

intarea
Internet-Draft
Intended status: Informational
Expires: January 29, 2018

P. Pfister, Ed.
Cisco
D. Schinazi
T. Pauly
Apple
E. Vyncke
Cisco
B. Bruneau
Ecole Polytechnique
July 28, 2017

Discovering Provisioning Domain Names and Data
draft-bruneau-intarea-provisioning-domains-02

Abstract

An increasing number of hosts and networks are connected to the Internet through multiple interfaces, some of which may provide multiple ways to access the internet by the mean of multiple IPv6 prefix configurations.

This document describes a way for hosts to retrieve additional information about their network access characteristics. The set of configuration items required to access the Internet is called a Provisioning Domain (PvD) and is identified by a Fully Qualified Domain Name (FQDN). This identifier, retrieved using a new Router Advertisement (RA) option, is associated with the set of information included within the RA and may later be used to retrieve additional information associated with the PvD by the mean of an HTTP request.

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 January 29, 2018.

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	3
2. Terminology	3
3. Provisioning Domain Identification using Router Advertisements	4
3.1. PvD ID Option for Router Advertisements	4
3.2. Router Behavior	5
3.3. Host Behavior	5
3.3.1. DHCPv6 configuration association	6
3.3.2. DHCPv4 configuration association	7
3.3.3. Interconnection Sharing by the Host	7
4. Provisioning Domain Additional Information	7
4.1. Retrieving the PvD Additional Information	7
4.2. Providing the PvD Additional Information	9
4.3. PvD Additional Information Format	9
4.3.1. Connectivity Characteristics Information	10
4.3.2. Private Extensions	11
4.3.3. Example	11
5. Security Considerations	11
6. Privacy Considerations	12
7. IANA Considerations	12
8. Acknowledgements	12
9. References	13
9.1. Normative references	13
9.2. Informative references	13
Appendix A. Changelog	15
A.1. Version 00	15
A.2. Version 01	15
A.3. Version 02	16
Appendix B. Connection monetary cost	16
B.1. Conditions	17
B.2. Price	17

B.3. Examples	18
Authors' Addresses	19

1. Introduction

It has become very common in modern networks that hosts have internet or more specific network access through different networking interfaces, tunnels, or next-hop routers. The concept of Provisioning Domain (PvD) was defined in [RFC7556] as a set of network configuration information which can be used by hosts in order to access the network.

This specification provides a way to identify explicit PvDs with Fully Qualified Domain Names called PvD IDs, which are included in a new Router Advertisement [RFC4861] option. This new option, when present, is used to associate the correlated set of configuration information with the identified PvD. It is worth noting that multiple PvDs with different PvD IDs could be provisioned on any host interface, as well as noting that the same PvD ID could be used on different interfaces in order to inform the host that both PvDs, on different interfaces, ultimately provide identical services.

This document also introduces a way for hosts to retrieve additional information related to a specific PvD by the mean of an HTTP-over-TLS query using an URI derived from the PvD ID. The retrieved JSON object contains additional network information that would typically be considered unfit, or too large, to be directly included in the Router Advertisements. This information can be used by the networking stack, the applications, or even be partially displayed to the users (e.g., by displaying a localized network service name).

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:

PvD: A Provisioning Domain, 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].

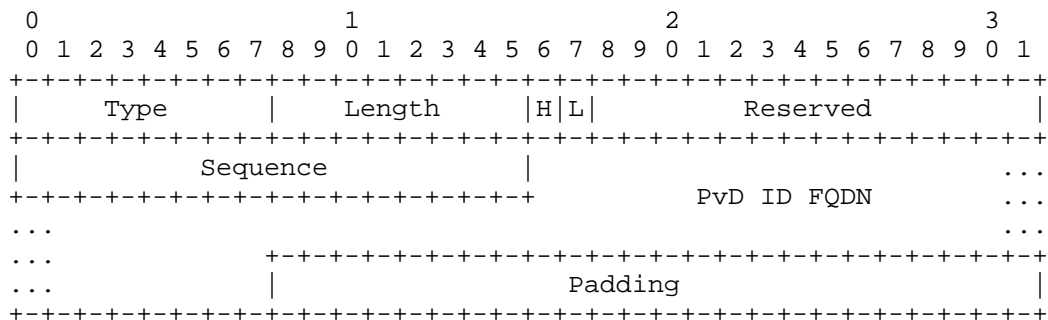
Implicit PvD: A PvD associated with a set of configuration information that, in the absence of a PvD ID, is associated with the advertising router.

3. Provisioning Domain Identification using Router Advertisements

Each provisioning domain is 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 ambiguity. The same PvD ID MAY be used in several access networks when the set of configuration information is identical (e.g. in all home networks subscribed to the same service).

3.1. PvD ID Option for Router Advertisements

This document introduces a new Router Advertisement (RA) option called the PvD ID Router Advertisement Option, used to convey the FQDN identifying a given PvD.



PvD ID Router Advertisements Option format

Type : (8 bits) To be defined by IANA.

Length : (8 bits) The length of the option (including the Type and Length fields) in units of 8 octets.

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

L-flag : (1 bit) Whether the router is also providing IPv4 access using DHCPv4 (see Section 3.3.2).

Reserved : (14 bits) Reserved for later use. It MUST be set to zero by the sender and ignored by the receiver.

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

PvD ID FQDN : An ASCII string representation of the FQDN used as PvD ID. The string ends at the first byte set to zero, or the end of the option, whichever comes first.

Padding : Zero or more padding octets such as to set the option length (Type and Length fields included) to eight times the value of the Length field. It MUST be set to zero by the sender and ignored by the receiver.

Routers MUST NOT include more than one PvD ID Router Advertisement Option in each RA. In case multiple PvD ID options are found in a given RA, hosts MUST ignore all but the first PvD ID option.

Note: The existence and/or size of the sequence number is subject to discussion. The validity of a PvD Additional Information object is included in the object itself, but this only allows for 'pull based' updates, whereas the RA options usually provide 'push based' updates.

