Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: June 30, 2019

K. Larose
Agilicus
D. Dolson

December 27, 2018

CAPPORT Architecture
draft-ietf-capport-architecture-03

Abstract

This document aims to document consensus on the CAPPORT architecture. DHCP or Router Advertisements, an optional signaling protocol, and an HTTP API are used to provide the solution. The role of Provisioning Domains (PvDs) is described.

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 June 30, 2019.

Table of Contents

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 generally require a web server, some method to allow/block traffic, and some method to alert the user. Common methods of alerting the user involve modifying HTTP or DNS traffic.

Problems with captive portal implementations have been described in [I-D.nottingham-capport-problem]. [If that document cannot be published, consider putting its best parts into an appendix of this document.]

This document standardizes an architecture for implementing captive portals that provides tools for addressing most of those problems. We are guided by these principles:

- Solutions SHOULD NOT require the forging of responses from DNS or HTTP servers, or any other protocol. In particular, solutions SHOULD NOT require man-in-the-middle proxy of TLS traffic.

- 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 3GPP.

- Solutions MAY allow a device to be alerted that it is in a captive network when attempting to use any application on the network.

- Solutions SHOULD allow a device to learn that it is in a captive network before any application attempts to use the network.

- The state of captivity SHOULD be explicitly available to devices (in contrast to modification of DNS or HTTP, which is only indirectly machine-detectable by the client--by comparing responses to well-known queries with expected responses).

- The architecture MUST provide a path of incremental migration, acknowledging a huge variety of portals and end-user device implementations and software versions.

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 yet describe a mechanism for such devices to escape captivity.

The architecture uses the following mechanisms:

- Network provisioning protocols provide end-user devices with a 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 [RFC7710]. Other protocols (such as RADIUS), Provisioning Domains [I-D.ietf-intarea-provisioning-domains], or static configuration may also be used. A device MAY query this API at any time to determine whether the network is holding the device in a captive state.

- o End-user devices can be notified of captivity with Captive Portal Signals in response to traffic. This notification should work with any Internet protocol, not just clear-text HTTP. This notification does not carry the portal URI; rather it provides a notification to the User Equipment that it is in a captive state. This document will specify requirements for a signaling protocol which could generate Captive Portal Signals.

- o Receipt of a Captive Portal Signal informs an end-user device that it could be captive. In response, the device MAY query the provisioned API to obtain information about the network state. The device MAY take immediate action to satisfy the portal (according to its configuration/policy).

The architecture attempts to provide privacy, 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", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.2. Terminology

Captive Network: 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: The network equipment which enforces the traffic restriction.

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 network.

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

Captive Portal Signal: A notification from the network used to inform 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.

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 Captive Portal Enforcement, described in Section 2.4, until site-specific requirements have been met.

At this time we consider only devices with web browsers, with web applications being the means of satisfying Captive Portal Conditions.

- An example interactive User Equipment is a smart phone.
- 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 mechanism for notifying the user of the Captive Portal
- SHOULD have a web browser so that the user may navigate the Captive Portal user interface.
- MAY restrict application access to networks not granting full network access. E.g., a device connected to a mobile network may be connecting to a WiFi network; the operating system MAY avoid updating the default route until network access restrictions have been lifted (excepting access to the Captive Portal server). 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 access beyond the captive network, (b) the requirements may be fulfilled by manually visiting

the captive portal web application, and (c) legacy devices must continue to be supported.

2.2. Provisioning Service

Here we discuss candidate mechanisms for provisioning the User Equipment with the URI of the API to query captive portal state and navigate the portal.

2.2.1. DHCP or Router Advertisements

A standard for providing a portal URI using DHCP or Router Advertisements is described in [RFC7710]. The CAPPORT architecture expects this URI to indicate the API described in Section 2.3.

Although it is not clear from RFC7710 what protocol should be executed at the specified URI, some readers might have assumed it to be an HTML page, and hence there might be User Equipment assuming a browser should open this URI. For backwards compatibility, it is RECOMMENDED that the server check the "Accept" field when serving the URI, and serve HTML pages for "text/html" and serve the API for "application/json". [REVISIT: are these details appropriate?]

2.2.2. Provisioning Domains

Although still a work in progress, [I-D.ietf-intarea-provisioning-domains] proposes a mechanism for User Equipment to be provided with PvD Bootstrap Information containing the URI for a JSON file containing key-value pairs to be downloaded over HTTPS. This JSON file would fill the role of the Captive Portal API described in Section 2.3.

The PvD security model provides secure binding between the information provided by the trusted Router Advertisement and the HTTPS server.

One key-value pair can be used to indicate the network has restricted access, requiring captive portal navigation by a user. E.g., key="captivePortal" and value=<URI of portal>. The key-value pair should provide a different result when access is not restricted. E.g., key="captivePortal" and value="".

This JSON file is extensible, allowing new key-value pairs to indicate such things as network access expiry time, URI for API access by IOT devices, etc.

The PvD server MUST support multiple (repeated) queries from each User Equipment, always returning the current captive portal

information. The User Equipment is expected to make this query upon receiving (and validating) a Captive Portal Signal (see Section 2.5).

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 a device to perform clear-text "canary" HTTP 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, returning the current state of captivity for the equipment.

At minimum the API MUST provide: (1) the state of captivity and (2) a URI for a browser to present the portal application to the user. The API SHOULD provide evidence to the caller that it supports the present architecture.

When user equipment receives Captive Portal Signals, the user equipment MAY query the API to check the state. 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 for privacy and server authentication. The implementation of the API MUST ensure both privacy 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

The Captive Portal Enforcement component restricts network access to User Equipment according to site-specific policy. Typically User Equipment is permitted access to a small number of services and is denied general network access until it has performed some action.

The Captive Portal Enforcement component:

- Allows traffic through for allowed User Equipment.

- Blocks (discards) traffic for disallowed User Equipment.
- May signal User Equipment using the Captive Portal Signaling protocol if traffic is blocked.
- Permits disallowed User Equipment to access necessary APIs and web pages to fulfill requirements of exiting captivity.
- Updates allow/block rules per User Equipment in response to operations from the Captive Portal back-end.

