses (Section 3.2). For example, here is the RA sent for non-PvD-aware hosts:

- o RA Header: router lifetime = 6000 (non-PvD-aware hosts will use this router as a default router)
- o Prefix Information Option: length = 4, prefix = 2001:db8:cafe::/64
- o Recursive DNS Server Option: length = 3, addresses = [2001:db8:cafe::53]
- o PvD Option header: length = 3 + 2, PvD ID FQDN = foo.example.org., R-flag = 1 (actual length of the header 24 bytes = 3 * 8 bytes)
- * RA Header: router lifetime = 0 (PvD-aware hosts will not use this router as a default router), implicit length = 2

And here is the RA sent for PvD-aware hosts:

- o RA Header: router lifetime = 0 (non-PvD-aware hosts will not use this router as a default router)
- o PvD Option header: length = 3 + 2 + 4 + 3, PvD ID FQDN = bar.example.org., R-flag = 1 (actual length of the header 24 bytes = 3 * 8 bytes)
- * RA Header: router lifetime = 1600 (PvD-aware hosts will use this router as a default router), implicit length = 2
- * Prefix Information Option: length = 4, prefix = 2001:db8:f00d::/64
- * Recursive DNS Server Option: length = 3, addresses = [2001:db8:f00d::53]

In the above example, non-PvD-aware hosts will only use the first listed RA sent by their default router and using the 2001:db8:cafe::/64 prefix. PvD-aware hosts will autonomously configure addresses from both PIOs, but will only use the source address in 2001:db8:f00d::/64 to communicate past the first hop router since only the router sending the second RA will be used as default router; similarly, they will use the DNS server 2001:db8:f00d::53 when communicating from this address.

5.3. Enabling Multi-homing for PvD-Aware Hosts

In this example, the goal is to have one prefix from one RA be usable by both non-PvD-aware and PvD-aware hosts; and to have another prefix usable only by PvD-aware hosts. This allows PvD-aware hosts to be able to effectively multi-home on the network.

The first RA is usable by all hosts. The only difference for PvD-aware hosts is that they can explicitly identify the PvD ID associated with the RA. PvD-aware hosts will also use this prefix to communicate with non-PvD-aware hosts on the same network.

- o RA Header: router lifetime = 6000 (non-PvD-aware hosts will use this router as a default router)
- o Prefix Information Option: length = 4, prefix = 2001:db8:cafe::/64
- o Recursive DNS Server Option: length = 3, addresses = [2001:db8:cafe::53]
- o PvD Option header: length = 3, PvD ID FQDN = foo.example.org., R-flag = 0 (actual length of the header 24 bytes = 3 * 8 bytes)

The second RA contains a prefix usable only by PvD-aware hosts. Non-PvD-aware hosts will ignore this RA; hence, the only PvD-aware hosts will be multi-homed.

- o RA Header: router lifetime = 0 (non-PvD-aware hosts will not use this router as a default router)
- o PvD Option header: length = 3 + 2 + 4 + 3, PvD ID FQDN = bar.example.org., R-flag = 1 (actual length of the header 24 bytes = 3 * 8 bytes)
 - * RA Header: router lifetime = 1600 (PvD-aware hosts will use this router as a default router), implicit length = 2
 - * Prefix Information Option: length = 4, prefix = 2001:db8:f00d::/64
 - * Recursive DNS Server Option: length = 3, addresses = [2001:db8:f00d::53]

Note: the above examples assume that the router has received its PvD IDs from upstream routers or via some other configuration mechanism. Another document could define ways for the router to generate its own PvD IDs to allow the above scenario in the absence of PvD ID provisioning.

5.4. Providing Additional Information to PvD-Aware Hosts

In this example, the router indicates that it provides additional information using the H-flag. The Sequence Number on the PvD Option is set to 7 in this example.