3.2. Router Behavior

A router MAY insert at most one PvD ID Option in its RAs. The included PvD ID is associated with all the other options included in the same RA (e.g., Prefix Information [RFC4861], Recursive DNS Server [RFC6106], Routing Information [RFC4191] options).

In order to provide multiple independent PvDs, a router MUST send multiple RAs using different source link-local addresses (LLA) (as proposed in [I-D.bowbakova-rtgwg-enterprise-pa-multihoming]), each of which MAY include a PvD ID option. In such cases, routers MAY originate the different RAs using the same datalink layer address.

If the router is actually a VRRP instance [RFC5798], then the procedure is identical except that the virtual datalink layer address is used as well as the virtual IPv6 addresses.

3.3. Host Behavior

RAs are used to configure IPv6 hosts. When a host receives an RA message including a PvD ID Option, it MUST associate all the configuration objects which are updated by the received RA (e.g., Prefix Information [RFC4861], Recursive DNS Server [RFC6106], Routing Information [RFC4191] options) with the PvD identified by the PvD ID Option, even if some objects are already associated with a different explicit or implicit PvD.

If the received RA does not include a PvD ID Option, the host MUST associate the configuration objects which are updated by the received RA with an implicit PvD, even if some objects were already associated

with a different explicit or implicit PvD. This implicit PvD is identified by the link-local address of the router sending the RA and the interface on which the RA was received.

This document does not update the way Router Advertisement options are processed. But in addition to the option processing defined in other documents, hosts implementing this specification **MUST** associate each created or updated object (e.g. address, default route, more specific route, DNS server list) with the PvD associated with the received RA.

Note: There is a discussion whether there can be multiple implicit PvDs on a single interface (i.e. whether the router link-local address should be used to identify the implicit PvDs).

While resolving names, executing the default address selection algorithm [RFC6724] or executing the default router selection algorithm ([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 for a given connection. 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].

3.3.1. DHCPv6 configuration association

When a host retrieves configuration elements using DHCPv6, they **MUST** be associated with the explicit or implicit PvD of the RA received on the same interface, using the same link-local address, and with the O-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 link-local addresses.

3.3.2. 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 ID Options L-flag is set and, in the case of a non point-to-point link, using the same link-layer address. If no such PvD is found, or whenever multiple different PvDs are found, the configuration elements coming from DHCPv4 MUST be associated with an IPv4-only implicit PvD identified by the interface on which the DHCPv4 transaction happened.

3.3.3. Interconnection Sharing by the Host

The situation when a host becomes also a router by acting as a router or ND proxy on a different interface (such as WiFi) to share the connectivity of another interface (such as cellular), also known as "tethering" is TBD but it is expected that the one or several PvD associated to the shared interface will also be advertised to the clients.

4. Provisioning Domain Additional Information

Once a new PvD ID is discovered, it may be used to retrieve additional information about the characteristics of the provided connectivity. 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 hosts 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 3.3.

4.1. Retrieving the PvD Additional Information

When the H-flag of the PvD ID 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: Should the PvD AI retrieval be a MAY or a SHOULD ? Could the object contain critical data, or should it only contain informational data ?

Note that the DNS name resolution of <PvD-ID> as well as the actual query MUST be performed using the PvD associated with the PvD ID. In other words, the name resolution, 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 3.3. 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 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 included and 399 included it MUST follow the redirection(s). If the HTTP status of the answer is between 200 included and 299 included 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 watch the PvD ID Sequence field for change. In case a different value than the one in the RA Sequence field is observed, 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 deprecate the object and stop using it.

Hosts retrieving a new PvD Additional Information object MUST check for the presence and validity of the mandatory fields Section 4.3. A retrieved object including an outdated expiration time or missing a mandatory element MUST be ignored. In order to avoid traffic spikes toward the server hosting the PvD Additional Information when an object expires, a host which last retrieved an object at a time A, including a validity time B, SHOULD renew the object at a uniformly random time in the interval $[(B-A)/2, A]$.

The PvD Additional Information object includes a set of IPv6 prefixes which MUST be checked against all the Prefix Information Options advertised in the Router Advertisement. If any of the prefixes included in the Prefix Information Options is not included in 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.

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. The server MAY also use the client address to select the right JSON object to be returned.

4.2. Providing the PvD Additional Information

Whenever the H-flag is set in the PvD RA Option, a valid PvD Additional Information object MUST be made available to all hosts receiving the RA. In particular, when a captive portal is present, hosts MUST still be allowed to access the object, even before logging into the captive portal.

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

4.3. PvD Additional Information Format

The PvD Additional Information is a JSON object.

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

JSON key	Description	Type	Example
name	Human-readable service name	UTF-8 string	"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 "expiration" 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.

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"
noInternet	No Internet, set when the PvD only provides restricted access to a set of services.	boolean	true
characteristics	Connectivity characteristics	JSON object	See Section 4.3.1
metered	metered, when the access volume is limited.	boolean	false

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. Connectivity Characteristics Information

The following set of keys can be used to signal certain characteristics of the connection towards the PvD.

They should reflect characteristics of the overall access technology which is not limited to the link the host is connected to, but rather a combination of the link technology, CPE upstream connectivity, and further quality of service considerations.

JSON key	Description	Type	Example
maxThroughput	Maximum achievable throughput	object({down(int), up(int)}) in kb/s	{"down": 10000, "up": 5000}
minLatency	Minimum achievable latency	object({down(int), up(int)}) in ms	{"down": 10, "up": 20}
rl	Maximum achievable reliability	object({down(int), up(int)}) in losses every 1000 packets	{"down": 0.1, "up": 1}

4.3.2. 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.

