

MIF  
Internet-Draft  
Intended status: Standards Track  
Expires: August 28, 2016

J. Korhonen  
Broadcom Limited  
S. Krishnan  
Ericsson  
S. Gundavelli  
Cisco Systems  
February 25, 2016

Support for multiple provisioning domains in IPv6 Neighbor Discovery  
Protocol  
draft-ietf-mif-mpvd-ndp-support-03

Abstract

The MIF working group is producing a solution to solve the issues that are associated with nodes that can be attached to multiple networks. One part of the solution requires associating configuration information with provisioning domains. This document details how configuration information provided through IPv6 Neighbor Discovery Protocol can be associated with provisioning domains.

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 August 28, 2016.

Copyright Notice

Copyright (c) 2016 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. Terminology . . . . .	2
3. PVD Container option . . . . .	3
4. Set of allowable options . . . . .	5
5. Security Considerations . . . . .	6
6. IANA Considerations . . . . .	6
7. Acknowledgements . . . . .	6
8. References . . . . .	6
8.1. Normative References . . . . .	6
8.2. Informative References . . . . .	7
Appendix A. Examples . . . . .	7
A.1. One implicit PVD and one explicit PVD . . . . .	8
Authors' Addresses . . . . .	10

## 1. Introduction

The MIF working group is producing a solution to solve the issues that are associated with nodes that can be attached to multiple networks based on the Multiple Provisioning Domains (MPVD) architecture work [RFC7556]. One part of the solution requires associating configuration information with Provisioning Domains (PVD). This document describes an IPv6 Neighbor Discovery Protocol (NDP) [RFC4861] mechanism for explicitly indicating provisioning domain information along with any configuration which is associated with that provisioning domain. The proposed mechanism uses an NDP option that indicates the identity of the provisioning domain and encapsulates the options that contain the configuration information as well as optional authentication/authorization information. The solution defined in this document aligns as much as possible with the existing IPv6 Neighbor Discovery security, namely with Secure Neighbor Discovery (SeND) [RFC3971].

## 2. Terminology

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. PVD Container option

The PVD container option (PVD\_CO) is used to encapsulate the configuration options that belong to the explicitly identified provisioning domain. The PVD container option always encapsulates exactly one PVD identity. The PVD container option MAY occur multiple times in a Router Advertisement (RA) message. In this case each PVD container MUST belong to a different provisioning domain. The PVD container options MUST NOT be nested. The PVD Container option is defined only for the RA NDP message.

Since implementations are required to ignore any unrecognized options [RFC4861], the backward compatibility and the reuse of existing NDP options is implicitly enabled. Implementations that do not recognize the PVD container option will ignore it, and any PVD container option "encapsulated" NDP options without associating them into any provisioning domain (since the implementation has no notion of provisioning domains). For example, the PVD container could "encapsulate" a Prefix Information Option (PIO), which would mark that this certain advertised IPv6 prefix belongs and originates from a specific provisioning domain. However, if the implementation does not understand provisioning domains, then this specific PIO is also skipped and not configured on the interface.

The optional security for the PVD container is based on X.509 certificates [RFC6487] and reuses mechanisms already defined for SeND [RFC3971] [RFC6495]. However, the use of PVD containers does not assume or depend on SeND being deployed or even implemented. The PVD containers SHOULD be signed per PVD certificates, which provides both integrity protection and proves that the configuration information source is authorized for advertising the given information. See [RFC6494] for discussion how to enable deployments where the certificates needed to sign PVD containers belong to different administrative domains i.e., to different provisioning domains.

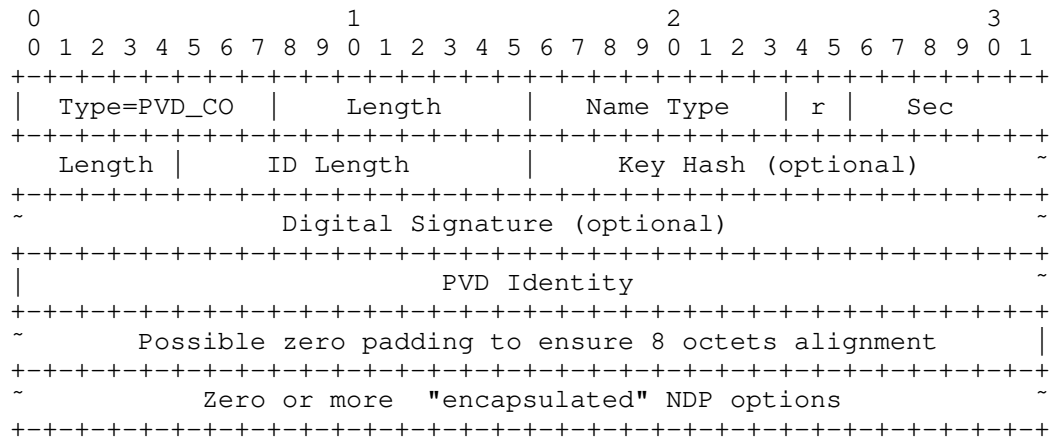


Figure 1: PVD Container Option

**Type**

PVD Container; Set to TBD1.

**Length**

Length of the PVD\_CO. The actual length depends on the number of "encapsulated" NDP options, length of the PVD Identity, and the optional Key Hash/Digital Signature/Padding.

**Name Type**

Names the algorithm used to identify a specific X.509 certificate using the method defined for the Subject Key Identifier (SKI) extension for the X.509 certificates. The usage and the Name Type registry aligns with the mechanism defined for SeND [RFC6495]. Name Type values starting from 3 are supported and an implementation MUST at least support SHA-1 (value 3). Note that if Sec Length=0 the Name field serves no use and MUST be set to 0.

**r**

Reserved. MUST be set to 0 and ignored when received.

**Sec Length**

11-bit length of the Key Hash and Digital Signature in a units of 1 octet. When no security is enabled the Sec Length MUST be set to value of 0.

#### ID Length

11-bit length of the PVD Identity in a units of 1 octet. The ID Length MUST be greater than 0.

#### Key Hash

This field is only present when Sec Length>0. A hash of the public key using the algorithm identified by the Name Type. The procedure how the Key Hash is calculated is defined in [RFC3971] and [RFC6495].