- o RA Header: router lifetime = 6000
- o Prefix Information Option: length = 4, prefix = 2001:db8:cafe::/64
- o Recursive DNS Server Option: length = 3, addresses=[2001:db8:cafe::53]
- o PvD Option header: length = 3, PvD ID FQDN = cafe.example.com., Sequence Number = 7, R-flag = 0, H-flag = 1 (actual length of the header with padding 24 bytes = 3 * 8 bytes)

A PvD-aware host will fetch `https://cafe.example.com/.well-known/pvd` to get the additional information. The following example shows a GET request that the host sends, in HTTP/2 syntax [RFC7540]:

```
:method = GET
:scheme = https
:authority = cafe.example.com
:path = /.well-known/pvd
accept = application/pvd+json
```

The HTTP server will respond with the JSON additional information:

```
:status = 200
content-type = application/pvd+json
content-length = 116

{
  "identifier": "cafe.example.com.",
  "expires": "2020-05-23T06:00:00Z",
  "prefixes": ["2001:db8:cafe::/48"],
}
```

At this point, the host has the additional information, and knows the expiry time. When either the expiry time passes, or a new Sequence Number is provided in an RA, the host will re-fetch the additional information.

For example, if the router sends a new RA with the Sequence Number set to 8, the host will re-fetch the additional information:

- o PvD Option header: length = 3 + 5 + 4 , PvD ID FQDN =
cafe.example.com., Sequence Number = 8, R-flag = 0, H-flag = 1
(actual length of the header with padding 24 bytes = 3 * 8 bytes)

However, if the router sends a new RA, but the Sequence Number has not changed, the host would not re-fetch the additional information (until and unless the expiry time of the additional information has passed).

6. Security Considerations

Since the PvD ID RA option can contain an RA header and other RA options, any security considerations that apply for specific RA options continue to apply when used within a PvD ID option.

Although some solutions such as IPsec or SeND [RFC3971] can be used in order to secure the IPv6 Neighbor Discovery Protocol, in practice actual deployments largely rely on link layer or physical layer security mechanisms (e.g., 802.1x [IEEE8021X]) in conjunction with RA Guard [RFC6105].

If multiple RAs are sent for a single PvD to avoid fragmentation, dropping packets can lead to processing only part of a PvD ID option, which could lead to hosts receiving only part of the contained options. As discussed in Section 3.2, routers MUST include the PvD ID option in all fragments generated.

This specification does not improve the Neighbor Discovery Protocol security model, but simply validates that the owner of the PvD FQDN authorizes its use with the prefix advertised by the router. In combination with implicit trust in the local router (if present), this gives the host some level of assurance that the PvD is authorized for use in this environment. However, when the local router cannot be trusted, no such guarantee is available.

It must be noted that Section 4.4 of this document only provides reasonable assurance against misconfiguration but does not prevent a hostile network access provider from advertising incorrect information that could lead applications or hosts to select a hostile PvD. However, a host that correctly implements the multiple PvD architecture ([RFC7556]) using the mechanism described in this document will be less susceptible to some attacks than a host that does not by being able to check for the various misconfigurations or inconsistencies described in this document.

Since expiration times provided in PvD Additional Information use absolute time, these values can be skewed for hosts without an

accurate time base, or due to clock skew. Such time values MUST NOT be used for security-sensitive functionality or decisions.

An attacker generating RAs on a local network can use the H-flag and the PvD ID to cause hosts on the network to make requests for PvD Additional Information from servers. This can become a denial-of-service attack, in which an attacker can amplify its attack by triggering TLS connections to arbitrary servers in response to sending UDP packets containing RA messages. To mitigate this attack, hosts MUST:

- o limit the rate at which they fetch a particular PvD's Additional Information;
- o limit the rate at which they fetch any PvD Additional Information on a given local network;
- o stop making requests for a PvD ID that does not respond with valid JSON;
- o stop making requests for all PvD IDs once a certain number of failures is reached on a particular network.