4.3.3. 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"],
  "characteristics": {
    "maxThroughput": { "down":200000, "up": 50000 },
    "minLatency": { "down": 0.1, "up": 1 }
  }
}

{
  "name": "Bar 4G",
  "localizedName": "Bar US 4G",
  "expires": "2017-07-23T06:00:00Z",
  "prefixes": ["2001:db8:1::/48", "2001:db8:4::/48"],
  "metered": true,
  "characteristics": {
    "maxThroughput": { "down":80000, "up": 20000 }
  }
}
```

5. Security Considerations

Although some solutions such as IPsec or SEND [RFC3971] can be used in order to secure the IPv6 Neighbor Discovery Protocol, 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 configuration retrieval mechanisms with HTTP-over-TLS communications.

During the exchange, the server authenticity is verified by the mean of a certificate, validated based on the FQDN found in the Router Advertisement (e.g. using a list of pre-installed CA certificates, or DNSSEC [RFC4035] with DNS Based Authentication of Named Entities [RFC6698]). This authentication creates a secure binding between the information provided by the trusted Router Advertisement, and the HTTP server. But this does not mean the Advertising Router and the PvD server belong to the same entity.

The IPv6 prefixes list included in the PvD Additional Information JSON object is used to validate that the prefixes included in the Router Advertisements are really part of the PvD. An adversarial router willing to fake the use of a given explicit PvD, without any access to the actual PvD, would need to perform NAT66 in order to circumvent this check.

It is also RECOMMENDED that the PvD server checks the source addresses of incoming connexions (see Section 4.1). This check ensures that the internet access provided by any router advertising a given PvD eventually reaches the internet using the actual PvD (Tunneling can still be used).

For privacy reasons, it is desirable that the PvD Additional Information object may only be retrieved by the hosts using the given PvD. Host identity SHOULD be validated based on the client address that is used during the HTTP query.

6. Privacy Considerations

TBD

7. IANA Considerations

IANA is kindly requested to allocate a new IPv6 Neighbor Discovery option number for the PvD ID Router Advertisement option.

The URI used to retrieve the PvD Additional Information JSON object is the well known URI (see [RFC5785]) with the URI suffix "pvd".

TBD: JSON keys will need a new registry.

8. Acknowledgements

Many thanks to M. Stenberg and S. Barth for their earlier work: [I-D.stenberg-mif-mpvd-dns].

Thanks also to Ray Bellis, Lorenzo Colitti, Thierry Danis, Marcus Keane, Erik Kline, Jen Lenkova, Mark Townsley, James Woodyatt and Mikael Abrahamson for useful and interesting discussions.

Finally, many thanks to Thierry Danis for his implementation work ([github]), Tom Jones for his integration effort into the Neat project and Rigil Salim for his implementation work.

9. References

9.1. Normative references

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2461] Narten, T., Nordmark, E., and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, December 1998.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <<http://www.rfc-editor.org/info/rfc2818>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <<http://www.rfc-editor.org/info/rfc7159>>.

9.2. Informative references

- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<http://www.rfc-editor.org/info/rfc3339>>.
- [RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.
- [RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <<http://www.rfc-editor.org/info/rfc4035>>.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.

- [RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <<http://www.rfc-editor.org/info/rfc5785>>.
- [RFC5798] Nadas, S., Ed., "Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6", RFC 5798, DOI 10.17487/RFC5798, March 2010, <<http://www.rfc-editor.org/info/rfc5798>>.
- [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, <<http://www.rfc-editor.org/info/rfc6105>>.
- [RFC6106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 6106, November 2010.
- [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <<http://www.rfc-editor.org/info/rfc6698>>.
- [RFC6724] Thaler, D., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.
- [RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <<http://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, <<http://www.rfc-editor.org/info/rfc8028>>.
- [I-D.bowbakova-rtgwg-enterprise-pa-multihoming]
Baker, F., Bowers, C., and J. Linkova, "Enterprise Multihoming using Provider-Assigned Addresses without Network Prefix Translation: Requirements and Solution", draft-bowbakova-rtgwg-enterprise-pa-multihoming-01 (work in progress), October 2016.
- [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.

- [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.
- [PEN] IANA, "Private Enterprise Numbers", <<https://www.iana.org/assignments/enterprise-numbers>>.
- [IEEE8021X]
IEEE, "IEEE Standards for Local and Metropolitan Area Networks: Port based Network Access Control, IEEE Std", .
- [github] Cisco, "IPv6-mPvD github repository", <<https://github.com/IPv6-mPvD>>.

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.

Appendix B. Connection monetary cost

NOTE: This section is included as a request for comment on the potential use and syntax.

The billing of a connection can be done in a lot of different ways. The user can have a global traffic threshold per month, after which his throughput is limited, or after which he/she pays each megabyte. He/she can also have an unlimited access to some websites, or an unlimited access during the weekends.

An option is to split the bill in elementary billings, which have conditions (a start date, an end date, a destination IP address...). The global billing is an ordered list of elementary billings. To know the cost of a transmission, the host goes through the list, and the first elementary billing whose conditions are fulfilled gives the cost. If no elementary billing conditions match the request, the host MUST make no assumption about the cost.

B.1. Conditions

Here are the potential conditions for an elementary billing. All conditions MUST be fulfill.

Key	Description	Type	JSON Example
beginDate	Date before which the billing is not valid	ISO 8601	"1977-04-22T06:00:00Z"
endDate	Date after which the billing is not valid	ISO 8601	"1977-04-22T06:00:00Z"
domains	FQDNs whose the billing is limited	array(string)	["deezer.com", "spotify.com"]
prefixes4	IPv4 prefixes whose the billing is limited	array(string)	["78.40.123.182/32", "78.40.123.183/32"]
prefixes6	IPv6 prefixes whose the billing is limited	array(string)	["2a00:1450:4007:80e::200e/64"]

B.2. Price

Here are the different possibilities for the cost of an elementary billing. A missing key means "all/unlimited/unrestricted". If the elementary billing selected has a trafficRemaining of 0 kb, then it means that the user has no access to the network. Actually, if the last elementary billing has a trafficRemaining parameter, it means that when the user will reach the threshold, he/she will not have access to the network anymore.

Key	Description	Type	JSON Example
pricePerGb	The price per Gigabit	float (currency per Gb)	2
currency	The currency used	ISO 4217	"EUR"
throughputMax	The maximum achievable throughput	float (kb/s)	100000
trafficRemaining	The traffic remaining	float (kB)	12000000

B.3. Examples

Example for a user with 20 GB per month for 40 EUR, then reach a threshold, and with unlimited data during weekends and to example.com:

```
[
  {
    "domains": ["example.com"]
  },
  {
    "prefixes4": ["78.40.123.182/32", "78.40.123.183/32"]
  },
  {
    "beginDate": "2016-07-16T00:00:00Z",
    "endDate": "2016-07-17T23:59:59Z",
  },
  {
    "beginDate": "2016-06-20T00:00:00Z",
    "endDate": "2016-07-19T23:59:59Z",
    "trafficRemaining": 12000000
  },
  {
    "throughputMax": 100000
  }
]
```

If the host tries to download data from example.com, the conditions of the first elementary billing are fulfilled, so the host takes this elementary billing, finds no cost indication in it and so deduces that it is totally free. If the host tries to exchange data with foobar.com and the date is 2016-07-14T19:00:00Z, the conditions of the first, second and third elementary billing are not fulfilled.

But the conditions of the fourth are. So the host takes this elementary billing and sees that there is a threshold, 12 GB are remaining.

Another example for a user abroad, who has 3 GB per year abroad, and then pay each MB:

```
[
  {
    "beginDate": "2016-02-10T00:00:00Z",
    "endDate": "2017-02-09T23:59:59Z",
    "trafficRemaining": 3000000
  },
  {
    "pricePerGb": 30,
    "currency": "EUR"
  }
]
```

Authors' Addresses

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

Email: ppfister@cisco.com

David Schinazi
Apple

Email: dschinazi@apple.com

Tommy Pauly
Apple

Email: tpauly@apple.com

Eric Vyncke
Cisco
De Kleetlaan, 6
Diegem 1831
Belgium

Email: evyncke@cisco.com

Basile Bruneau
Ecole Polytechnique
Vannes 56000
France

Email: basile.bruneau@polytechnique.edu

Captive Portal WG
Internet-Draft
Intended status: Informational
Expires: January 4, 2018

M. Donnelly
M. Cullen
Painless Security
July 3, 2017

Captive Portal (CAPPORT) API
draft-donnelly-capport-detection-02

Abstract

This document describes an HTTP API that allows User Equipment to detect the existence of a Captive Portal on the local network and determine the properties of the Captive Portal.

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 January 4, 2018.

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
2. Requirements Notation	2
3. Workflow	3
4. Use of the DHCP Captive-Portal Option	3
5. CAPPORT API	3
5.1. URLs and HTTP Methods	4
5.1.1. Associating User Equipment with its URL	4
5.1.2. Interactive URL	4
5.1.3. CAPPORT API POST URL	4
5.2. JSON Data Structures	4
5.2.1. CAPPORT Common Elements	4
5.2.1.1. Toplevel Object	4
5.2.1.2. Networks Object	5
5.2.1.3. Network Object	5
5.2.2. User Equipment Request	6
5.2.2.1. CAPPORT API Server Response	6
5.2.2.1.1. Network State Object	6
6. IANA Considerations	7
7. Security Considerations	7
7.1. Privacy Considerations	7
8. Acknowledgements	7
9. References	7
9.1. Normative References	7
9.2. Informative References	8
Authors' Addresses	8

1. Introduction

This document describes a HyperText Transfer Protocol (HTTP) Application Program Interface (API) that allows User Equipment to detect the existence of a Captive Portal (CAPPORT) on the local network and determine the properties of the Captive Portal. The API defined in this document has been designed to meet the requirements of the CAPPORT API, as discussed in the CAPPORT Architecture [I-D.larose-capport-architecture].

2. 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].

3. Workflow

The CAPPORT protocol consists of two phases. In the first phase User Equipment acquires an IP address and determines the URL of the local CAPPORT API Server, if any. The second phase consists of the User Equipment querying the CAPPORT API Server for the requirements for accessing its protected networks.

During the first phase, User Equipment uses the Dynamic Host Configuration Protocol (DHCP) or IPv6 Router Advertisements (RAs) to acquire an IP address and to determine the URL for the local CAPPORT API Server. This details for the first phase are described in RFC 7710[RFC7710], and the rest of this document assumes that the User Equipments already has a URL to reach the CAPPORT API Server.

The second phase begins with the User Equipment accessing the URL provided in the first phase with JSON content type. The CAPPORT API Server responds with the current status of the User Equipment's access to the protected networks.

The User Equipment SHOULD request the same URL with an HTML content type to start the process to gain access to the Captive Portal.

4. Use of the DHCP Captive-Portal Option

As described above, to use the CAPPORT API, User Equipment needs a URL that can be used to reach the CAPPORT API Server. DHCP Servers and IPv6 Routers SHOULD provide, and User Equipment SHOULD obtain, the required URL using the DHCP Captive-Portal Option or the IPv6 RA Captive-Portal Option, as described in [RFC7710].

To provide backwards compatibility with the original use of the DHCP and RA options described in RFC7710, the CAPPORT API defined in this document is exclusively accessed using HTTP with an Accept header value of "application/json". Captive Portals that implement the CAPPORT API SHOULD respond to an HTTP GET that has an Accept header of "text/html" with HTML content that, when displayed in a web browser, will allow the user to interactively meet the Captive Portal requirements for network access.

5. CAPPORT API

This section defines the CAPPORT API.

5.1. URLs and HTTP Methods

This section describes the URLs that can be used to access the CAPPORT API.

5.1.1. Associating User Equipment with its URL

The CAPPORT API Server SHOULD associate an incoming request with a particular User Equipment consistently. [TODO: specify how this would happen.]

5.1.2. Interactive URL

The CAPPORT API Server SHOULD respond to HTTP GET requests to the provided URL that specify an Accept header value of "text/html" with HTML content instead of this protocol. When the User Equipment wants to satisfy the conditions for network access, it SHOULD display this interactive URL in a web browser to allow the user to complete the network access outside of this protocol.

5.1.3. CAPPORT API POST URL

The CAPPORT API Server SHOULD respond to HTTP POST requests to the provided URL that specify an Accept header value of "application/json" with the CAPPORT API protocol.

5.2. JSON Data Structures

The CAPPORT API data structures are specified in JavaScript Object Notation (JSON)[RFC7159]. This document specifies the structure of the JSON structures and message using the JSON Content Rules (JCR) defined in draft-newton-json-content-rules [I-D.newton-json-content-rules].

5.2.1. CAPPORT Common Elements

This section describes structures that are shared between requests and responses.

5.2.1.1. Toplevel Object

The CAPPORT API will contain JSON-formatted data. The toplevel object contains a networks object whose value is an array of zero or more network objects.