#### Digital Signature

This field is only present when Sec Length>0. A signature calculated over the PVD\_CO option including all option data from the beginning of the option until to the end of the container. The procedure of calculating the signature is identical to the one defined for SeND [RFC3971]. During the signature calculation the contents of the Digital Signature option MUST be treated as all zero.

#### PVD Identity

The provisioning domain identity. The contents of this field is defined in a separate document [I-D.ietf-mif-mpvd-id].

Implementations MUST ensure that the PVD container option meets the 8 octets NDP option alignment requirement as described in [RFC4861].

If the PVD\_CO does not contain a digital signature, then other means to secure the integrity of the NDP message SHOULD be provided, such as utilizing SeND. However, the security provided by SeND is for the entire NDP message and does not allow verifying whether the sender of the NDP message is actually authorized for the information for the provisioning domain.

If the PVD\_CO contains a signature and the verification fails, then the whole PVD\_CO option MUST be silently ignored and the event SHOULD be logged.

#### 4. Set of allowable options

The PVD container option MAY be used to encapsulate any allocated IPv6 NDP options, which may appear more than once in a NDP message. The PVD container option MUST NOT be used to encapsulate other PVD\_CO option(s).

## 5. Security Considerations

An attacker may attempt to modify the information provided inside the PVD container option. These attacks can easily be prevented by using SeND [RFC3971] or per PVD container signature that would detect any form of tampering with the IPv6 NDP message contents.

A compromised router may advertise configuration information related to provisioning domains it is not authorized to advertise. e.g. A coffee shop router may provide configuration information purporting to be from an enterprise and may try to attract enterprise related traffic. The only real way to avoid this is that the provisioning domain container contains embedded authentication and authorization information from the owner of the provisioning domain. Then, this attack can be detected by the client by verifying the authentication and authorization information provided inside the PVD container option after verifying its trust towards the provisioning domain owner (e.g. a certificate with a well-known/common trust anchor).

A compromised configuration source or an on-link attacker may try to capture advertised configuration information and replay it on a different link or at a future point in time. This can be avoided by including some replay protection mechanism such as a timestamp or a nonce inside the PVD container to ensure freshness of the provided information. This specification does not define a replay protection solution. Rather it is assumed that if replay protection is required, the access network and hosts also deploy existing security solutions such as SeND [RFC3971].

## 6. IANA Considerations

This document defines two new IPv6 NDP options into the "IPv6 Neighbor Discovery Option Formats" registry. Option TBD1 is described in Section 3.

## 7. Acknowledgements

The authors would like to thank the members of the MIF architecture design team for their comments that led to the creation of this draft.

## 8. References

### 8.1. Normative References

- [I-D.ietf-mif-mpvd-id]  
Krishnan, S., Korhonen, J., Bhandari, S., and S. Gundavelli, "Identification of provisioning domains", draft-ietf-mif-mpvd-id-02 (work in progress), October 2015.
- [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>>.
- [RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <<http://www.rfc-editor.org/info/rfc3971>>.
- [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, <<http://www.rfc-editor.org/info/rfc4861>>.
- [RFC6494] Gagliano, R., Krishnan, S., and A. Kukec, "Certificate Profile and Certificate Management for SEcure Neighbor Discovery (SEND)", RFC 6494, DOI 10.17487/RFC6494, February 2012, <<http://www.rfc-editor.org/info/rfc6494>>.
- [RFC6495] Gagliano, R., Krishnan, S., and A. Kukec, "Subject Key Identifier (SKI) SEcure Neighbor Discovery (SEND) Name Type Fields", RFC 6495, DOI 10.17487/RFC6495, February 2012, <<http://www.rfc-editor.org/info/rfc6495>>.

## 8.2. Informative References

- [RFC6487] Huston, G., Michaelson, G., and R. Loomans, "A Profile for X.509 PKIX Resource Certificates", RFC 6487, DOI 10.17487/RFC6487, February 2012, <<http://www.rfc-editor.org/info/rfc6487>>.
- [RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <<http://www.rfc-editor.org/info/rfc7556>>.

## Appendix A. Examples

#### A.1. One implicit PVD and one explicit PVD

Figure 2 shows how the NDP options are laid out in an RA for one implicit provisioning domain and one explicit provisioning domain. The example does not include security (and signing of the PVD container). The assumption is the PVD identity consumes total 18 octets (for example encoding a NAI Realm string "dana.example.com").

The explicit provisioning domain contains a specific PIO for 2001:db8:abad:cafe::/64 and the MTU of 1337 octets. The implicit provisioning domain configures a prefix 2001:db8:cafe:babe::/64 and the link MTU of 1500 octets. There are two cases: 1) the host receiving the RA implements provisioning domains and 2) the host does not understand provisioning domains.

1. The host recognizes the PVD\_CO and "starts" a provisioning domain specific configuration. Security is disabled, thus there are no Key Hash or Digital Signature fields to process. The prefix 2001:db8:abad:cafe::/64 is found and configured on the interface. Once the PVD\_ID option is located the interface prefix configuration for 2001:db8:abad:cafe::/64 and the MTU of 1337 octets can be associated to the provisioning domain found in the PVD\_CO option.

The rest of the options are parsed and configured into the implicit provisioning domain since there is no encapsulating provisioning domain. The interface is configured with prefix 2001:db8:cafe:babe::/64. The implicit provisioning domain uses the link MTU of 1500 octets, whereas the "dana.example.com" provisioning domain uses the MTU of 1337 octets (this means when packets are sourced using 2001:db8:abad:cafe::/64 prefix the link MTU is different than when sourcing packets using 2001:db8:cafe:babe::/64 prefix).

2. The host ignores the PVD\_CO and ends up configuring one prefix on its interface ( 2001:db8:cafe:babe::/64) with a link MTU of 1500 octets.