Details are provided in Section 4.1. This attack can be targeted at generic web servers, in which case the host behavior of stopping requesting for any server that doesn't behave like a PvD Additional Information server is critical. Limiting requests for a specific PvD ID might not be sufficient if the attacker changes the PvD ID values quickly, so hosts also need to stop requesting if they detect consistent failure when on a network that is under attack. For cases in which an attacker is pointing hosts at a valid PvD Additional Information server (but one that is not actually associated with the local network), the server SHOULD reject any requests that do not originate from the expected IPv6 prefix as described in Section 4.2.

7. Privacy Considerations

Retrieval of the PvD Additional Information over HTTPS requires early communications between the connecting host and a server which may be located further than the first hop router. Although this server is likely to be located within the same administrative domain as the default router, this property can't be ensured. To minimize the leakage of identity information while retrieving the PvD Additional Information, hosts SHOULD make use of an IPv6 temporary address and SHOULD NOT include any privacy-sensitive data, such as a User-Agent header field or an HTTP cookie.

Hosts might not always fetch PvD Additional Information, depending on whether or not they expect to use the information. However, if a host whitelisted only certain PvD IDs for which to fetch Additional Information, an attacker could send various PvD IDs in RAs to detect which PvD IDs are whitelisted by the client. To avoid this, hosts SHOULD either fetch Additional Information for all eligible PvD IDs on a given local network, or fetch the information for none of them.

From a user privacy perspective, retrieving the PvD Additional Information is not different from establishing a first connection to a remote server, or even performing a single DNS lookup. For example, most operating systems already perform early queries to static web sites, such as `http://captive.example.com/hotspot-detect.html`, in order to detect the presence of a captive portal.

The DNS queries associated with the PvD Additional Information MUST use the DNS servers indicated by the associated PvD, as described in Section 4.1. This ensures the name of the PvD Additional Information server is not unintentionally sent on another network, thus leaking identifying information about the networks with which the client is associated.

There may be some cases where hosts, for privacy reasons, should refrain from accessing servers that are located outside a certain network boundary. In practice, this could be implemented as a whitelist of 'trusted' FQDNs and/or IP prefixes that the host is allowed to communicate with. In such scenarios, the host SHOULD check that the provided PvD ID, as well as the IP address that it resolves into, are part of the allowed whitelist.

Network operators SHOULD restrict access to PvD Additional Information to only expose it to hosts that are connected to the local network, especially if the Additional Information would provide information about local network configuration to attackers. This can be implemented by whitelisting access from the addresses and prefixes that the router provides for the PvD, which will match the prefixes contained in the PvD Additional Information. This technique is described in Section 4.2.

8. IANA Considerations

Upon publication of this document, IANA is asked to remove the 'reclaimable' tag off the value 21 for the PvD Option (from the IPv6 Neighbor Discovery Option Formats registry).

8.1. New entry in the Well-Known URIs Registry

IANA is asked to add a new entry in the Well-Known URIs registry [RFC8615] with the following information:

URI suffix: 'pvd'

Change controller: IETF

Specification document: this document

Status: permanent

Related information: N/A

8.2. Additional Information PvD Keys Registry

IANA is asked to create and maintain a new registry called "Additional Information PvD Keys", which will reserve JSON keys for use in PvD additional information. The initial contents of this registry are given in Section 4.3, including both the table of mandatory keys and the table of optional keys.

The status of a key as mandatory or optional is intentionally not denoted in the table to allow for flexibility in future use cases. Any new assignments of keys will be considered as optional for the purpose of the mechanism described in this document.

New assignments for Additional Information PvD Keys Registry will be administered by IANA through Expert Review [RFC8126]. Experts are requested to ensure that defined keys do not overlap in names or semantics, and represent non-vendor-specific use cases. Vendor-specific keys SHOULD use sub-dictionaries, as described in Section 4.3.

IANA is asked to place this registry in a new page, entitled "Provisioning Domains (PvDs)".