```
$toplevel = {  
  $networks ,  
  $session_token ?  
}
```

The `toplevel` object MUST contain a `networks` object.

The CAPPORT API Server responses MUST contain a `session_token` object. The session-token object contains a session token which will be used in ICMP requests as discussed in RFC 7710.

QUESTION: Should the session token just be provided by the server, or should it be negotiated between the client and server using something like a DH exchange?

5.2.1.2. Networks Object

The `networks` object represents the list of networks being acted on in this CAPPORT session.

```
$networks = {  
  ( "DEFAULT" || // ) = $network +  
}
```

The `networks` object is a JSON object whose keys are network names and whose values are network objects. Thus a single response could be used in gaining access to multiple protected networks at once. The first request to the CAPPORT API Server will contain no networks, and acts as a discovery request.

The CAPPORT API Server SHOULD use the special name `DEFAULT` for one network that provides access to the greater Internet.

5.2.1.3. Network Object

The `network` object represents a network protected by the Captive Portal.

```
$network = {  
  "conditions" : [ $condition + ] ,  
  "state" : $network_state ? ,  
  "details" : $network_details ?  
}
```

The `network` object MUST contain a 'conditions' key whose value is an array of one or more `$condition` objects, which represent the unmet conditions for gaining access to this network. The conditions object SHOULD NOT contain conditions that have already been met.

CAPPORT API Server responses MUST contain the 'state' key, whose value is the \$network_state object, which represents the state of access that the User Equipment has to the network.

CAPPORT API Server responses SHOULD contain the 'details' key, whose value is the \$network_details object, which provides relevant information about the network.

5.2.2. User Equipment Request

For the initial CAPPORT request from the User Equipment, the JSON object will consist of the toplevel object (Section 5.2.1.1) with its required networks (Section 5.2.1.2) objects. The networks object will contain no networks, and the session_token object will be empty. This acts as a discovery request.

```
{
  "networks" : {}
  "session-token" : ""
}
```

Figure 1

5.2.2.1. CAPPORT API Server Response

5.2.2.1.1. Network State Object

The network_state object details the current state of the User Equipment access to the protected network.

