

SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2019

Z. Ali
C. Filsfils
N. Kumar
C. Pignataro
F. Iqbal
R. Gandhi
Cisco Systems, Inc.
J. Leddy
Comcast
S. Matsushima
SoftBank
R. Raszuk
Bloomberg LP
D. Voyer
Bell Canada
G. Dawra
LinkedIn
B. Peirens
Proximus
M. Chen
Huawei
G. Naik
Drexel University
October 22, 2018

Operations, Administration, and Maintenance (OAM) in Segment
Routing Networks with IPv6 Data plane (SRv6)
draft-ali-spring-srv6-oam-02.txt

Abstract

This document defines building blocks that can be used for Operations, Administration, and Maintenance (OAM) in Segment Routing Networks with IPv6 Dataplane (SRv6). The document also describes some SRv6 OAM mechanisms that can be realized using these building blocks.

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."

Copyright Notice

Copyright (c) 2018 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. Conventions Used in This Document.....	3
2.1. Abbreviations.....	3
2.2. Terminology and Reference Topology.....	4
3. OAM Building Blocks.....	5
3.1. O-flag in Segment Routing Header.....	5
3.2. OAM Segments.....	7

3.2.1. End.OP: OAM Endpoint with Punt.....	7
3.2.2. End.OTP: OAM Endpoint with Timestamp and Punt.....	8
4. OAM Mechanisms.....	8
4.1. Ping.....	9
4.1.1. Classic Ping.....	9
4.1.2. Pinging a SID Function.....	10
4.1.2.1. End-to-end ping using END.OP/ END.OTP.....	11
4.1.2.2. Segment-by-segment ping using O-flag (Proof of Transit).....	11
4.2. Error Reporting.....	13
4.3. Traceroute.....	13
4.3.1. Classic Traceroute.....	13
4.3.2. Traceroute to a SID Function.....	15
4.3.2.1. Hop-by-hop traceroute using END.OP/ END.OTP....	16
4.3.2.2. Tracing SRv6 Overlay.....	17
4.4. Monitoring of SRv6 Paths.....	19
5. Security Considerations.....	20
6. IANA Considerations.....	20
6.1. ICMPv6 type Numbers Registry.....	20
7. References.....	21
7.1. Normative References.....	21
7.2. Informative References.....	22
8. Acknowledgments.....	22

1. Introduction

This document defines building blocks that can be used for Operations, Administration, and Maintenance (OAM) in Segment Routing Networks with IPv6 Dataplane (SRv6). The document also describes some SRv6 OAM mechanisms that can be implemented using these building blocks.

Additional OAM mechanisms will be added in a future revision of the document.

2. Conventions Used in This Document

2.1. Abbreviations

ECMP: Equal Cost Multi-Path.

SID: Segment ID.

SL: Segment Left.

SR: Segment Routing.

SRH: Segment Routing Header.

SRv6: Segment Routing with IPv6 Data plane.

TC: Traffic Class.

UCMP: Unequal Cost Multi-Path.

2.2. Terminology and Reference Topology

This document uses the terminology defined in [I-D.draft-filsfils-spring-srv6-network-programming]. The readers are expected to be familiar with the same.

Throughout the document, the following simple topology is used for illustration.

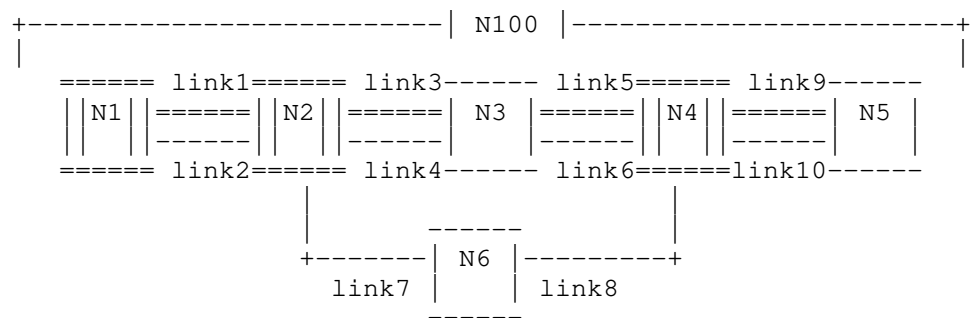


Figure 1 Reference Topology

In the reference topology:

Nodes N1, N2, and N4 are SRv6 capable nodes.