2.5. Captive Portal Signal

User Equipment may send traffic outside the captive network prior to the Enforcement device granting it access. The Enforcement Device rightly blocks or resets these requests. However, lacking a signal from the Enforcement Device or interaction with the API server, the User Equipment can only guess at whether it is captive. Consequently, allowing the Enforcement Device to signal to the User Equipment that there is a problem with its connectivity may improve the user's experience.

An Enforcement Device may also want to inform the User Equipment of a pending expiry of its access to the external network, so providing the Enforcement Device the ability to preemptively signal may be desirable.

A specific Captive Portal Signaling Protocol is out of scope for this document. However, in order to ensure that future protocols fit into the architecture, requirements for a Captive Portal Signaling Protocol follow:

1. The notification SHOULD NOT be easy to spoof. If an attacker can send spoofed notifications to the User Equipment, they can cause the User Equipment to unnecessarily access the API. Rather than relying solely on rate limits to prevent problems, a good protocol will strive to limit the feasibility of such attacks.
2. It SHOULD be possible to send the notification before the captive portal closes. This will help ensure seamless connectivity for the user, as the User Equipment will not need to wait for a network failure to refresh its login. On receipt of preemptive notification, the User Equipment can prompt the user to refresh.
3. The signal SHOULD NOT include any information other than an indication that traffic is restricted, which can be used as a prompt to contact the API.

The Captive Portal Signaling Protocol does not provide any means of indicating that the network prevents access to some destinations. The intent is to rely on the Captive Portal API and the web portal to which it points to communicate local network policies.

The Captive Portal Enforcement function MAY send Captive Portal Signals when disallowed User Equipment attempts to send to the network. These signals MUST be rate-limited to a configurable rate.

The signals MUST NOT be sent to the Internet devices. The indications are only sent to the User Equipment.

2.6. Component Diagram

The following diagram shows the communication between each component.

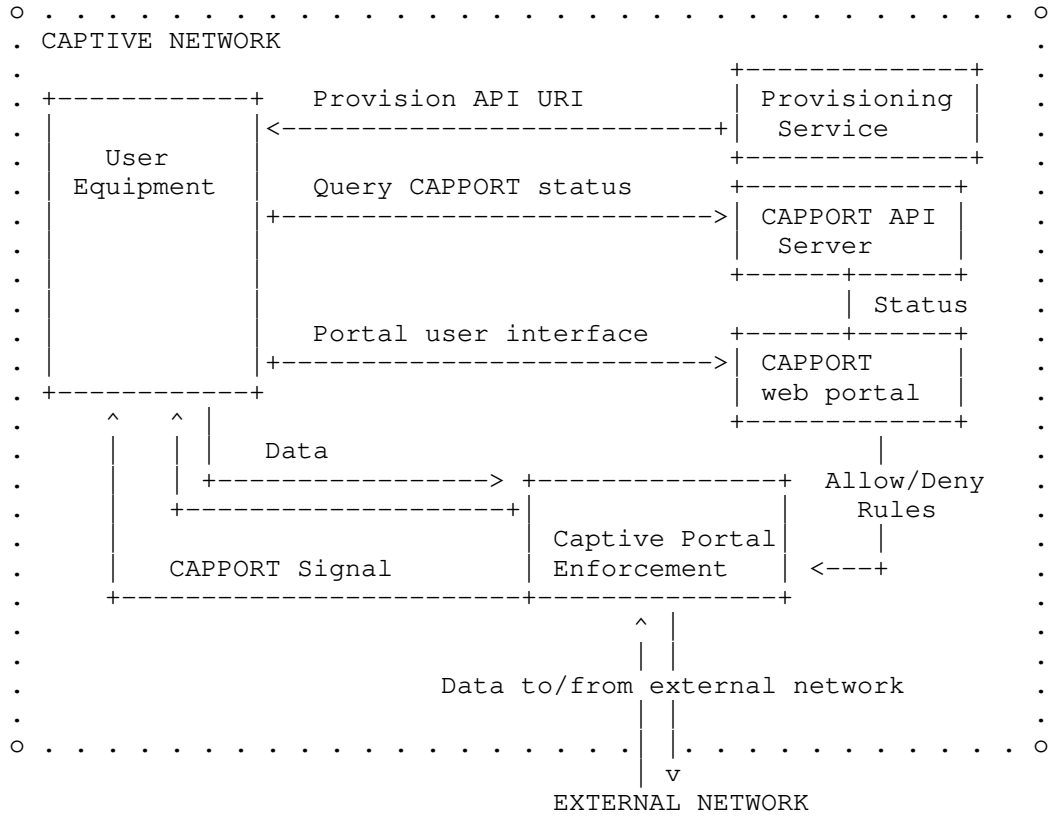


Figure 1: Captive Portal Architecture Component Diagram

In the diagram:

- During provisioning (e.g., DHCP), the User Equipment acquires the URI for the CAPPORT API.

- The User Equipment queries the API to learn of its state of captivity. If captive, the User Equipment presents the portal user interface to the user.

- The User Equipment attempts to communicate to the external network through the Captive Portal enforcement device.

- The Captive Portal Enforcement device either allows the User Equipment's packets to the external network, or if a signal has been implemented, responds with a Captive Portal Signal.

- The CAPPORT web portal server directs the Captive Portal Enforcement device to either allow or deny external network access for the User Equipment.

Although the provisioning, API, and web portal functions are shown as discrete blocks, they could of course be combined into a single element.

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 be able 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 MAY be a field contained in packets sent by the User Equipment to the External Network. Or, an Identifier MAY 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
- 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

In order to uniquely identify the User Equipment, at most one user equipment interacting with the other components of the Captive Portal MUST have a given value of the identifier.

Over time, the user equipment identified by the 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 is extended externally to devices such as billing systems, or where the identity of the User Equipment could imply liability.