8.3. PvD Option Flags Registry

IANA is also asked to create and maintain a new registry entitled "PvD Option Flags" reserving bit positions from 0 to 12 to be used in the PvD Option bitmask. Bit position 0, 1 and 2 are assigned by this document (as specified in Figure 1). Future assignments require Standards Action [RFC8126].

Since these flags apply to an IPv6 Router Advertisement Option, IANA is asked to place this registry under the existing "Internet Control

Message Protocol version 6 (ICMPv6) Parameters" page, as well as providing a link on the new "Provisioning Domains (PvDs)" page.

8.4. PvD JSON Media Type Registration

This document registers the media type for PvD JSON text, "application/pvd+json".

Type Name: application

Subtype Name: pvd+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 6.

Interoperability considerations: This document specifies the 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 networks advertising additional Provisioning Domain information, and clients looking up such information.

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: IETF

Change controller: IETF

9. Acknowledgments

Many thanks to M. Stenberg and S. Barth for their earlier work: [I-D.stenberg-mif-mpvd-dns], as well as to Basile Bruneau who was author of an early version of this document.

Thanks also to Marcus Keane, Mikael Abrahamsson, Ray Bellis, Zhen Cao, Tim Chown, Lorenzo Colitti, Michael Di Bartolomeo, Ian Farrer, Phillip Hallam-Baker, Bob Hinden, Tatuya Jinmei, Erik Kline, Ted Lemon, Paul Hoffman, Dave Thaler, Suresh Krishnan, Gorrry Fairhurst, Jen Lenkova, Veronika McKillop, Mark Townsley and James Woodyatt for useful and interesting discussions and reviews.

Finally, special thanks to Thierry Danis for his valuable inputs and implementation efforts, Tom Jones for his integration effort into the NEAT project and Rigil Salim for his implementation work.

10. References

10.1. Normative References

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [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>>.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<https://www.rfc-editor.org/info/rfc3339>>.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <<https://www.rfc-editor.org/info/rfc4191>>.
- [RFC4343] Eastlake 3rd, D., "Domain Name System (DNS) Case Insensitivity Clarification", RFC 4343, DOI 10.17487/RFC4343, January 2006, <<https://www.rfc-editor.org/info/rfc4343>>.

- [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>>.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<https://www.rfc-editor.org/info/rfc4941>>.
- [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <<https://www.rfc-editor.org/info/rfc6724>>.
- [RFC6980] Gont, F., "Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery", RFC 6980, DOI 10.17487/RFC6980, August 2013, <<https://www.rfc-editor.org/info/rfc6980>>.
- [RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <<https://www.rfc-editor.org/info/rfc7493>>.
- [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>>.
- [RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <<https://www.rfc-editor.org/info/rfc7556>>.
- [RFC8028] Baker, F. and B. Carpenter, "First-Hop Router Selection by Hosts in a Multi-Prefix Network", RFC 8028, DOI 10.17487/RFC8028, November 2016, <<https://www.rfc-editor.org/info/rfc8028>>.
- [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>>.
- [RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <<https://www.rfc-editor.org/info/rfc8615>>.

10.2. Informative References

- [I-D.kline-mif-mpvd-api-reqs]
Kline, E., "Multiple Provisioning Domains API Requirements", draft-kline-mif-mpvd-api-reqs-00 (work in progress), November 2015.
- [I-D.stenberg-mif-mpvd-dns]
Stenberg, M. and S. Barth, "Multiple Provisioning Domains using Domain Name System", draft-stenberg-mif-mpvd-dns-00 (work in progress), October 2015.
- [IEEE8021X]
IEEE, "IEEE Standards for Local and Metropolitan Area Networks, Port-based Network Access Control, IEEE Std".
- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <<https://www.rfc-editor.org/info/rfc2131>>.
- [RFC3646] Droms, R., Ed., "DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3646, DOI 10.17487/RFC3646, December 2003, <<https://www.rfc-editor.org/info/rfc3646>>.
- [RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "Secure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <<https://www.rfc-editor.org/info/rfc3971>>.
- [RFC4389] Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery Proxies (ND Proxy)", RFC 4389, DOI 10.17487/RFC4389, April 2006, <<https://www.rfc-editor.org/info/rfc4389>>.
- [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>>.