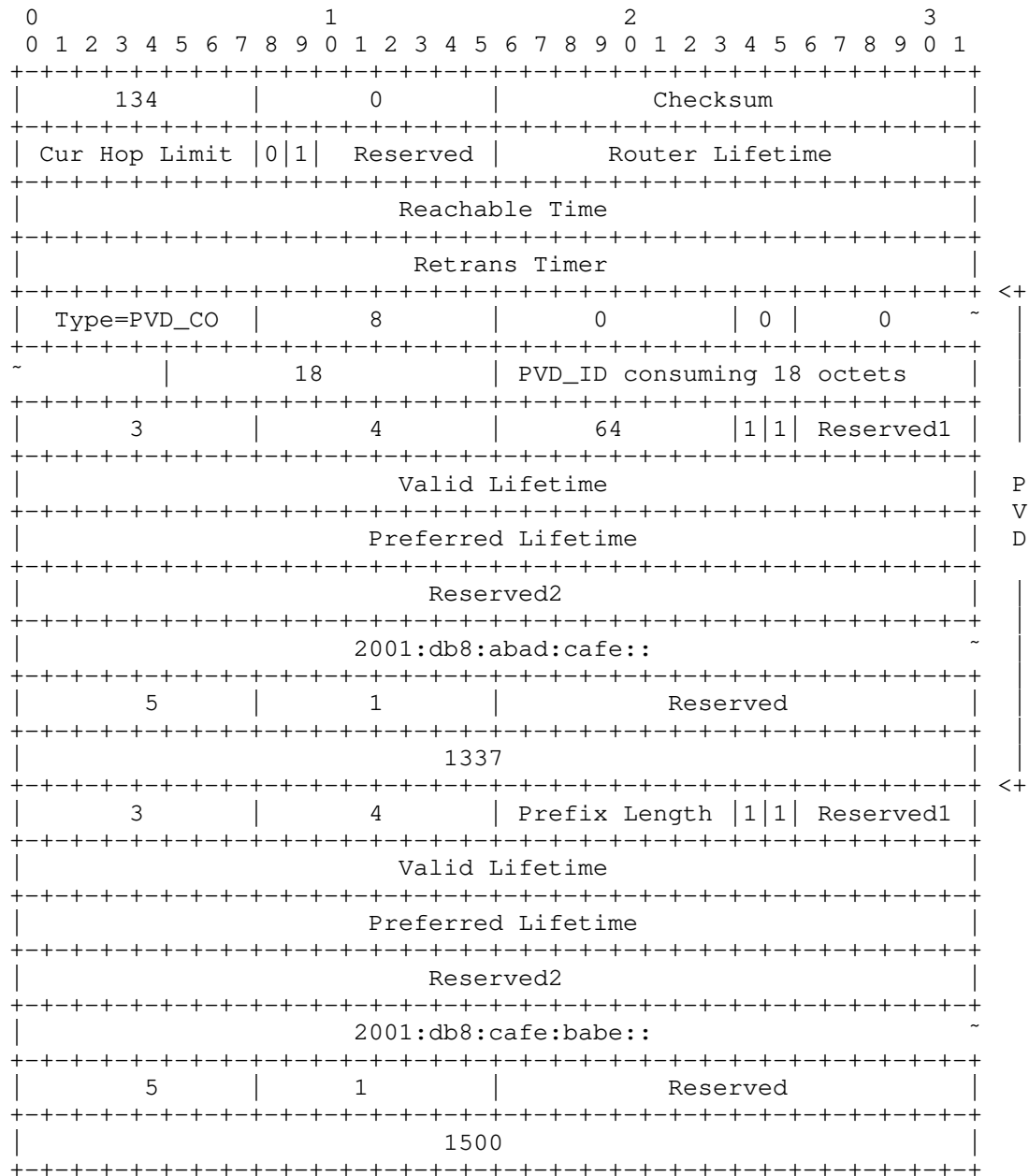


Figure 2: An RA with one implicit PVD and one explicit PVD

Authors' Addresses

Jouni Korhonen  
Broadcom Limited  
3151 Zanker Road  
San Jose, CA 95134  
USA

Email: [jouni.nospam@gmail.com](mailto:jouni.nospam@gmail.com)

Suresh Krishnan  
Ericsson  
8400 Decarie Blvd.  
Town of Mount Royal, QC  
Canada

Phone: +1 514 345 7900 x42871  
Email: [suresh.krishnan@ericsson.com](mailto:suresh.krishnan@ericsson.com)

Sri Gundavelli  
Cisco Systems  
170 West Tasman Drive  
San Jose, CA 95134  
USA

Email: [sgundave@cisco.com](mailto:sgundave@cisco.com)

Internet Engineering Task Force  
Internet-Draft  
Intended status: Informational  
Expires: May 04, 2016

E. Kline  
Google Japan KK  
November 01, 2015

Multiple Provisioning Domains API Requirements  
draft-kline-mif-mpvd-api-reqs-00

Abstract

RFC 7556 [RFC7556] provides the essential conceptual guidance an API designer would need to support use of PvDs. This document aims to capture the requirements for an API that can be used by applications that would be considered "advanced", according to section 6.3 [1] of RFC 7556 [RFC7556]. The "basic" [2] and "intermediate" [3] API support levels can in principle be implemented by means of layers wrapping the advanced API.

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 May 04, 2016.

Copyright Notice

Copyright (c) 2015 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 . . . . .	3
2. High level requirements . . . . .	3
2.1. Requirements for an API . . . . .	3
2.2. Requirements for supporting operating systems . . . . .	5
2.2.1. Source address selection . . . . .	5
2.2.2. Route isolation . . . . .	6
2.2.3. Automatic PvD metadata marking . . . . .	6
2.2.4. Additional system and library support . . . . .	7
3. Conceptual PvDs . . . . .	7
3.1. The 'default' PvD . . . . .	7
3.2. The 'unspecified' PvD . . . . .	8
3.3. The 'null' PvD . . . . .	8
3.4. The 'loopback' PvD . . . . .	8
4. Requirements for new API functionality . . . . .	9
4.1. Learning PvD availability . . . . .	9
4.2. Learning network configuration information comprising a PvD . . . . .	9
4.3. Scoping functionality to a specific PvD . . . . .	10
4.4. Explicit versus Implicit PvDs . . . . .	10
4.5. Policy restrictions . . . . .	11
4.6. Programmatic reference implementation considerations . . . . .	11
5. Existing networking APIs . . . . .	12
5.1. Updating existing APIs . . . . .	12
5.2. Requirements for name resolution APIs . . . . .	12
6. Acknowledgements . . . . .	13
7. IANA Considerations . . . . .	13
8. Security Considerations . . . . .	13
9. References . . . . .	13
9.1. Normative References . . . . .	13
9.2. Informative References . . . . .	14
Author's Address . . . . .	14

## 1. Introduction

RFC 7556 [RFC7556] provides the essential conceptual guidance an API designer would need to support use of PvDs. This document aims to capture the requirements for an API that can be used by applications that would be considered "advanced", according to section 6.3 [4] of RFC 7556 [RFC7556]. The "basic" [5] and "intermediate" [6] API support levels can in principle be implemented by means of layers wrapping the advanced API.