3.2.3. Visible to the API

Since the API will need to perform operations which rely on the identity of the user equipment, such as query whether it is captive, the API needs to be able to relate requests 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 it should be permitted to communicate with 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 an Identifier

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

3.4. Examples of an Identifier

This section provides some examples of identifiers, along with some evaluation of whether they are good identifiers. The list of identifiers is not exhaustive. Other identifiers may be used. An important point to note is that whether the identifiers are good 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 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 property 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 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, 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.

4. Solution Workflow

This section aims to improve understanding by describing a possible workflow of solutions adhering to the architecture.

4.1. Initial Connection

This section describes a possible work-flow when User Equipment initially joins a Captive Network.

1. The User Equipment joins the Captive Network 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., [RFC7710]).

3. The User Equipment accesses the CAPPORT API to receive parameters of the Captive Network, including web-portal URI. (This step replaces the clear-text query to a canary URL.)

4. If necessary, the User navigates the web portal to gain access to the external network.

5. The Captive Portal API server indicates to the Captive Portal Enforcement device that the User Equipment is allowed to access the external network.

6. The User Equipment attempts a connection outside the captive network

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 work-flow when access is about to expire.

1. Precondition: the API server 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's UI indicates that the length of time left for its access has fallen below a threshold

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 web-portal URI again

6. The User extends their access through the web-portal

7. The User Equipment's access to the outside network continues uninterrupted

5. Acknowledgments

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

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 a network is operated by who claims to be operating it. The Provisioning Domain Architecture [RFC7556] provides some discussion on authenticating an operator.

Given that a user chooses to visit a Captive Portal URI, the URI location SHOULD be securely provided to the user's device. 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.

7.2. Authenticated APIs

The solution described here assumes that when the User Equipment needs to trust the API server, server authentication will be performed using TLS mechanisms.

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.

7.4. Risk of Nuisance Captive Portal

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 informs 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 API server are rate limited, limiting the impact of a repeated attack.

7.5. User Options

The Signal could inform 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 indications for privacy reasons. However, there is 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.

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>.

[RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <https://www.rfc-editor.org/info/rfc7556>.

[RFC7710] Kumari, W., Gudmundsson, O., Ebersman, P., and S. Sheng, "Captive-Portal Identification Using DHCP or Router Advertisements (RAs)", RFC 7710, DOI 10.17487/RFC7710, December 2015, <https://www.rfc-editor.org/info/rfc7710>.

8.2. Informative References

[I-D.ietf-intarea-provisioning-domains] Pfister, P., Vyncke, E., Pauly, T., Schinazi, D., and W. Shao, "Discovering Provisioning Domain Names and Data", draft-ietf-intarea-provisioning-domains-03 (work in progress), October 2018.

[I-D.nottingham-capport-problem] Nottingham, M., "Captive Portals Problem Statement", draft-nottingham-capport-problem-01 (work in progress), April 2016.

Authors' Addresses

Kyle Larose
Agilicus

Email: kyle@agilicus.com

David Dolson

Email: ddolson@acm.org

intarea
Internet-Draft
Intended status: Standards Track
Expires: April 22, 2019

P. Pfister
E. Vyncke, Ed.
Cisco
T. Pauly
D. Schinazi
Apple
W. Shao
Telecom-ParisTech
October 19, 2018

Discovering Provisioning Domain Names and Data
draft-ietf-intarea-provisioning-domains-03

Abstract

An increasing number of hosts access the Internet via multiple interfaces or, in IPv6 multi-homed networks, via multiple IPv6 prefix configurations context.

This document describes a way for hosts to identify such contexts, called Provisioning Domains (PvDs), where Fully Qualified Domain Names (FQDNs) act as PvD identifiers. Those identifiers are advertised in a new Router Advertisement (RA) option and, when present, are associated with the set of information included within the RA.

Based on this FQDN, hosts can retrieve additional information about their network access characteristics via an HTTP over TLS query. This allows applications to select which Provisioning Domains to use as well as to provide configuration parameters to the transport layer and above.

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 April 22, 2019.

Table of Contents

1. Introduction

It has become very common in modern networks for hosts to access the internet through different network interfaces, tunnels, or next-hop routers. To describe the set of network configurations associated with each access method, the concept of Provisioning Domain (PvD) was defined in [RFC7556].

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, deriving from it. The PVD ID Router Advertisement option may also contain a set of other RA options. Since such options are only considered by hosts implementing this specification, network operators may configure hosts that are 'PvD-aware' with PvDs that are ignored by other hosts.

Since PvD IDs are used to identify different ways to access the internet, multiple PvDs (with different PvD IDs) could 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 identical services.

This document also introduces a way for hosts to retrieve additional information related to a specific PvD by means of an HTTP over TLS query using an URI derived from the PvD ID. The retrieved JSON object contains additional information that would typically be considered unfit, or 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.

2. Terminology

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 [RFC2119].

In addition, 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. 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) which MUST belong to the network operator in order to avoid naming collisions. 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 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.

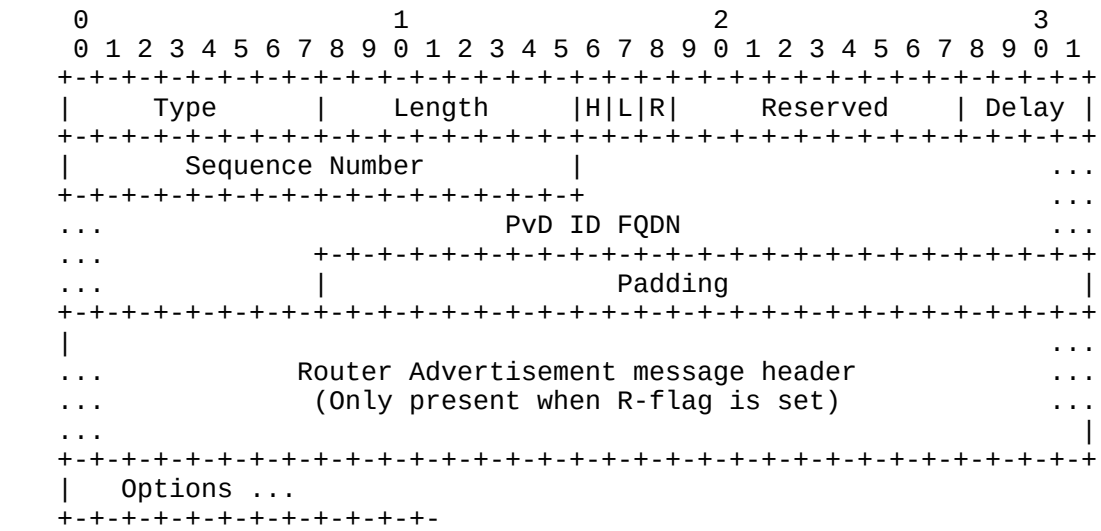


Figure 1: PvD ID Router Advertisements Option format

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 router is also providing IPv4 information using DHCPv4 (see Section 3.4.2).