```
$network_state = {
  "permitted" : boolean ,
  "expires" : datetime ? ,
  "bytes_remaining" : integer ?
}
```

The network_state object MUST contain the "permitted" key, whose boolean value indicates whether the User Equipment is permitted to access the protected network.

The network_state object SHOULD contain the "expires" key if the access to the protected network will expire at a known time in the future. The value is a datetime object of the time the access will expire. If there is not a known expiration time, the key SHOULD be omitted.

The network_state object SHOULD contain the "bytes_remaining" key if the access to the protected network will expire after the User

Equipment transfers a known number of bytes. The value is an integer of the number of bytes remaining. If there is not a known limit for this User Equipment, the key MAY be omitted or its value MAY be -1.

6. IANA Considerations

This document does not require any IANA allocations. Please remove this section before RFC publication.

7. Security Considerations

The CAPPORT API described in this document is intended to automate a process that is currently accomplished by a user filling out a HTML form in a Web Browser. Therefore, this mechanism should meet the requirement of being no less secure than presenting the user with a HTML form for completion in a Web Browser, and submitting that form to a Captive Portal.

TBD: Provide complete security requirements and analysis.

7.1. Privacy Considerations

Information passed in this protocol may include a user's personal information, such as a full name and credit card details. Therefore, it is important that CAPPORT API Servers do not allow access to the CAPPORT API over unencrypted sessions.

8. Acknowledgements

This document was written using xml2rfc, as described in [RFC7749]

9. References

9.1. Normative References

[I-D.larose-capport-architecture]

Larose, K. and D. Dolson, "CAPPORT Architecture", draft-larose-capport-architecture-01 (work in progress), June 2017.

[I-D.newton-json-content-rules]

Newton, A. and P. Cordell, "A Language for Rules Describing JSON Content", draft-newton-json-content-rules-08 (work in progress), March 2017.

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

9.2. Informative References

- [RFC7749] Reschke, J., "The "xml2rfc" Version 2 Vocabulary", RFC 7749, DOI 10.17487/RFC7749, February 2016, <<http://www.rfc-editor.org/info/rfc7749>>.

Authors' Addresses

Mark Donnelly
Painless Security
14 Summer Street, Suite 202
Malden, MA 02148
USA

Email: mark@painless-security.com
URI: <http://www.painless-security.com>

Margaret Cullen
Painless Security
14 Summer Street, Suite 202
Malden, MA 02148
USA

Phone: +1 781 405-7464
Email: margaret@painless-security.com
URI: <http://www.painless-security.com>

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: December 31, 2017

K. Larose
D. Dolson
Sandvine
June 29, 2017

CAPPORT Architecture
draft-larose-capport-architecture-01

Abstract

This document aims to document consensus on the CAPPORT architecture. DHCP or Router Advertisements, ICMP, 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 <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 December 31, 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 Language	4
1.2. Terminology	4
2. Components	4
2.1. User Equipment	5
2.2. Provisioning Service	5
2.2.1. DHCP or Router Advertisements	6
2.2.2. Provisioning Domains	6
2.3. Captive Portal API Server	6
2.4. Captive Portal Enforcement	7
2.5. ICMP/ICMP6	8
2.6. Component Diagram	8
3. Solution Workflow	10
3.1. Initial Connection	10
3.2. Conditions Expire	10
4. Acknowledgements	11
5. IANA Considerations	11
6. Security Considerations	11
6.1. Trusting the Network	11
6.2. Authenticated APIs	12
6.3. Risk of Nuisance Captive Portal	12
6.4. User Options	13
7. References	13
7.1. Normative References	13
7.2. Informative References	13
Authors' Addresses	14

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 fulfil 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:

- o 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.
- o 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.
- o Solutions SHOULD allow a device to be alerted that it is in a captive network when attempting to use any application on the network. (Versus requiring a user to visit a clear-text HTTP site in order to receive a notification.)
- o 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).
- o The architecture MUST provide a path of incremental migration, acknowledging a huge variety of portals and end-user device implementations and software versions.
- o The architecture SHOULD improve security by providing mechanisms for trust, allowing alerts from trusted network operators to be distinguished from attacks from untrusted agents.

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:

- o 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.bruneau-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 are notified of captivity with ICMP/ICMP6 messages in response to traffic. This notification can 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.
- o Receipt of the ICMP/ICMP6 messages informs an end-user device that it is captive. In response, the device SHOULD 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 and notifies the User Equipment it is in a captive state.

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.

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.

- o An example interactive User Equipment is a smart phone.
- o SHOULD support provisioning of the URI for the Captive Portal API (e.g., by DHCP)
- o MAY distinguish Captive Portal API access per network interface, in the manner of Provisioning Domain Architecture [RFC7556].
- o SHOULD have a mechanism for notifying the user of the Captive Portal
- o SHOULD have a web browser so that the user may navigate the Captive Portal user interface.
- o SHOULD be able to receive and validate the Captive Portal ICMP message types, and to access the Captive Portal API in response.
- o 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.bruneau-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.

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) an ICMP Captive Portal message (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 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 (and validates) ICMP Captive Portal alerts, the user equipment SHOULD query the API to check the state. The User Equipment SHOULD rate-limit these API queries in the event of ICMP flooding by an attacker. (See Section 6.)

The API MUST be extensible to support future use-cases by allowing extensible information elements. Suggestions include quota information, expiry time, method of providing credentials, security token for validating ICMP messages.

This document does not specify the details of the API.

The CAPPORT API SHOULD support TLS for privacy and server authentication.