This document also attempts to make some of the API implementation requirements more concrete by discussion and example.

### 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].

## 2. High level requirements

As described in section 2 [7] of RFC 7556 [RFC7556], a Provisioning Domain ("PvD") is fundamentally a "consistent set of network configuration information." This includes information like:

- o the list of participating interfaces
- o IPv4 and IPv6 addresses
- o IPv4 and IPv6 routes: both default routes and more specifics (such as may be learned via RFC 4191 [RFC4191] Route Information Options ("RIOs"))
- o DNS nameservers, search path, et cetera
- o HTTP proxy configuration

and undoubtedly many more configuration elements yet to be specified (like metering hints, transmission medium and speed, captive portal URL, et cetera).

This configuration information as a whole may not be able to be learned atomically, may need to be synthesized from multiple sources including administrative provisioning, and cannot be presumed to be unchanging over the lifetime of a node's association with a given PvD.

In order for an application to make consistent use [8] of a given PvD's network configuration several requirements are placed upon the API itself and the host operating system providing the API.

### 2.1. Requirements for an API

At the highest level, the requirements for an API that enables applications to make sophisticated use of multiple PvDs amount to providing mechanisms by which they can:

- R1 observe accessible PvDs

It MUST be possible for an application to be informed of the set of all PvDs it can currently access, and to be informed of changes to this set.

- R2 observe configuration elements of an accessible PvD

It MUST be possible to learn requested configuration information of any accessible PvD, and to be informed of any changes to the configuration information comprising an accessible PvD.

- R3 scope networking functionality to a specified PvD

For every existing API function that interacts with the node's networking stack, be it at a relatively high level like `getaddrinfo()` [9] or at the level of something like Sockets API's `sendmsg()`, there MUST be a means by which an application can specify the PvD within which networking operations are to be restricted.

- R4 use one and only specified scope per networking functionality invocation

For every unique invocation of a networking API function, there MUST only be one specified PvD to which networking functionality is to be restricted. At any given point in an application's lifetime there MAY be several encapsulating layers of unspecified PvDs (Section 3.2) through which the implementation must progressively search to find a specified PvD, but ultimately a networking function MUST use one and only one PvD for its operations, even if that PvD is a "null PvD" (Section 3.3).

- R5 make consistent use of programmatic references to PvDs

For uniformity and simplicity, every PvD-aware API functional element SHOULD use (as return values of function calls, function arguments, et cetera) the same programmatic reference for PvDs, e.g. a construct containing a PvD identifier [10] or some equivalent shorthand reference token (see Section 4.6 for a discussion of implementation considerations). Regardless of the implementation strategy chosen, a given programmatic reference MUST remain constant over the lifetime of the node's continuous attachment to the PvD to which it refers (until a disconnection or disassociation event occurs). Additionally, references MAY change with successive re-associations to the same PvD whereas PvD identifiers, by definition, will not.

It is important to note that there is always a provisioning domain within which networking functionality is scoped. For simply-

connected hosts this may be the implicit PvD [11] created by a single networking interface connected to a traditional, shared LAN segment. For multihomed hosts the "default provisioning domain" is likely a matter of policy, but MAY be a "null" PvD, i.e. one completely devoid of networking configuration information (no addresses, no routes, et cetera). See Section 3 for further discussion.

The utility of such an API (allowing applications to learn of and control the scope of networking functionality) suggests that the Provisioning Domain is perhaps a more useful operational definition for the original IPv6 concept of a "site-local scope" than the ill-fated [RFC3879], "ill-defined concept" [12] of a site. It also suggests one possible way by which operating system support for a PvD-aware API might be implemented.

## 2.2. Requirements for supporting operating systems

The multiple PvD model of host behaviour is perhaps closer to the Strong End System Model than the Weak End System Model characterized in RFC 1122 [RFC1122] section 3.3.4.2 [13], but owing to its recognition of a many-to-many relationship between interfaces and PvDs should be considered a unique model unto itself.

In the PvD-aware End System Model, the "two key requirement issues related to multihoming" are restated as:

- a. A host MAY silently discard an incoming datagram whose destination address does not correspond to any PvD associated with the physical (or virtual) interface through which it is received.
- b. A host MUST restrict itself to sending (non-source-routed) IP datagrams only through the physical (or virtual) interfaces that correspond to the PvD associated with the IP source address of the datagrams.

In order to support a PvD-aware application's use of multiple PVDs, several additional requirements must be met by the host operating system, especially when performing functions on behalf of applications or when no direct application intervention is possible, as discussed in the following sections.

### 2.2.1. Source address selection

Whenever a source address is to be selected on behalf of an application it is essential for consistent use that only source addresses belonging to the specified PvD be used a candidate set. (See RFC 6418 [RFC6418] section 3.5 [14] for references to issues arising from poor source address selection.)

For nodes following the PvD-aware End System Model, RFC 6724 [RFC6724] section 4 [15] is amended as follows:

- R6 The candidate source addresses MUST be restricted to the set of unicast addresses associated with the concurrently specified PvD.

Additionally, source address selection policies from PvDs other than the concurrently specified PvD MUST NOT be applied.

#### 2.2.2. Route isolation

Whenever a routing lookup for a given destination is to be performed, it is essential that only routes belonging to the currently specified PvD be consulted. Applications and libraries that use the inherent routing reachability check (and subsequent source address selection) performed during something like the Sockets API connect() call on a UDP socket to learn reachability information cheaply cannot function correctly otherwise. RFC 6418 [RFC6418] section 4.2 [16] contains more discussion and references to issues arising from insufficiently isolated routing information.

For nodes following the PvD-aware End System Model:

- R7 The set of routes consulted for any routing decision MUST be restricted to the routes associated with the concurrently specified PvD.

#### 2.2.3. Automatic PvD metadata marking

In many cases, an application can examine a source address or the destination address of a received datagram and use that address's association with a given PvD to learn, for example, the PvD with which an incoming connection may be associated. It may, however, be impossible for an application to make this determination on its own if, for example, an incoming TCP connection is destined to a RFC 1918 [RFC1918] address that happens to be configured in multiple PvDs at the same time. In such circumstances, the supporting operating system will need to provide additional assistance.