Nodes N3, N5 and N6 are classic IPv6 nodes.

Node N100 is a controller.

Node k has a classic IPv6 loopback address A:k::/128.
A SID at node k with locator block B and function F is represented by B:k:F::

The IPv6 address of the nth Link between node X and Y at the X side is represented as 2001:DB8:X:Y:Xn::, e.g., the IPv6 address of link6 (the 2nd link) between N3 and N4 at N3 in Figure 1 is 2001:DB8:3:4:32::. Similarly, the IPv6 address of link5 (the 1st link between N3 and N4) at node 3 is 2001:DB8:3:4:31::.

B:k:1:: is explicitly allocated as the END function at Node k.

B:k::Cij is explicitly allocated as the END.X function at node k towards neighbor node i via jth Link between node i and node j. e.g., B:2:C31 represents END.X at N2 towards N3 via link3 (the 1st link between N2 and N3). Similarly, B:4:C52 represents the END.X at N4 towards N5 via link10.

<S1, S2, S3> represents a SID list where S1 is the first SID and S3 is the last SID. (S3, S2, S1; SL) represents the same SID list but encoded in the SRH format where the rightmost SID (S1) in the SRH is the first SID and the leftmost SID (S3) in the SRH is the last SID.

(SA, DA) (S3, S2, S1; SL) represents an IPv6 packet, SA is the IPv6 Source Address, DA the IPv6 Destination Address, (S3, S2, S1; SL) is the SRH header that includes the SID list <S1, S2, S3>.

3. OAM Building Blocks

This section defines the various building blocks that can be used to implement OAM mechanisms in SRv6 networks. The following section describes some SRv6 OAM mechanisms that can be implemented using these building blocks.

3.1. O-flag in Segment Routing Header

[I-D. draft-ietf-6man-segment-routing-header] describes the Segment Routing Header (SRH) and how SR capable nodes use it. The draft [I-D. draft-ietf-6man-segment-routing-header] also define an OAM flag (SRH.Flags.O), which indicates that this packet is an operations and management (OAM) packet. The SRH draft also defines the processing rules for the O-flag in the SRH.Flags. The O-flag is one of the OAM building blocks considered in this document.

3.2. OAM Segments

OAM Segment IDs (SIDs) is another components of the building blocks needed to implement SRv6 OAM mechanisms. This document defines a couple of OAM SIDs. Additional SIDs will be added in the later version of the document.

3.2.1. End.OP: OAM Endpoint with Punt

Many scenarios require punting of SRv6 OAM packets at the desired nodes in the network. The "OAM Endpoint with Punt" function (End.OP for short) represents a particular OAM function to implement the punt behavior for an OAM packet. It is described using the pseudocode as follows:

When N receives a packet destined to S and S is a local End.OP SID, N does:

1. Punt the packet to CPU for SW processing (slow-path) ;; Ref1

Ref1: Hardware (microcode) only punts the packet. There is no requirement for the hardware to manipulate any TLV in the SRH (or elsewhere). Software (slow path) implements the required OAM mechanisms.

Please note that in an SRH containing END.OP SID, it is RECOMMENDED to set the SRH.Flags.O-flag = 0.

3.2.2. End.OTP: OAM Endpoint with Timestamp and Punt

Scenarios demanding performance management of an SR policy/ path requires hardware timestamping before hardware punts the packet to the software for OAM processing. The "OAM Endpoint with Timestamp and Punt" function (End.OTP for short) represents an OAM SID function to implement the timestamp and punt behavior for an OAM packet. It is described using the pseudocode as follows:

When N receives a packet destined to S and S is a local End.OTP SID, N does:

- ```

1. Timestamp the packet ;; Ref1
2. Punt the packet to CPU for SW processing (slow-path) ;; Ref2

```

Ref1: Timestamping is done in hardware, as soon as possible during the packet processing.

Ref2: Hardware (microcode) only punts the packet. There is no requirement for the hardware to manipulate any TLV in the SRH (or elsewhere). Software (slow path) implements the required OAM mechanisms.

Please note that in an SRH containing END.OTP SID, it is RECOMMENDED to set the SRH.Flags.O-flag = 0.

## 4. OAM Mechanisms

This section describes how OAM mechanisms can be implemented using the OAM building blocks described in the previous section. Additional OAM mechanisms will be added in a future revision of the document.