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:

- o Allows traffic through for allowed User Equipment.
- o Blocks (discards) traffic and sends ICMP notifications for disallowed User Equipment.
- o Permits disallowed User Equipment to access necessary APIs and web pages to fulfill requirements of exiting captivity.
- o Updates allow/block rules per User Equipment in response to operations from the Captive Portal back-end.

As an upgrade path, captive portals MAY continue to support methods that work today, such as modification of port-80 HTTP responses to

redirect users to the portal. Various user-equipment vendors probe canary URLs to identify the state captivity [reference Mariko Kobayashi's survey]. While doing so, ICMP messages SHOULD also be sent, to activate work-flows in supporting devices. [TODO: give some thought to precise recommendations for backwards compatibility.]

2.5. ICMP/ICMP6

ICMP messages have been selected for indicating to a sender that packets could not be delivered for reason of a network policy (in particular, captive portal). ICMP is already used to indicate reasons that packets could not be delivered (network unreachable, port unreachable, packet too large, etc.).

A mechanism to trigger captive portal work-flows in the User Equipment is proposed in [I-D.wkumari-capport-icmp-unreach].

The Captive Portal Enforcement function is REQUIRED to send such ICMP messages when disallowed User Equipment attempts to send to the network. These ICMP messages MUST be rate-limited to a configurable rate.

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

The User Equipment operating system is NOT REQUIRED to deliver the impact of the ICMP message to the application that triggered it. The User Equipment might be able to satisfy the Captive Portal requirements quickly enough that existing transport connections are not impacted.

2.6. Component Diagram

The following diagram shows the communication between each component.

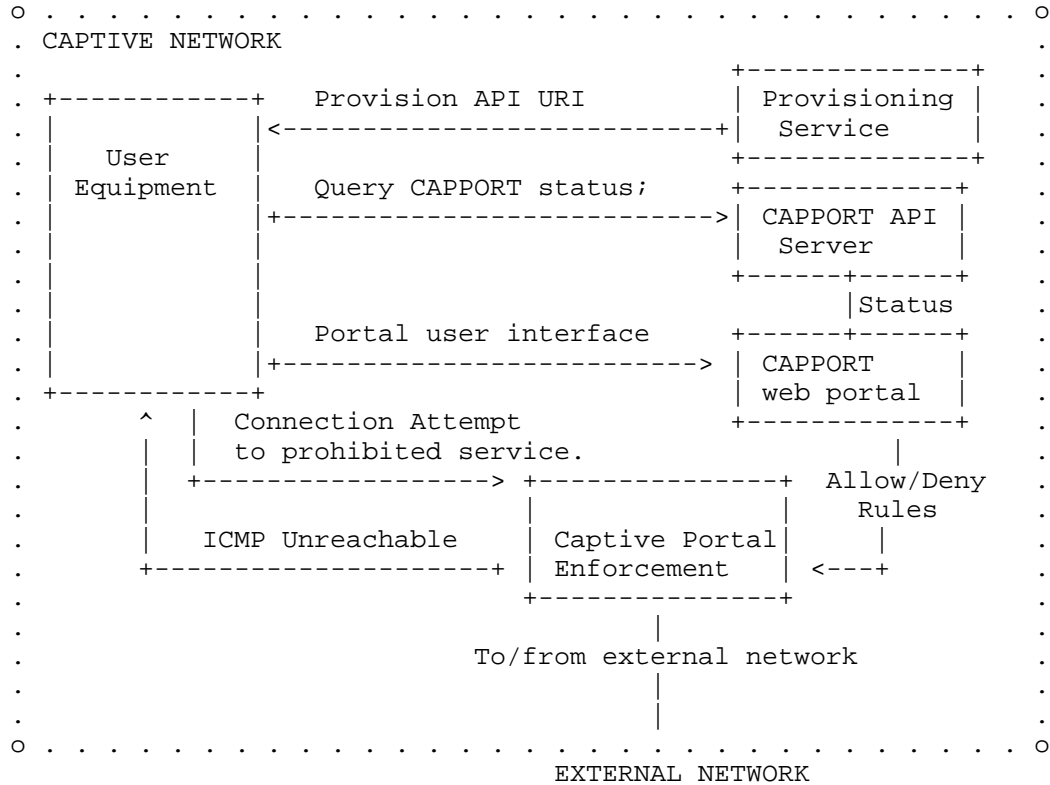


Figure 1: Captive Portal Architecture Component Diagram

In the diagram:

- o During provisioning (e.g., DHCP), the User Equipment acquires the URI for the CAPPORT API.
- o 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.
- o The User Equipment attempts to communicate to the external network through the Captive Portal enforcement device.
- o The Captive Portal Enforcement device either allows the User Equipment's packets to the external network, or responds with an ICMP message.

- o 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. Solution Workflow

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

3.1. 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.

3.2. Conditions Expire

This section describes a possible work-flow when conditions expire and the user visits the portal again (e.g., low quota, or time expiry).

1. Pre-condition: the Captive Portal Enforcement has been configured to detect an expiry condition, which has now occurred.
2. The User Equipment sends a packet to the outside network.
3. The Captive Portal Enforcement detects that the packet is from an expired User Equipment.
4. The Captive Portal Enforcement sends an ICMP message to the User Equipment indicating that it needs to refresh its access. [I-D.wkumari-capport-icmp-unreach].
5. The User Equipment verifies the ICMP message is plausible.
6. The User Equipment queries the Captive Portal API to refresh parameters and status of the Captive Network.
7. If necessary, the User once again navigates the web portal to gain access to the external network.
8. The Captive Portal API Server gives more quota (time, bytes, etc.) to the User Equipment by indicating to the Captive Portal Enforcement the new, extended quota.
9. The User Equipment accesses the external network.

4. Acknowledgements

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

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

6.1. 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.

6.2. Authenticated APIs

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

6.3. Risk of Nuisance Captive Portal

It is possible for any user on the Internet to send ICMP packets in an attempt to cause the receiving equipment to go to the captive portal. This has been considered and addressed in the following ways:

The ICMP packet does not carry the URL, making this method safer than HTTP 3xx-redirect methods currently in use. The User Equipment does not have to use clear-text HTTP to solicit the URL of the portal.

Because the ICMP messages will carry embedded packets sent by the sender, the receiver of the ICMP message can validate that the transport header is plausibly one it sent (i.e., the transport-layer 5-tuple matches an open connection; there is no need to save every packet it sent). This validation requires an off-path attacker to guess the 5-tuple in order to affect a flow. It is trivial for an on-path attacker to send a plausible ICMP packet. (This is not a new ICMP attack.) The impact of getting a valid ICMP packet to the User Equipment is that it will visit the CAPPORT API to check the status. For this reason we recommend the User Equipment rate-limit these requests to the API.

We considered that the ICMP packet could carry a short secret token that would be known to the User Equipment and Captive Portal Enforcement device but would not be available to an attacker, even to an on-path attacker. Although possible to guess by brute force, the impact would be at worst a nuisance visit to the API. We suggest that a 32-bit token would be sufficient to deter nuisance attacks.

Even when redirected, the User Equipment securely authenticates with API servers.

6.4. User Options

The ICMP messaging informs 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.

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <<http://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, <<http://www.rfc-editor.org/info/rfc7710>>.

7.2. Informative References

- [I-D.bruneau-intarea-provisioning-domains] Bruneau, B., Pfister, P., dschinazi@apple.com, d., Pauly, T., and E. Vyncke, "Discovering Provisioning Domain Names and Data", draft-bruneau-intarea-provisioning-domains-01 (work in progress), June 2017.
- [I-D.nottingham-capport-problem] Nottingham, M., "Captive Portals Problem Statement", draft-nottingham-capport-problem-01 (work in progress), April 2016.

[I-D.wkumari-capport-icmp-unreach]

Bird, D. and W. Kumari, "Captive Portal ICMP Messages",
draft-wkumari-capport-icmp-unreach-02 (work in progress),
April 2017.

Authors' Addresses

Kyle Larose
Sandvine
408 Albert Street
Waterloo, ON N2L 3V3
Canada

Phone: +1 519 880 2400
Email: klarose@sandvine.com

David Dolson
Sandvine
408 Albert Street
Waterloo, ON N2L 3V3
Canada

Phone: +1 519 880 2400
Email: ddolson@sandvine.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.

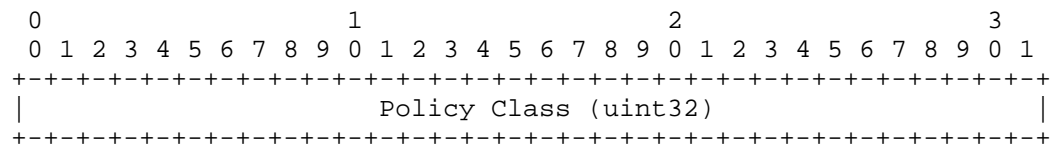
```