For nodes following the PvD-aware End System Model:



- R8 When performing networking functionality on behalf of an application, the supporting operating system MUST record and make available to the application either (1) all the information the application might need to make a determination of the applicable PvD on its own or (2) the API's PvD programmatic reference directly.

A supporting operating system SHOULD record and make available the API's PvD programmatic reference; other approaches invite ambiguity among applications' interpretation of available information.

#### 2.2.4. Additional system and library support

Frequently, operating systems have several additional supporting libraries and services for more advance networking functionality. Using the system's own PvD API, and fulfilling the above requirements, it should be possible to extend these services to provide correct per-PvD isolation of information and enable consistent application use of PvDs.

### 3. Conceptual PvDs

#### 3.1. The 'default' PvD

Because there is always one specified provisioning domain to which an individual invocation of networking functionality is restricted (Section 2.1) there must necessarily exist a system "default PvD". This provisioning domain is the one which networking functionality MUST use when no other specified PvD can be determined.

Using the system's default PvD enables support of basic [17] uses of the PvD API (i.e. backward compatibility for unmodified applications).

The operating system MAY change the default PvD accordingly to policy. It is expected that nodes will use a variety of information, coupled with administrative policy, to promote one of any number of concurrently available PvDs to be the system's default PvD.

- R9 A PvD-aware API implementation MUST include a mechanism for applications to learn the programmatic reference to the system's concurrent default PvD.

- R10 A PvD-aware API implementation SHOULD contain a mechanism enabling an application to be notified of changes to the concurrent default PvD in a comparatively efficient manner (i.e. more efficient than polling).

### 3.2. The 'unspecified' PvD

An application may at some times wish to be specific about which PvD should be used for networking operations and at other times may prefer to defer the choice of specific PvD to one specified elsewhere (including the system default PvD).

For example, if an application has specified the PvD to be used for all functions called by its process and child processes (Section 4.3), it may indicate that certain invocations should instead use the system default PvD by using a programmatic reference to the "unspecified PvD".

R11 API implementors MUST reserve a programmatic reference to represent an "unspecified PvD": an indication that the application defers the selection of a specific PvD.

R12 When invoked without a specific PvD, or with a programmatic reference to the "unspecified PvD", networking functionality MUST find a specific PvD to be used by examining the successive encapsulating layers of possible specificity supported by the API (Section 4.3), e.g. look first for a "fiber-specific default" PvD, then a "thread-specific default" PvD, a "process-specific default" PvD, and ultimately use the system's default PvD if no other specified PvD can be found.

### 3.3. The 'null' PvD

If there are no PvDs accessible to an application, whether as a matter of policy (insufficient privileges) (Section 4.5) or as a matter of natural circumstance (the node is not connected to any network), the construct of a 'null' PvD may be useful to ensure networking functions fail (and fail quickly).

R13 API implementors MAY reserve a programmatic reference to represent a "null PvD": an unchanging provisioning domain devoid of any and all networking configuration information.

It is possible for operating systems to enforce that only PvD-aware applications may function normally by administratively configuring the default PvD to be the "null PvD".

### 3.4. The 'loopback' PvD

TBD: is it useful to have a "loopback" PvD, i.e. one consisting solely of all addresses configured on the node and all locally delivered routes?

#### 4. Requirements for new API functionality

##### 4.1. Learning PvD availability

R14 A PvD-aware API MUST implement a mechanism whereby an application can receive a set of the API's PvD programmatic references representing the complete set of PvDs (both explicit [18] and implicit [19]) with which the node is currently associated.

R15 A PvD-aware API implementation SHOULD contain a mechanism enabling an application to be notified of changes in the above set of actively associated PvDs in a comparatively efficient manner (i.e. more efficient than polling).

It may also be of use to applications to receive notifications of pending changes to the set of currently connected PvDs. For example, if it is known that a connection to a PvD is scheduled to be terminated shortly, an application may be able to take some appropriate action (migrate connections to another PvD, send notifications, et cetera).

##### 4.2. Learning network configuration information comprising a PvD

R16 A PvD-aware API MUST include a mechanism whereby by an application, using the API's PvD programmatic reference, can receive elements of the network configuration information that comprise a PvD. At a minimum, this mechanism MUST be capable of answering queries for:

- \* the PvD identifier
- \* all participating interfaces
- \* all IPv4 and all non-deprecated IPv6 addresses
- \* all configured DNS nameservers

A PvD's network configuration information is neither guaranteed to be learned atomically nor is it guaranteed to be static. Addresses, routes, and even DNS nameservers and participating interfaces may each change over the lifetime of the node's association to a given PvD. Timely notification of such changes may be of particular importance to some applications.

- R17 A PvD-aware API implementation SHOULD contain a mechanism enabling an application to be notified of changes in the networking configuration information comprising a PvD in a comparatively efficient manner (i.e. more efficient than polling).
- R18 A network configuration query API implementation SHOULD take extensibility into account, to support querying for configuration information not yet conceived of with minimal adverse impact to applications.

#### 4.3. Scoping functionality to a specific PvD

- R19 A PvD-aware API implementation MUST include a mechanism for an application to specify the programmatic reference of the PvD to which all networking functionality MUST be restricted when not otherwise explicitly specified (a configurable, application-specific "default PvD").
- R20 The API implementation MUST support setting such a "default PvD" for an application's entire process (and by extension its child processes). Additionally, the API SHOULD support an application setting a "default PvD" at every granularity of "programming parallelization", i.e. not only per-process, but also per-thread, per-fiber, et cetera. At every supported layer of granularity, if no PvD reference has been set the next coarser layer's setting MUST be consulted (up to and including the system's default PvD) when identifying the specified PvD to be used.
- R21 For every degree of granularity at which an application may specify a "default PvD" there MUST exist a corresponding mechanism to retrieve any concurrently specified implementation-specific PvD programmatic reference. If no PvD has been specified for at the granularity of a given query, the "unspecified PvD" must be returned.

With access to this functionality it is possible to start non-PvD-aware applications within a single PvD context with no adverse impact. Furthermore, with judicious use of a sufficiently granular API, existing general purpose networking APIs can be wrapped to appear PvD-aware.