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

Delay : (4 bits) Unsigned integer used to delay HTTP GET queries from hosts by a randomized backoff (see Section 4.1).

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

Sequence Number: (16 bits) Sequence number for the PvD Additional Information, as described in Section 4.

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

Padding : Zero or more padding octets to the next 8 octets boundary. 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 'Type', 'Code' and 'Checksum' fields (i.e. the first 32 bits), MUST be set to zero by the sender and ignored by the receiver. The 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 such as to be ignored by hosts that are not 'PvD-aware'.

Here is an example of a PvD option with example.org as the PvD ID FQDN and including a RDNSS and prefix information options (it also have the sequence number 123, presence of additional information to be fetched with a delay indicated as 5):

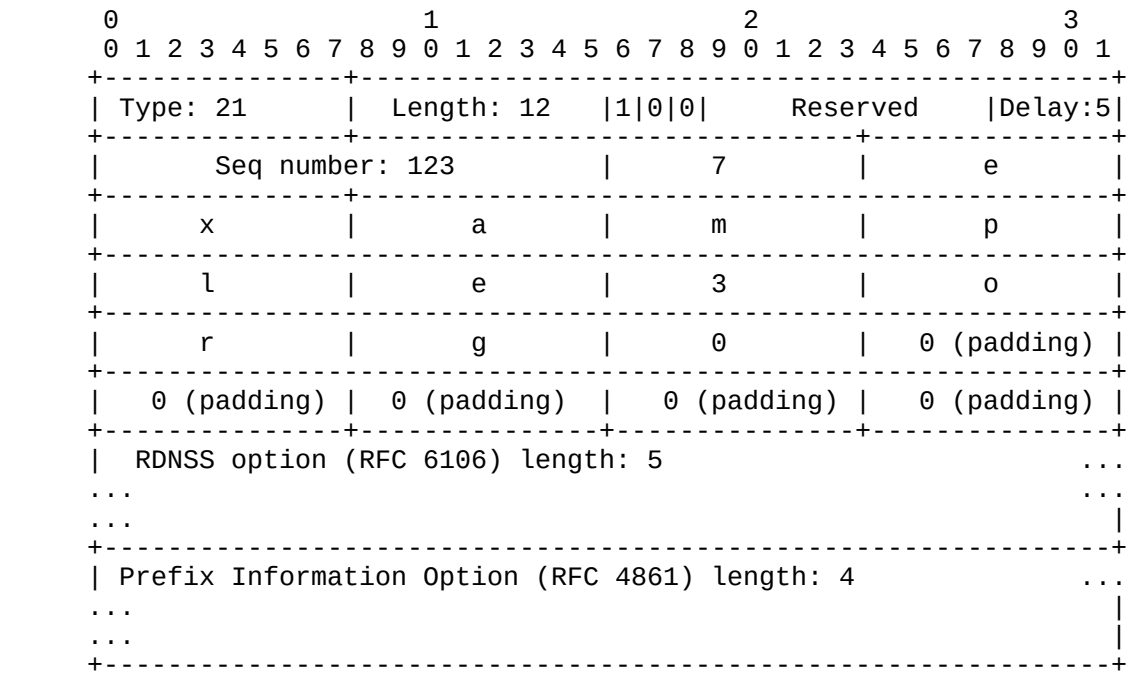


Figure 2

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. In particular, 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 others.

In order to provide multiple different PvDs, a router MUST send multiple RAs. Different explicit PvDs MAY be advertised with RAs using the same IPv6 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.

As specified in [RFC4861], when the set of options causes the size of an advertisement to exceed the link MTU, multiple router advertisements can be sent, 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. But 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. This ensure the backward compatibility required in section 3.3 of [RFC7556]. This behavior allows for a mixed-mode network with 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.

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

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 which last updated 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 (i.e., A=a), assuming ASCII with zero parity while non-alphabetic codes must match exactly (see also Section 3.1 of [RFC1035]). For example, "pvd.example.com." or "PvD.Example.coM." would refer to the same PvD.

While resolving names, executing the default address selection algorithm [RFC6724] or executing the default router selection algorithm when forwarding packets ([RFC2461], [RFC4191] and [RFC8028]), hosts MAY consider only the configuration associated with an arbitrary set 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 configuration elements using DHCPv6 (e.g., addresses or DNS recursive resolvers), they MUST be associated with the explicit or implicit PvD of the RA received on the same interface, sent from the same LLA, and with the O-flag or M-flag set [RFC4861]. If no such PvD is found, or whenever multiple different PvDs are found, the host behavior is unspecified.

This process requires hosts to keep track of received RAs, associated PvD IDs, and routers LLA; it also assumes that the router either acts as a DHCPv6 server or relay and uses the same LLA for DHCPv6 and RA traffic (which may not be the case when the router uses VRRP to send its RA).

3.4.2. DHCPv4 configuration association

When a host retrieves configuration elements from DHCPv4, they MUST be associated with the explicit PvD received on the same interface, whose PVD Options L-flag is set and, in the case of a non point-to-point link, using the same datalink address. If no such PvD is found, or whenever multiple different PvDs are found, the configuration elements coming from DHCPv4 MUST be associated with the implicit PvD identified by the interface on which the DHCPv4 transaction happened. The case of multiple explicit PvD for an IPv4 interface is undefined.