      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.



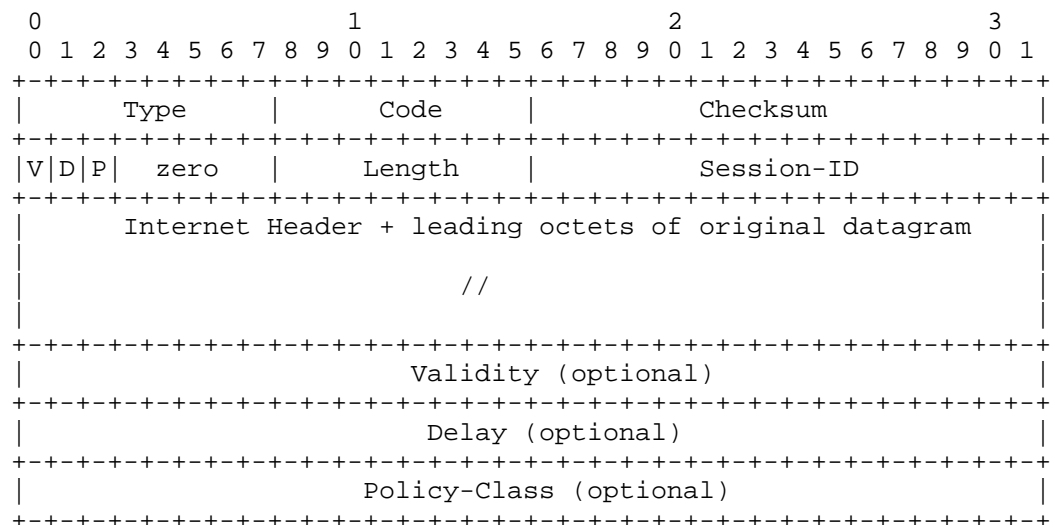
2.6. Message Code/C-Type

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.



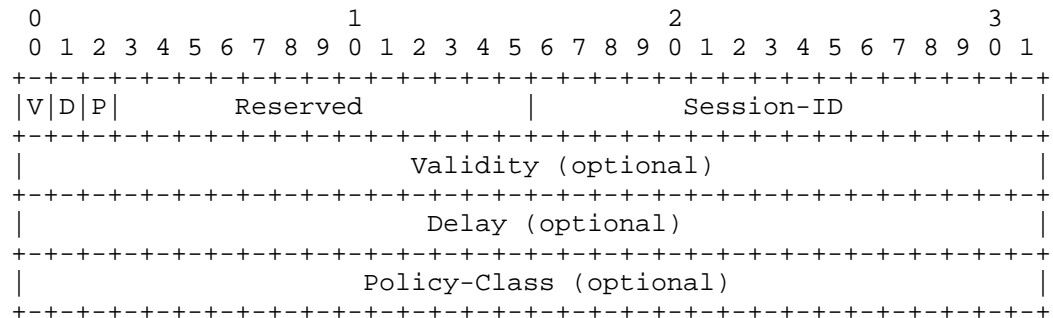
Length Number of 4 byte words of original datagram.

## 2.8. Extension Object

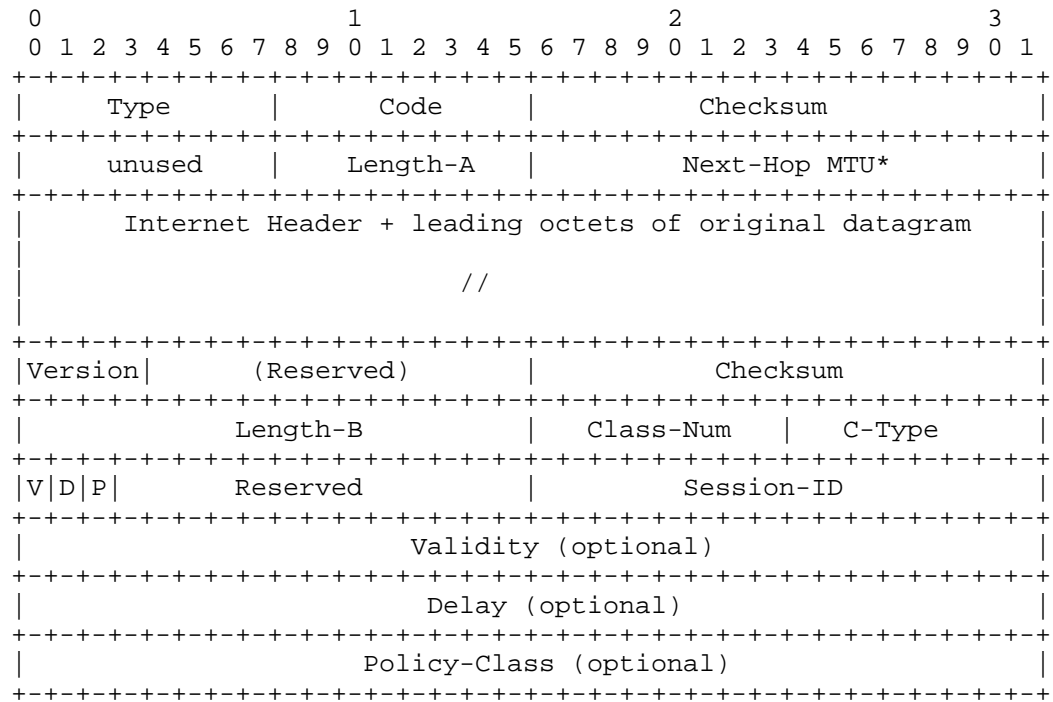
This document defines an extension object that can be appended to selected multi-part ICMP messages ([RFC4884]). This extension permits the CP-NAS to inform Capport ICMP Compliant devices that their connection has been blocked due to an Access Policy requiring interaction with the Captive Portal.

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.

- 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](mailto:dbird@google.com)

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

Email: [warren@kumari.net](mailto:warren@kumari.net)