#### 4.4. Explicit versus Implicit PvDs

R22 Because programmatic references to PvDs are returned for both explicit and implicit PvDs, the MPvD API implementation MUST be equally applicable and useful for any valid type of PvD; it MUST NOT be necessary for a PvD-aware application to distinguish between explicit and implicit PvDs to function properly.

#### 4.5. Policy restrictions

This document does not make recommendations about policies governing the use of any or all elements of a PvD API, save only to note that some restrictions on use may be deemed necessary or appropriate.

R23 A PvD API implementation MAY implement policy controls whereby access to PvD availability information, configuration elements, and/or explicit scoping requests is variously permitted or denied to certain applications.

#### 4.6. Programmatic reference implementation considerations

PvD identifiers may be of a length or form not easily handled directly in some programming environments, and unauthenticated PvD identifiers are assumed to be only probabilistically unique [20]. As such, API implementations should consider using some alternative programmatic reference (a node-specific "handle" or "token"), which is fully under the control of the operating system, to identify an instance of a single provisioning domain's network configuration information.

Even though a PvD identifier may uniquely correspond to, say, a network operator, there is no guarantee that the configuration information (delegated prefixes, configured IP addresses, and so on) will be the same with every successive association to the same PvD identifier. An implementation may elect to change the value of the programmatic reference to a given PvD identifier for each temporally distinct association. Doing so presents some advantages worth considering:

Collisions in the PvD identifier space will inherently be treated as distinct by applications not concerned solely with identifiers.

Changing the value of a reference can disabuse application writers of inappropriately caching configuration information from one association instance to another.

Whether two PvDs are "identical" is perhaps better left to applications to decide since "PvD equivalence" for a given application may alternatively be determined by successfully accessing some restricted resource.

This document makes no specific requirement on the type of programmatic reference used by the API.

## 5. Existing networking APIs

### 5.1. Updating existing APIs

From the perspective of a PvD-aware operating system, all previously existing non-PvD-enabled networking functionality had historically been executed within the context of a single, implicit provisioning domain. A sufficiently granular API to specify which PvD is to be used to scope subsequent networking functionality (Section 4.3) can be used to wrap non-PvD-aware APIs, giving them this new PvD-aware capability. However,

R24 Operating system implementors SHOULD consider updating existing networking APIs to take or return programmatic references to PvDs directly.

This may mean creating new functions with an additional PvD programmatic reference argument, adding a PvD programmatic reference field to an existing structure or class that is itself an argument or return type, or finding other means by which to use a programmatic reference with minimal or no disruption to existing applications or libraries.

### 5.2. Requirements for name resolution APIs

RFC 3493 [RFC3493] `getaddrinfo()` [21] and `getnameinfo()` [22] APIs deserve explicit discussion. Previously stated requirements make it clear that it MUST be possible for an application to perform normal name resolution constrained to the DNS configuration within a specified PVD. This MUST be possible using at least the techniques of Section 4.3.

The following additional requirements are places on PvD-aware implementations of these functions:

R25 All DNS protocol communications with a PvD's nameservers MUST be restricted to use only source addresses and routes associated with the PvD.

R26 If `getaddrinfo()` is called with the `AI_ADDRCONFIG` flag specified, IPv4 addresses shall be returned only if an IPv4 address is configured within the specified provisioning domain and IPv6 addresses shall be returned only if an IPv6 address is configured within the specified provision domain. The loopback address is (still) not considered for this case as valid as a configured address.

## 6. Acknowledgements

The core concepts presented in this document were developed during the Android multinetworking effort by Lorenzo Colitti, Robert Greenwalt, Paul Jensen, and Sreeram Ramachandran.

Additional thanks to the coffee shops of Tokyo.

## 7. IANA Considerations

This memo includes no request to IANA.

## 8. Security Considerations

An important new security impact of a PvD-aware API is that it becomes much simpler (by design) to write a well-functioning application to create a bridging data path between two PvDs that would not otherwise have been so easily connected.

For some operating systems, existing APIs already make this bridging possible, though some functionality like DNS resolution may have been difficult to implement. Indeed, the very aim of an MPvD API is to make implementing a PvD-aware application simple and to make its functioning more "correct" ("first class" support for such functionality).

Operating system implementations have several points of potential policy control including:

- o use of certain PvDs MAY be restricted by policy (e.g. only approved users, groups, or applications might be permitted access), and/or
- o use of more than one PvD (or the MPvD API itself) MAY be similarly restricted.

## 9. References

### 9.1. Normative References

- [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, March 1997.
- [RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, DOI 10.17487/RFC3493, February 2003, <<http://www.rfc-editor.org/info/rfc3493>>.
- [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, <<http://www.rfc-editor.org/info/rfc6724>>.
- [RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <<http://www.rfc-editor.org/info/rfc7556>>.

## 9.2. Informative References

- [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, February 1996.
- [RFC3879] Huitema, C. and B. Carpenter, "Deprecating Site Local Addresses", RFC 3879, DOI 10.17487/RFC3879, September 2004, <<http://www.rfc-editor.org/info/rfc3879>>.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <<http://www.rfc-editor.org/info/rfc4191>>.
- [RFC6418] Blanchet, M. and P. Seite, "Multiple Interfaces and Provisioning Domains Problem Statement", RFC 6418, DOI 10.17487/RFC6418, November 2011, <<http://www.rfc-editor.org/info/rfc6418>>.

Author's Address



Erik Kline  
Google Japan KK  
6-10-1 Roppongi  
Mori Tower, 44th floor  
Minato, Tokyo 106-6126  
JP

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

IETF  
Internet-Draft  
Intended status: Informational  
Expires: October 13, 2016

P. McCann, Ed.  
J. Kaippallimalil, Ed.  
Huawei  
April 11, 2016

Communicating Prefix Cost to Mobile Nodes  
draft-mccann-dmm-prefixcost-03

Abstract

In a network implementing Distributed Mobility Management, it has been agreed that Mobile Nodes (MNs) should exhibit agility in their use of IP addresses. For example, an MN might use an old address for ongoing socket connections but use a new, locally assigned address for new socket connections. Determining when to assign a new address, and when to release old addresses, is currently an open problem. Making an optimal decision about address assignment and release must involve a tradeoff in the amount of signaling used to allocate the new addresses, the amount of utility that applications are deriving from the use of a previously assigned address, and the cost of maintaining an address that was assigned at a previous point of attachment. As the MN moves farther and farther from the initial point where an address was assigned, more and more resources are used to redirect packets destined for that IP address to its current location. The MN currently does not know the amount of resources used as this depends on mobility path and internal routing topology of the network(s) which are known only to the network operator. This document provides a mechanism to communicate to the MN the cost of maintaining a given prefix at the MN's current point of attachment so that the MN can make better decisions about when to release old addresses and assign new ones.