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. WiFi) is known as 'tethering'. Techniques such as ND-proxy [RFC4389], 64share [RFC7278] or prefix delegation (e.g. using DHCPv6-PD [RFC3633]) 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:

The same PVD-ID FQDN.

The same H-bit, Delay and Sequence Number values.

The L bit set whenever the host is sharing IPv4 connectivity received from the same upstream interface.

The bits from the Reserved field set to 0.

The values of the R-bit, 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.

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 [RFC7159].

The purpose of this additional set of information is to securely provide additional information to applications about the connectivity that is provided using a given interface and source address pair. 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.

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 [RFC5785]. Inversely, hosts MUST NOT do so whenever the H-flag is not set.

Note that the DNS name resolution of the PvD ID, the PKI checks as well as the actual query 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. If the host has

a temporary address per [RFC4941] in this PvD, then hosts SHOULD use a temporary address to fetch the PvD Additional Information and SHOULD 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 abandon 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 host MAY get a file containing a single JSON object. When a JSON object could not be retrieved, an error message SHOULD be logged and/or displayed in a rate-limited fashion.

After retrieval of the PvD Additional Information, hosts MUST keep track of the Sequence Number value received in subsequent RAs including the same PvD ID. In case the new value is greater than the value that was observed when the PvD Additional Information object was retrieved (using serial number arithmetic comparisons [RFC1982]), or whenever the validity time included in the PVD Additional Information JSON object is expired, hosts MUST either perform a new query and retrieve a new version of the object, or, failing that, deprecate the object and stop using the additional information provided in the JSON object.

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.

When a host performs an object update after it detected a change in the PvD Option Sequence number, it MUST delay the query by a random time between zero and 2**(Delay * 2) milliseconds, where 'Delay' corresponds to the 4 bits long unsigned integer in the last received PvD Option.

When a host last retrieved an object at time A including a validity time B, and is configured to keep the object up to date, it MUST perform the update at a uniformly random time in the interval [(B-A)/2,B].

In the example Figure 2, the delay field value is 5, this means that host MUST delay the query by a random number between 0 and 2**(5 * 2) milliseconds, i.e., between 0 and 1024 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 value while changing the Delay value depending on the criticality of the update and its PvD Additional Information servers capacity.

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 the PIO is not covered by at least one of the listed prefixes, the PvD associated with the tested prefix MUST be considered unsafe and MUST NOT be used. While this does not prevent a malicious network provider, it does complicate some attack scenarios, and may help detecting misconfiguration.

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, PKI and HTTP over TLS operations related to the retrieval of the object, even before logging into the captive portal.

Routers MAY increment the PVD Option Sequence number in order to inform host that a new PvD Additional Information object is available and should be retrieved.

The server providing the JSON files SHOULD also check whether the client address is part of the prefixes listed into 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
name	Human-readable service name	UTF-8 string [RFC3629]	"Awesome Wifi"
expires	Date after which this object is not valid	[RFC3339]	"2017-07-23T06:00:00Z"
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 a valid string associated with the "name" key at the root of the object, or a valid date associated with the "expires" key, also at the root of the object, MUST be ignored. In such cases, an error message SHOULD be logged and/or displayed in a rate-limited fashion. 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
localizedName	Localized user-visible service name, language can be selected based on the HTTP Accept-Language header in the request.	UTF-8 string	"Wifi Genial"
dnsZones	DNS zones searchable and accessible	array of DNS zones	["example.com","sub.example.org"]
noInternet	No Internet, set when the PvD only provides restricted access to a set of services	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.

4.3.1. Private Extensions

JSON keys starting with "x-" are reserved for private use and can be utilized to provide information that is specific to vendor, user or enterprise. It is RECOMMENDED to use one of the patterns "x-FQDN-KEY" or "x-PEN-KEY" where FQDN is a fully qualified domain name or PEN is a private enterprise number [PEN] under control of the author of the extension to avoid collisions.

4.3.2. Example

Here are two examples based on the keys defined in this section.

```
{
  "name": "Foo Wireless",
  "localizedName": "Foo-France Wifi",
  "expires": "2017-07-23T06:00:00Z",
  "prefixes" : ["2001:db8:1::/48", "2001:db8:4::/48"],
}

{
  "name": "Bar 4G",
  "localizedName": "Bar US 4G",
  "expires": "2017-07-23T06:00:00Z",
  "prefixes": ["2001:db8:1::/48", "2001:db8:4::/48"],
}
```

4.4. Detecting misconfiguration and misuse

When a host retrieves the PvD Additional Information, it MUST verify that the TLS server certificate is valid for the performed request (e.g., that the Subject Name is equal to the PvD ID expressed as an FQDN). This authentication creates a secure binding between the information provided by the trusted Router Advertisement, and the HTTPS server. But this does not mean the Advertising Router and the PvD server belong to the same entity.

Hosts MUST verify that all prefixes in the RA PIO are covered by a prefix from the PvD Additional Information. An adversarial router willing to fake the use of a given explicit PvD, without any access to the actual PvD Additional Information, would need to perform NAT66 in order to circumvent this check.

It is also RECOMMENDED that the HTTPS server checks the IPv6 source addresses of incoming connections (see Section 4.1). This check give reasonable assurance that neither NPTv6 [RFC6296] nor NAT66 were used and restricts the information to the valid network users.

Note that this check cannot be performed when the HTTPS query is performed over IPv4. Therefore, the PvD ID FQDN SHOULD NOT have a DNS A record whenever all hosts using the given PvD have IPv6 connectivity.

5. Operational Considerations

This section describes some use cases of PvD. For the sake of simplicity, the RA messages will not be described in the usual ASCII art but rather in an indented list. For example, a RA message containing some options and a PvD option that also contains other options will be described as:

- RA Header: router lifetime = 6000
- Prefix Information Option: length = 4, prefix = 2001:db8:cafe::/64
- 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

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. For example, here is the RA for non-PvD-aware hosts:

- RA Header: router lifetime = 6000 (non-PvD-aware hosts will use this router as a default router)
- Prefix Information Option: length = 4, prefix = 2001:db8:cafe::/64
- Recursive DNS Server Option: length = 3, addresses= [2001:db8:cafe::53]
- 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 a RA example for PvD-aware hosts:

- RA Header: router lifetime = 0 (non-PvD-aware hosts will not use this router as a default router)

- PvD Option header: length = 3+ 2 + 4 + 3, PvD ID FQDN = 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 RA sent from 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 with this adress.