- [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>>.
- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <<https://www.rfc-editor.org/info/rfc6146>>.
- [RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, DOI 10.17487/RFC6147, April 2011, <<https://www.rfc-editor.org/info/rfc6147>>.
- [RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", RFC 6296, DOI 10.17487/RFC6296, June 2011, <<https://www.rfc-editor.org/info/rfc6296>>.
- [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>>.
- [RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <<https://www.rfc-editor.org/info/rfc7049>>.
- [RFC7278] Byrne, C., Drown, D., and A. Vizdal, "Extending an IPv6 /64 Prefix from a Third Generation Partnership Project (3GPP) Mobile Interface to a LAN Link", RFC 7278, DOI 10.17487/RFC7278, June 2014, <<https://www.rfc-editor.org/info/rfc7278>>.
- [RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <<https://www.rfc-editor.org/info/rfc7540>>.
- [RFC8106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 8106, DOI 10.17487/RFC8106, March 2017, <<https://www.rfc-editor.org/info/rfc8106>>.

- [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>>.
- [URN] IANA, "Uniform Resource Names (URN) Namespaces", <<https://www.iana.org/assignments/urn-namespaces/urn-namespaces.xhtml>>.

Authors' Addresses

Pierre Pfister
Cisco
11 Rue Camille Desmoulins
Issy-les-Moulineaux 92130
France

Email: ppfister@cisco.com

Eric Vyncke
Cisco
De Kleetlaan, 6
Diegem 1831
Belgium

Email: evyncke@cisco.com

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

Email: tpauly@apple.com

David Schinazi
Google LLC
1600 Amphitheatre Parkway
Mountain View, California 94043
United States of America

Email: dschinazi.ietf@gmail.com

Wenqin Shao
Cisco
11 Rue Camille Desmoulins
Issy-les-Moulineaux 92130
France

Email: wenshao@cisco.com

template
Internet-Draft
Intended status: Informational
Expires: October 4, 2017

D. Bird
W. Kumari
Google
April 2, 2017

Captive Portal ICMP Messages
draft-wkumari-capport-icmp-unreach-02

Abstract

This document defines a new ICMP Type for Captive Portal Messages. The ICMP Type will only be known to clients supporting this specification and provides both generic and flow 5-tuple specific notifications from the Captive Portal NAS.

Further, This document defines a multi-part ICMP extension to ICMP Destination Unreachable messages to signal, not only that the packet was dropped, but that it was dropped due to an Access Policy requiring Captive Portal interaction. Legacy clients will only be processing the ICMP Destination Unreachable.

[Editor note: The IETF is currently discussing improvements in captive portal interactions and user experience improvements. See: <https://www.ietf.org/mailman/listinfo/captive-portals>]

[RFC Editor: Please remove this before publication. This document is being stored in github at <https://github.com/wlanmac/draft-wkumari-capport-icmp-unreach> . Authors gratefully accept pull requests, and keep the latest (edit buffer) versions there, so commenters can follow along at home.]

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 <http://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 October 4, 2017.

Copyright Notice

Copyright (c) 2017 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 (<http://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
1.2. Terminology	3
2. Captive Portal ICMP	4
2.1. Session-ID	4
2.2. Flags	4
2.3. Validity	5
2.4. Delay	5
2.5. Policy Class	5
2.6. Message Code/C-Type	6
2.7. Message Type	6
2.8. Extension Object	7
3. Captive Portal URL Formatting	8
4. IANA Considerations	9
5. Security Considerations	9
6. Acknowledgements	9
7. References	10
7.1. Normative References	10
7.2. Informative References	10
Appendix A. Changes / Author Notes.	10
Authors' Addresses	11

1. Introduction

Captive Portals work by blocking (or redirecting) communications outside of a "walled garden" until the user has either authenticated, acknowledged an Acceptable Use Policy (AUP), or otherwise satisfied the requirements of the Captive Portal. Depending on the captive portal implementation, connections other than HTTP will either timeout (silently packets dropped) or meet with a different,

inaccurate, error condition (like a TCP reset, for TCP connections, or ICMP Destination Unreachable with existing codes).