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 13, 2016.

#### Copyright Notice

Copyright (c) 2016 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. Abbreviations . . . . .	4
2. Motivation . . . . .	4
3. Prefix Cost Sub-option . . . . .	5
4. Host Considerations . . . . .	6
5. Security Considerations . . . . .	7
6. IANA Considerations . . . . .	8
7. References . . . . .	8
7.1. Normative References . . . . .	8
7.2. Informative References . . . . .	8
Authors' Addresses . . . . .	9

#### 1. Introduction

Previous discussions on address agility in distributed mobility management have focused on "coloring" prefixes with one of a small number of categories, such as Fixed, Sustained, or Nomadic. The assumption here is that the MN should use a permanent home address for sessions that need a persistent IP address, and a local, ephemeral address for short-lived sessions such as browsing. However, a small set of address categories lacks expressive power and leads to false promises being made to mobile nodes. For example, the concept that a home address can be maintained permanently and offered as an on-link prefix by any access router to which the MN may be attached in future is simply not attainable in the real world. There will always exist some access routers that do not have arrangements in place with the home network to re-route (via tunneling or other mechanisms) the home prefix to the current point of attachment.

Conversely, the assumption that a Nomadic prefix will never be available to an MN after it changes its current point of attachment is too limiting. There is no reason why an MN should not be able to keep a prefix that was assigned by a first network after it moves to a second network, provided that measures are put in place to re-route such prefixes to the new attachment point.

Rather, this document argues that there is in reality a continuum of cost associated with an address as the MN moves from one attachment point to another or from one network to another. The sources of the cost are the increased latency, network bandwidth, and network state being maintained by a network-based mobility management scheme to route packets destined to the prefix to the MN's current point of attachment. By communicating this cost to the MN every time its attachment point changes, the MN can make intelligent decisions about when to release old addresses and when to acquire new ones.

The cost should be communicated to the MN because of several constraints inherent in the problem:

- (1) The MN is the entity that must make decisions about allocating new addresses and releasing old ones. This is because only the MN has the information about which addresses are still in use by applications or have been registered with other entities such as DNS servers.
- (2) Only the network has information about the cost of maintaining the prefix in a network-based mobility management scheme, because the MN cannot know the network topology that gives rise to the inefficiencies.

If the cost of maintaining a prefix is not made available to the mobile node, it may attempt to infer the cost through heuristic mechanisms. For example, it can measure increased end-to-end latency after a mobility event, and attribute the increased latency to a longer end-to-end path. However, this method does not inform the MN about the network bandwidth being expended or network state being maintained on its behalf. Alternatively, a MN may attempt to count mobility events or run a timer in an attempt to guess at which older prefixes are more costly and in need of being released. However, these methods fail because the number of mobility events is not an indication of how far the MN has moved in a topological sense from its original attachment point which is what gives rise to the costs outlined above. Re-allocating an address upon expiration of a timer may introduce unnecessary and burdensome signaling load on the network and air interface.

### 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 [1].

### 1.2. Abbreviations

ANDSF	Access Network Discovery and Selection Function
MN	Mobile Node
MPTCP	Multi-Path Transmission Control Protocol
ND	Neighbor Discovery
NGMN	Next Generation Mobile Networks
NUD	Neighbor Unreachability Detection
OMA-DM	Open Mobile Alliance - Device Management
PIO	Prefix Information Discovery
PGW	Packet data network Gateway
SeND	Secure Neighbor Discovery
SGW	Serving Gateway

## 2. Motivation

The Introduction speaks in general terms about the cost of a prefix. More specifically, we are talking about the aggregate amount of state being maintained in the network on behalf of the mobile node in addition to the transport resources being used (or wasted) to get packets to the MN's current point of attachment.

In a non-mobile network, the addresses can be assigned statically in a manner that is aligned with the topology of the network. This means that prefix aggregation can be used for maximum efficiency in the state being maintained in such a network. Nodes deep in the network need only concern themselves with a small number of short prefixes, and only nodes near the end host need to know longer more specific prefixes. In the best case, only the last-hop router(s) need to know the actual address assigned to the end host. Also, routing protocols ensure that packets follow the least-cost path to the end host in terms of number of routing hops or according to other policies defined by the service provider, and these routing paths can change dynamically as links fail or come back into service.

However, mobile nodes in a wide-area wireless network are often handled very differently. A mobile node is usually assigned a fixed gateway somewhere in the network, either in a fixed central location or (better) in a location near where the MN first attaches to the network. For example, in a 3GPP network this gateway is a PGW that can be allocated in the home or visited networks. Initially, the cost of such a prefix is the state entry in the fixed gateway plus

any state entries in intermediate tunneling nodes (like SGWs) plus whatever transport resources are being used to get the packet to the MN's initial point of attachment.

When an MN changes its point of attachment, but keeps a fixed address, the cost of the prefix changes (usually it increases). Even if the fixed gateway was initially allocated very close to the initial point of attachment, as the MN moves away from this point, additional state must be inserted into the network and additional transport resources must be provided to get the packets to the current point of attachment. For example, a new SGW might be allocated in a new network, and now the packets must traverse the network to which the MN first attached before being forwarded to their destination, even though there may be a better and more direct route to communication peers from the new network. Whatever aggregation was possible at the initial point of attachment is now lost and tunnels must be constructed or holes must be punched in routing tables to ensure continued connectivity of the fixed IP address at the new point of attachment. Over time, as the MN moves farther and farther from its initial point of attachment, these costs can become large. When summed over millions of mobile nodes, the costs can be quite large.

Obviously, the assignment of a new address at a current point of attachment and release of the older, more costly prefix will help to reduce costs and may be the only way to meet emerging more stringent latency requirements [8]. However, the MN does not in general know the current cost of a prefix because it depends on the network topology and the number of handovers that have taken place and whether these handovers have caused the MN to transition between different topological parts of the network. It is the purpose of the protocol extension defined in this document to communicate the current cost of a prefix to the MN so that it can make intelligent decisions about when to get a new address and when to release older addresses. Only the MN can make a decision about when to release an address, because it is the only entity that knows whether applications are still listening waiting to receive packets at the old address.