6. Security Considerations

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].

This specification does not improve the Neighbor Discovery Protocol security model, but extends the purely link-local trust relationship between the host and the default routers with HTTP over TLS communications which servers are authenticated as rightful owners of the FQDN received within the trusted PvD ID RA option.

It must be noted that Section 4.4 of this document only provides reasonable assurance against misconfiguration but does not prevent an hostile network access provider to advertize wrong information that

could lead applications or hosts to select an hostile PvD. Users should always apply caution when connecting to an unknown network.

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. Therefore, hosts willing to retrieve the PvD Additional Information before using it without leaking identity information, SHOULD make use of an IPv6 Privacy Address and SHOULD NOT include any privacy sensitive data, such as User Agent header or HTTP cookie, while performing the HTTP over TLS query.

From a 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 well known web sites, such as http://captive.example.com/hotspot-detect.html, in order to detect the presence of a captive portal.

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.

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).

IANA is asked to assign the value "pvd" from the Well-Known URIs registry.

IANA is asked to create and maintain a new registry entitled "Additional Information PvD Keys" containing ASCII strings. The initial content of this registry are given in Section 4.3; future assignments are to be made through Expert Review [BCP36].

Finally, IANA is asked to create and maintain a new registry entitled "PvD option Flags" reserving bit positions from 0 to 15 to be used in the PvD option bitmask. Bit position 0, 1 and 2 are reserved by this

document (as specified in Figure 1). Future assignments require a Standard Track RFC document.

9. Acknowledgements

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 Abrahamson, Ray Bellis, Zhen Cao, Tim Chow, Lorenzo Colitti, Ian Farrer, Bob Hinden, Tatuya Jinmei, Erik Kline, Ted Lemon, Jen Lenkova, Veronika McKillop, Mark Townsley and James Woodyatt for useful and interesting discussions.

Finally, special thanks to Thierry Danis and Wenqin Shao for their valuable inputs and implementation efforts ([github]), 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>.

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <https://www.rfc-editor.org/info/rfc1982>.

[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>.

[RFC2461] Narten, T., Nordmark, E., and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, DOI 10.17487/RFC2461, December 1998, <https://www.rfc-editor.org/info/rfc2461>.

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

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[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>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <https://www.rfc-editor.org/info/rfc7159>.

[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>.

10.2. Informative references

[github] Cisco, "IPv6-mPvD github repository", <https://github.com/IPv6-mPvD>.

[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".

[PEN] IANA, "Private Enterprise Numbers", <https://www.iana.org/assignments/enterprise-numbers>.

[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>.

[RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, DOI 10.17487/RFC3633, December 2003, <https://www.rfc-editor.org/info/rfc3633>.

[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>.

[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>.

[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>.

[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>.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <https://www.rfc-editor.org/info/rfc5785>.

[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>.

[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>.

[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>.

[RFC6731] Savolainen, T., Kato, J., and T. Lemon, "Improved Recursive DNS Server Selection for Multi-Interfaced Nodes", RFC 6731, DOI 10.17487/RFC6731, December 2012, <https://www.rfc-editor.org/info/rfc6731>.

[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>.

[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>.

[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>.

Appendix A. Changelog

Note to RFC Editors: Remove this section before publication.

A.1. Version 00

Initial version of the draft. Edited by Basile Bruneau + Eric Vyncke and based on Basile’s work.

A.2. Version 01

Major rewrite intended to focus on the the retained solution based on corridors, online, and WG discussions. Edited by Pierre Pfister. The following list only includes major changes.

PvD ID is an FQDN retrieved using a single RA option. This option contains a sequence number for push-based updates, a new H-flag, and a L-flag in order to link the PvD with the IPv4 DHCP server.

A lifetime is included in the PvD ID option.

Detailed Hosts and Routers specifications.

Additional Information is retrieved using HTTP-over-TLS when the PvD ID Option H-flag is set. Retrieving the object is optional.

The PvD Additional Information object includes a validity date.

DNS-based approach is removed as well as the DNS-based encoding of the PvD Additional Information.

Major cut in the list of proposed JSON keys. This document may be extended later if need be.

Monetary discussion is moved to the appendix.

Clarification about the ’prefixes’ contained in the additional information.

Clarification about the processing of DHCPv6.

A.3. Version 02

The FQDN is now encoded with ASCII format (instead of DNS binary) in the RA option.

The PvD ID option lifetime is removed from the object.

Use well known URI "https://<PvD-ID>/.well-known/pvd"

Reference RFC3339 for JSON timestamp format.

The PvD ID Sequence field has been extended to 16 bits.

Modified host behavior for DHCPv4 and DHCPv6.

Removed IKEv2 section.

Removed mention of RFC7710 Captive Portal option. A new I.D. will be proposed to address the captive portal use case.

A.4. WG Document version 00

Document has been accepted as INTAREA working group document

IANA considerations follow RFC8126 [RFC8126]

PvD ID FQDN is encoded as per RFC 1035 [RFC1035]

PvD ID FQDN is prepended by a one-byte length field

Marcus Keane added as co-author

dnsZones key is added back

draft of a privacy consideration section and added that a temporary address should be used to retrieve the PvD additional information

per Bob Hinden’s request: the document is now aiming at standard track and security considerations have been moved to the main section

A.5. WG Document version 01

Removing references to ’metered’ and ’characteristics’ keys. Those may be in scope of the PvD work, but this document will focus on essential parts only.

Removing appendix section regarding link quality and billing information.

The PvD RA Option may now contain other RA options such that PvD-aware hosts may receive configuration information otherwise invisible to non-PvD-aware hosts.

Clarify that the additional PvD Additional Information is not intended to modify host’s networking stack behavior, but rather provide information to the Application, used to select which PvDs must be used and provide configuration parameters to the transport layer.