A current option for captive portal networks is to reject traffic not in the walled garden by returning the Destination Unreachable either Host or Network Administratively Prohibited. However, these codes are typically permanent policies and do not specifically indicate a captive portal is in use.

This document defines a new ICMP Type for Captive Portal. The Captive Portal ICMP Type can be used to send flow 5-tuple specific or general notifications to user devices. As a new ICMP type, it is expected to be ignored by legacy devices.

This document also defines an Extension Object that can be appended to selected multi-part ICMP messages to inform the user device that they are behind a captive portal, in addition to the underlying ICMP information. Devices able to understand the extension get extra information about the captive portal access policy, whereas legacy devices just understand the underlying ICMP message.

The Captive Portal and Destination Unreachable types provide the Captive Portal NAS options in terms of what notifications legacy devices can and should understand.

The Captive Portal ICMP Messages only provide notification. They do not provide any configuration. For that, we use [RFC7710] and the Captive Portal URI it provides.

1.1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.2. Terminology

Capport ICMP device Device or operating system compliant to this specification.

CP-NAS Network Access Server implementing Captive Portal enforcement.

Legacy device Device or operating system not compliant to this specification.

2. Captive Portal ICMP

Captive Portal ICMP messages come in two flavors. Messages can be sent using the Captive Portal ICMP Type or they can be sent as an ICMP Extension to an existing ICMP Type, such as Destination Unreachable. Data is encoded into the packet slightly differently in each case, however, the field formats remain consistent. All fields are in network byte order.

Capport ICMP devices MUST support [RFC7710].

2.1. Session-ID

An unsigned short session identifier that groups ICMP messages. ICMP messages containing the same value MUST be assumed to be part of the same access policy. Any change in this value between ICMP messages from the same source IP address MUST be considered by the client to mean a change in access policy has occurred and previous notifications are no longer valid.

```

      0                               1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-----+-----+-----+-----+
|                               |
|           Session-ID        |
|                               |
+-----+-----+-----+-----+
```

2.2. Flags

In Captive Portal ICMP Messages, a flags field contains bit flags for optional payload data fields. All data fields are unsigned 32bit integers.

Bit flags and their (optional) respective data fields:

```

      0 1 2 3 4 5 6 7
+-----+-----+-----+-----+
|V|D|P|   zero   |
+-----+-----+-----+-----+
```

V - 1 bit Validity

D - 1 bit Delay

P - 1 bit Policy Class

Optional fields included in flags appear in the ICMP payload in the same order as the respective bits.

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Validity (optional)                 |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Delay (optional)                    |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Policy-Class (optional)             |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

2.3. Validity

The Validity time, in seconds, that this result should be considered valid and the OS SHOULD not attempt to access the same resource in the meantime. During the Validity time, the NAS MAY chose to silently drop the packets of the same flow 5-tuple to selectively cause legacy clients to time-out connections.

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Validity (seconds as uint32)         |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

2.4. Delay

The Delay time, in seconds, is the time in future when this result should be considered valid. This is used to give advanced notice that a change in access policy is about to happen.

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                                     Delay (seconds as uint32)           |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

2.5. Policy Class

The Policy Class is an unsigned integer that provides a "hint" to the captive portal. When a client is specifically responding to a Captive Portal ICMP message and is launching a browser, the Policy Class is given to the portal as a reason for the visitor to visit the portal.