Section 4 describes MN behavior when Router Advertisements with Prefix Cost is received.

### 3. Prefix Cost Sub-option

This document defines a prefix cost option to be carried in router advertisements. It is a sub-option that carries meta-data as defined by Korhonen et al. [7]

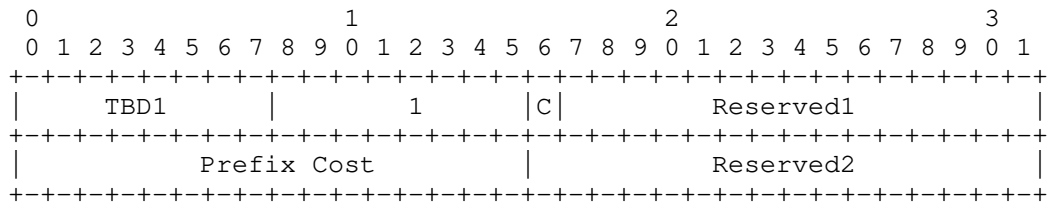


Figure 1: Prefix Cost suboption

The prefix cost is carried as a 16-bit, unsigned number in network byte order. An higher number indicates an increased cost.

This sub-option is appended in Router Advertisement messages that are sent on a periodic basis. No additional signaling cost is incurred to support this mechanism.

It should be noted that link layer events do not cause a change in the prefix cost.

The prefix cost is for a connection segment. No end-to-end congestion or flow control mechanisms are implied with this cost.

#### 4. Host Considerations

Prefix Cost in a Router Advertisement PIO serves as a hint for the MN to use along with application knowledge, MN policy configuration on network cost and available alternative routes to determine the IP addresses and routes used. For example, if the application is downloading a large file, it may want to maintain an IP address and route until the download is complete. On the other hand, some applications may use multiple connections (e.g., with MPTCP) and may not want to maintain an IP address above a configured cost. It could also be the case that the MN maintains the IP address even at high cost if there is no alternative route/address. These decisions are made based on configured policy, and interaction with applications, all of which are decided by the MN.

When the MN is ready to release an IP address, it may send a DHCPv6 [5] Release message. The network may also monitor the status of a high cost connection with Neighbor Unreachability Detection (NUD) [2], [6], and determine that an address is not used after the NUD times out. The network should not continue to advertise this high cost route following the explicit release of the address or NUD timeout. It can initiate the release of network resources dedicated to providing the IP address to the MN.

The operator of the network or host's service provider can configure policy that determines how the host should handle the prefix cost values. In a 3GPP network, the subscription provider may configure policies in the host via OMA-DM or S14 (ANDSF). For example, the service provider may configure rules to state that prefix cost values below 500 indicate low cost and ideal access network conditions, values from 501 - 5000 indicate that the host should try to relocate connections, and values above 5000 indicate a risk and impending loss of connectivity. The policies themselves can be (re-)configured as needed by the operator. Prefix cost information with each Router Advertisement allows the host to interpret a simple number and associated policies to (re-)select optimal routes. For networks service providers, when this cost is associated with charging, it can be a valuable tool in dynamically managing the utilization of network resources.

This draft does not aim to provide definitive guidance on how an OS or application process receives indications as a result of prefix cost option being conveyed in Router Advertisements. Only high level design options are listed here. New socket options or other APIs can be used to communicate the cost of an address in use on a given connection. For example, a new "prefix-cost" socket option, if set, can indicate that the application is interested in being notified when there is a change in the prefix cost. The actual mechanisms used to either notify or other means of busy polling on this change of prefix cost information need to be specified in other drafts. An alternative to the application discovering the changed prefix cost is to use a model where a connection manager handles the interface between the network and the application (e.g., Android Telephony Manager [9]). In this case, the connection manager is responsible to select and manage addresses based on policies (configured via OMA-DM or S14) and prefix cost obtained from the Router Advertisements.

## 5. Security Considerations

Security of the prefix cost option in the PIO needs to be considered. Neighbor Discovery (ND) and Prefix Information Option (PIO) security are described in [2] and [3]. A malicious node on a shared link can advertise a low cost route in the prefix cost option and cause the MN to switch. Alternatively, an incorrect higher cost route in the prefix cost option can result in the suboptimal use of network resources. In order to avoid such on-link attacks, SeND [4] can be used to reject Router Advertisements from nodes whose identities are not validated.



## 6. IANA Considerations

This memo defines a new Prefix Information Option (PIO) sub-option in Section 3.

## 7. References

### 7.1. Normative References

- [1] 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>>.
- [2] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<http://www.rfc-editor.org/info/rfc4861>>.
- [3] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <<http://www.rfc-editor.org/info/rfc4191>>.
- [4] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <<http://www.rfc-editor.org/info/rfc3971>>.
- [5] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <<http://www.rfc-editor.org/info/rfc3315>>.
- [6] Nordmark, E. and I. Gashinsky, "Neighbor Unreachability Detection Is Too Impatient", RFC 7048, DOI 10.17487/RFC7048, January 2014, <<http://www.rfc-editor.org/info/rfc7048>>.

### 7.2. Informative References

- [7] Korhonen, J., Gundavelli, S., Seite, P., and D. Liu, "IPv6 Prefix Properties", draft-korhonen-dmm-prefix-properties-05 (work in progress), February 2016.
- [8] NGMN Alliance, "NGMN 5G Whitepaper", February 2015.

- [9]           Android Telephony Developer's Forum,  
              <http://developer.android.com/reference/android/telephony/TelephonyManager.html>, "Android Telephony Manager".

Authors' Addresses

Peter J. McCann (editor)  
Huawei  
400 Crossing Blvd, 2nd Floor  
Bridgewater, NJ 08807  
USA

Phone: +1 908 541 3563  
Email: [peter.mccann@huawei.com](mailto:peter.mccann@huawei.com)

John Kaippallimalil (editor)  
Huawei  
5340 Legacy Dr., Suite 175  
Plano, TX 75024  
USA

Email: [john.kaippallimalil@huawei.com](mailto:john.kaippallimalil@huawei.com)