The RA option padding is used to increase the option size to the next 64 (was 32) bits boundary.

Better detail the Security model and Privacy considerations.

A.6. WG Document version 02

Use the IANA value of 21 in the text and update the IANA considerations section accordingly

add the Delay field to avoid the thundering herd effect

add Wenqin Shao as author

keep the 1 PvD per RA model

changed the intro (per Zhen Cao) "when choosing which PvD and transport should be used" => "when choosing which PvD should be used"

rename A-flag in R-flag to avoid A-flag of PIO

use the wording "PvD Option", removing the ID token as it is now a container with more then just an ID, removing ’RA’ in the option name to be consistent with other IANA NDP option

use "non-PvD-aware" rather than "PvD-ignorant"

added more reference to RFC 7556 (notably for PvD being globally unique, introducing PvD-aware host vs. PvD-aware node)

Section 3.4.3 renamed from "interconnection shared by node" to ’connection shared by node"

Section 3.4 renamed into "PvD-aware Host Behavior"

Added a section "Non-PvD-aware Host Behavior"

Authors’ Addresses

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

Email: ppfister@cisco.com

Eric Vyncke (editor)
Cisco
De Kleetlaan, 6
Diegem 1831
Belgium

Email: evyncke@cisco.com

Tommy Pauly
Apple

Email: tpauly@apple.com

David Schinazi
Apple

Email: dschinazi@apple.com

Wenqin Shao
Telecom-ParisTech
France

Email: wenqin.shao@telecom-paristech.fr

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.

Table of Contents

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.

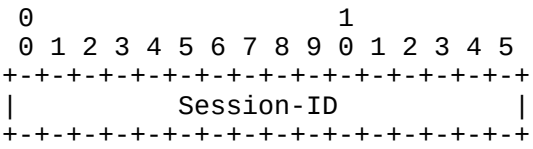
2. Captive Portal ICMP

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

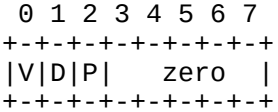
An unsigned short session identifier that groups ICMP messages. ICMP messages containing the same value MUST be assumed to be part of the same access policy. Any change in this value between ICMP messages from the same source IP address MUST be considered by the client to mean a change in access policy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



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:

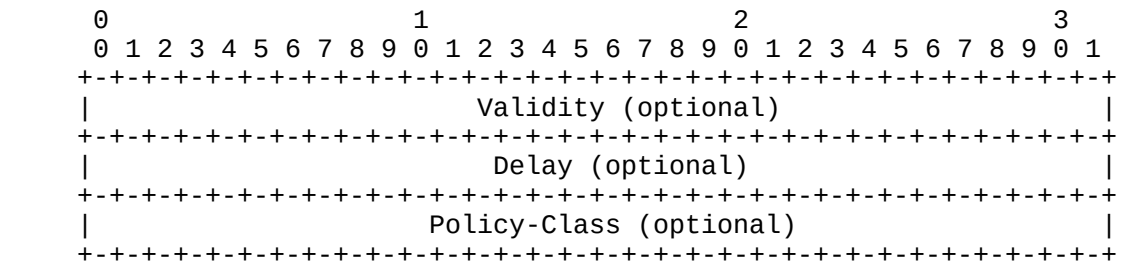


V - 1 bit Validity

D - 1 bit Delay

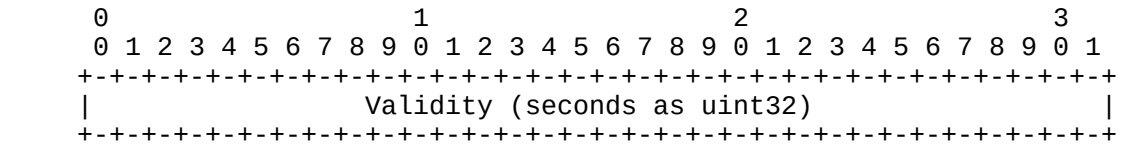
P - 1 bit Policy Class

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



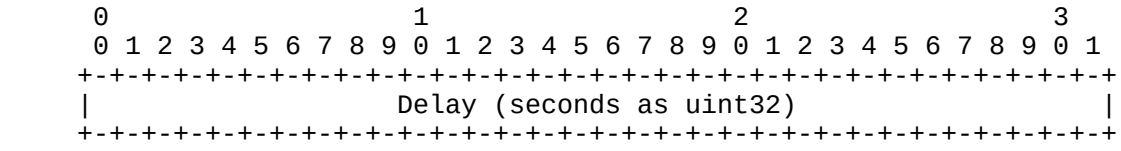
2.3. Validity

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.



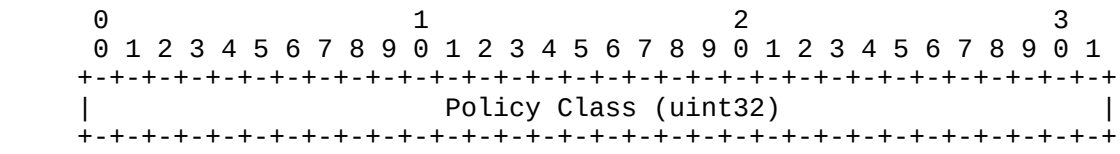
2.4. Delay

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.



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.