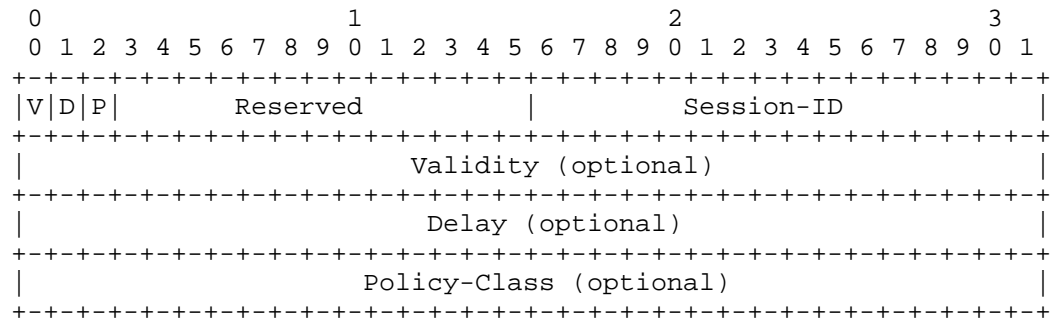
Length Number of 4 byte words of original datagram.

2.8. Extension Object

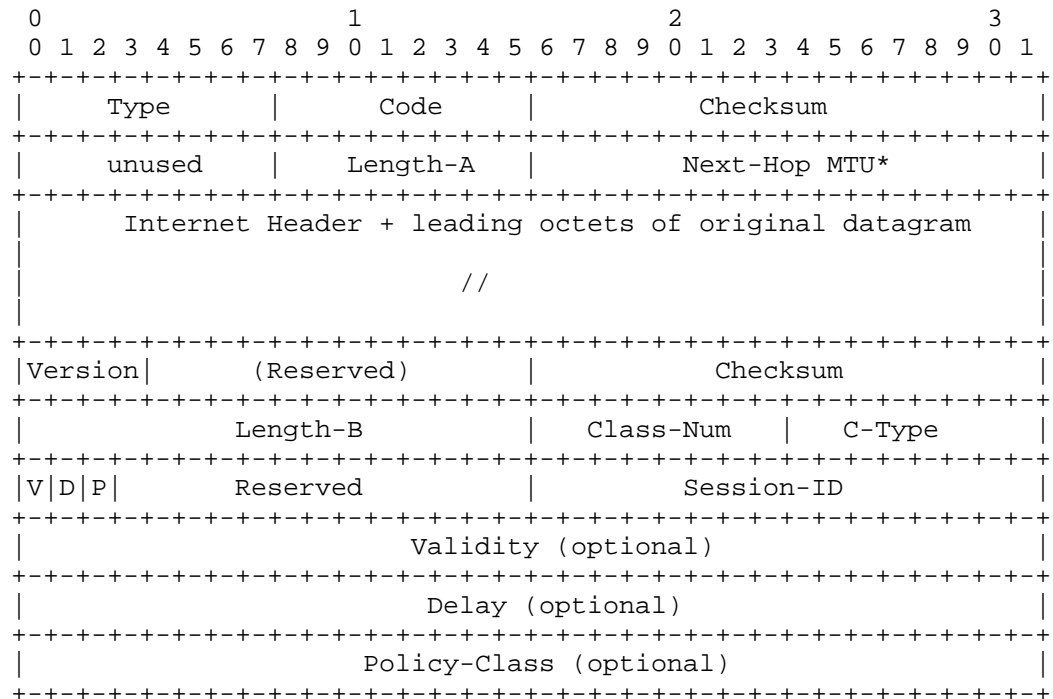
This document defines an extension object that can be appended to selected multi-part ICMP messages ([RFC4884]). This extension permits the CP-NAS to inform Capport ICMP Compliant devices that their connection has been blocked due to an Access Policy requiring interaction with the Captive Portal.

The Captive Portal Extension Object can be appended to the ICMP Destination Unreachable messages. When Legacy devices receive such messages, they will only understand the Destination Unreachable, ignoring the extensions.