2.6. Message Code/C-Type

Captive Portal Message Code and C-Types:

0 General Change of Authorization (change in policy)

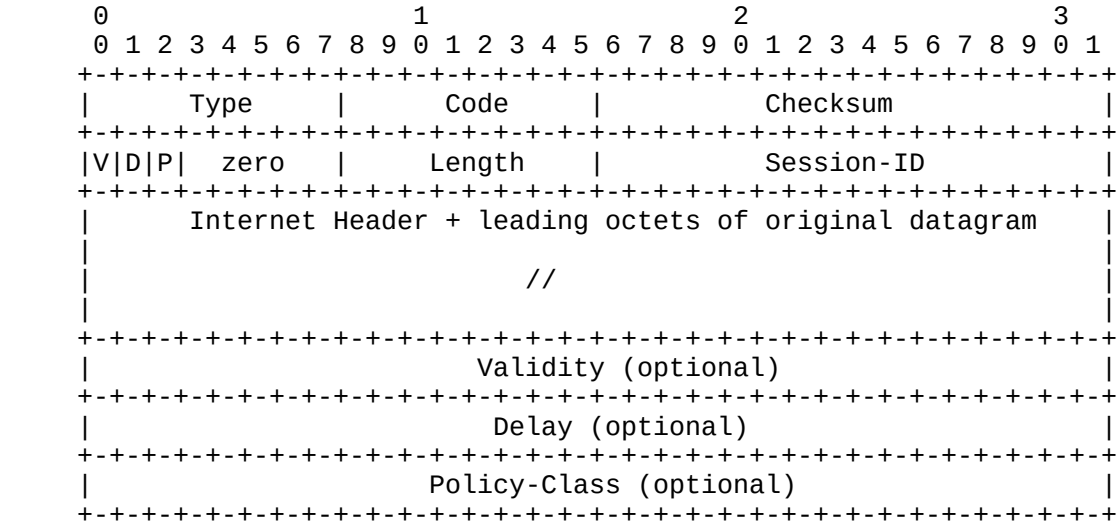
1 Packet/flow Error (dropped)

2 Packet/flow Overflow (dropped)

3 Packet/flow Warning (not dropped)

2.7. Message Type

The Captive Portal ICMP Type message is specifically for Capport ICMP Compliant devices. It is expected that Legacy devices will ignore such messages.



As shown in the figure above, the Captive Portal Flags and Session-ID and part of the ICMP header. The optional fields are in the ICMP payload, past the (optional) original datagram headers of a length defined by Length.

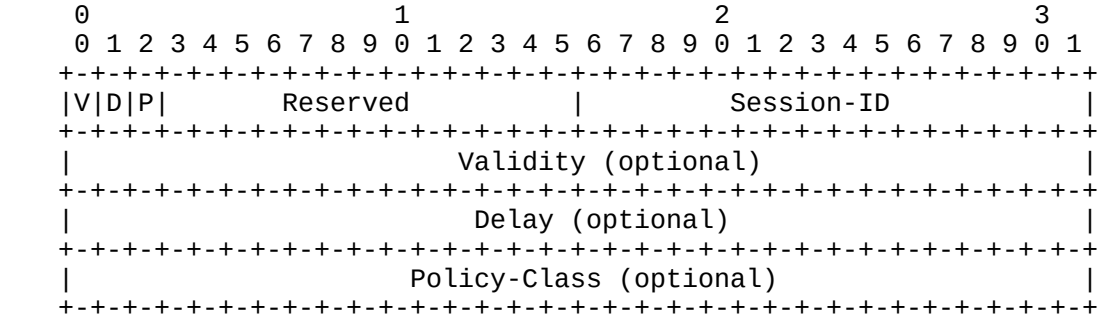
Length Number of 4 byte words of original datagram.

2.8. Extension Object

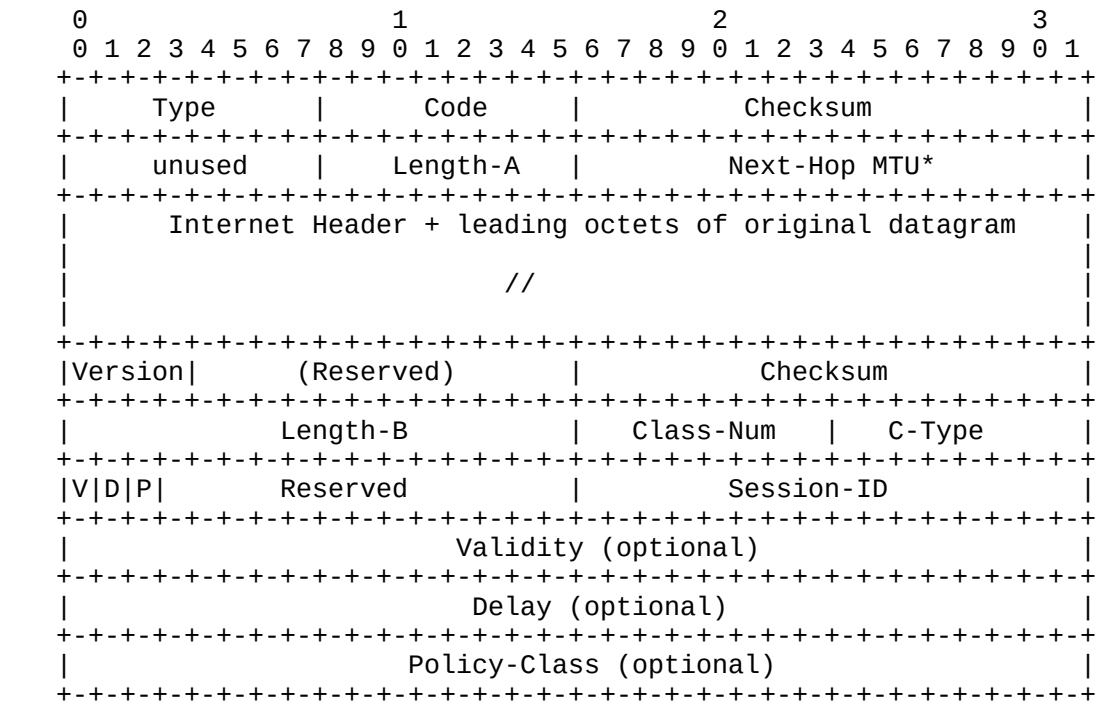
This document defines an extension object that can be appended to selected multi-part ICMP messages ([RFC4884]). This extension permits the CP-NAS to inform Capport ICMP Compliant devices that their connection has been blocked due to an Access Policy requiring interaction with the Captive Portal.

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.

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

From -00 to 01.

- Changed the Captive Portal URL to a URI, and specificed that this can ONLY contain a path element, which is appended to http://<gateway_ip>. This is to prevent hijacking connections to other addresses.

- Then removed the entire URL / URI scheme entirely.

From -genesis to -00.

- 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