When used in an Extension Object, the Captive Portal ICMP data fields are packed into an extension structure as shown below.



The following figure depicts the Destination Unreachable message with Captive Portal Extension. It must be preceded by an ICMP Extension Structure Header and an ICMP Object Header. Both are defined in [RFC4884].



Type Set to 3 for Destination Unreachable.

Code Can be any value Code value for Type.

Length-A Length, in 4 byte words, of original datagram.

Version Set to version 2, per RFC 4884.

Length-B Length of extension.

Class-Num Set to Captive Portal Class-Num.

C-Type See section 2.6.

3. Captive Portal URL Formatting

The Session-ID and Policy Class is used along with the RFC 7710 URI received from DHCP or IPv6 RA to send the user to the captive portal.

```
RFC_7710_URI . SEP .
'icmp_session=' . SESSION_ID . '&' .
'policy_class=' . [POLICY_CLASS[,POLICY_CLASS]]
```

RFC_7710_URI The URI received from DHCP or IPv6 RA per RFC 7710.

SEP If the RFC_7710_URI contains a '?', then SEP equals ampersand, otherwise a question mark.

SESSION_ID The Session-ID value in integer format.

POLICY_CLASS Zero or more Policy Class values gathers for the same Session-ID leading to the user notification..

Examples:

`https://wifi.domain.com/portal?icmp_session=10&policy_class=100`

`https://my.domain.com/?do=login&icmp_session=10&policy_class=100,20`

4. IANA Considerations

The IANA is requested to assign a Captive Portal ICMP Message Type, as well as Code values defined in section 2.6..

The IANA is also requested to assign a Class-Num identifier for the Captive Portal Extension Object from the ICMP Extension Object Classes and Class Sub-types registry.

The IANA is also requested to form and administer the corresponding class sub-type (C-Type) space per section 2.6.

5. Security Considerations

This method simply annotates existing ICMP Destination Unreachable messages to inform users why their connection was blocked. This technique can be used to inform captive portal detection probe software that there is a captive portal present (and potentially to connect to the URL handed out using draft-wkumari-dhc-capport. We anticipate that there will be a new solution devised (such as a well known URL / URI on captive portals) to allow the user / captive portal probe to do something more useful with this information.

6. Acknowledgements

The authors wish to thank the authors of RFC4950 (especially Ron Bonica) - I stole much of his text when writing the extension definition.

7. References

7.1. Normative References

- [RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <<http://www.rfc-editor.org/info/rfc792>>.
- [RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <<http://www.rfc-editor.org/info/rfc1122>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [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, <<http://www.rfc-editor.org/info/rfc3986>>.
- [RFC4884] Bonica, R., Gan, D., Tappan, D., and C. Pignataro, "Extended ICMP to Support Multi-Part Messages", RFC 4884, DOI 10.17487/RFC4884, April 2007, <<http://www.rfc-editor.org/info/rfc4884>>.
- [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, <<http://www.rfc-editor.org/info/rfc7710>>.

7.2. Informative References

- [I-D.ietf-sidr-iana-objects]
Manderson, T., Vegoda, L., and S. Kent, "RPKI Objects issued by IANA", draft-ietf-sidr-iana-objects-03 (work in progress), May 2011.

Appendix A. Changes / Author Notes.

[RFC Editor: Please remove this section before publication]

From -01 to 02.

- o Added a new ICMP Type, redefined message payload and flags, and introduces Codes/C-Types.

From -00 to 01.

- o Changed the Captive Portal URL to a URI, and specified that this can ONLY contain a path element, which is appended to `http://<gateway_ip>`. This is to prevent hijacking connections to other addresses.
- o Then removed the entire URL / URI scheme entirely.

From -genesis to -00.

- o Initial text.

Authors' Addresses

David Bird
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Email: dbird@google.com

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

Email: warren@kumari.net