[RFC4443] describes Internet Control Message Protocol for IPv6 (ICMPv6) that is used by IPv6 devices for network diagnostic and error reporting purposes. As Segment Routing with IPv6 data plane (SRv6) simply adds a new type of Routing Extension Header, existing ICMPv6 ping mechanisms can be used in an SRv6 network. This section describes the applicability of ICMPv6 in the SRv6 network and how the existing ICMPv6 mechanisms can be used for providing OAM functionality.

Throughout this document, unless otherwise specified, the acronym ICMPv6 refers to multi-part ICMPv6 messages [RFC4884]. The document does not propose any changes to the standard ICMPv6 [RFC4443], [RFC4884] or standard ICMPv4 [RFC792].

#### 4.1. Ping

There is no hardware or software change required for ping operation at the classic IPv6 nodes in an SRv6 network. That includes the classic IPv6 node with ingress, egress or transit roles. Furthermore, no protocol changes are required to the standard ICMPv6 [RFC4443], [RFC4884] or standard ICMPv4 [RFC792]. In other words, existing ICMP ping mechanisms work seamlessly in the SRv6 networks.

The following subsections outline some use cases of the ICMP ping in the SRv6 networks.

##### 4.1.1. Classic Ping

The existing mechanism to ping a remote IP prefix, along the shortest path, continues to work without any modification. The initiator may be an SRv6 node or a classic IPv6 node. Similarly, the egress or transit may be an SRv6 capable node or a classic IPv6 node.

If an SRv6 capable ingress node wants to ping an IPv6 prefix via an arbitrary segment list <S1, S2, S3>, it needs to initiate ICMPv6 ping with an SR header containing the SID list <S1, S2, S3>. This is illustrated using the topology in Figure 1. Assume all the links have IGP metric 10 except both links between node2 and node3, which have IGP metric set to 100. User issues a ping from node N1 to a loopback of node 5, via segment list <B:2:C31, B:4:C52>.

Figure 2 contains sample output for a ping request initiated at node N1 to the loopback address of node N5 via a segment list <B:2:C31, B:4:C52>.

```
> ping A:5:: via segment-list B:2:C31, B:4:C52
```

```
Sending 5, 100-byte ICMP Echos to B5::, timeout is 2 seconds:
```

```
!!!!
```

```
Success rate is 100 percent (5/5), round-trip min/avg/max = 0.625
/0.749/0.931 ms
```

Figure 2 A sample ping output at an SRv6 capable node



All transit nodes process the echo request message like any other data packet carrying SR header and hence do not require any change. Similarly, the egress node (IPv6 classic or SRv6 capable) does not require any change to process the ICMPv6 echo request. For example, in the ping example of Figure 2:

- Node N1 initiates an ICMPv6 ping packet with SRH as follows (A:1::, B:2:C31) (A:5::, B:4:C52, B:2:C31, SL=2, NH = ICMPv6) (ICMPv6 Echo Request).
- Node N2, which is an SRv6 capable node, performs the standard SRH processing. Specifically, it executes the END.X function (B:2:C31) on the echo request packet.
- Node N3, which is a classic IPv6 node, performs the standard IPv6 processing. Specifically, it forwards the echo request based on DA B:4:C52 in the IPv6 header.
- Node N4, which is an SRv6 capable node, performs the standard SRH processing. Specifically, it observes the END.X function (B:4:C52) with PSP (Penultimate Segment POP) on the echo request packet and removes the SRH and forwards the packet across link10 to N5.
- The echo request packet at N5 arrives as an IPv6 packet without a SRH. Node N5, which is a classic IPv6 node, performs the standard IPv6/ ICMPv6 processing on the echo request and responds, accordingly.

#### 4.1.2. Pinging a SID Function

The classic ping described in the previous section cannot be used to ping a remote SID function, as explained using an example in the following.

Consider the case where the user wants to ping the remote SID function B:4:C52, via B:2:C31, from node N1. Node N1 constructs the ping packet (A:1::, B:2:C31) (B:4:C52, B:2:C31, SL=1; NH=ICMPv6) (ICMPv6 Echo Request). The ping fails because the node N4 receives the ICMPv6 echo request with DA set to B:4:C52 but the next header is

ICMPv6, instead of SRH. To solve this problem, the initiator needs to mark the ICMPv6 echo request as an OAM packet.

The OAM packets are identified either by setting the O-flag in SRH or by inserting the END.OP/ END.OTP SIDs at an appropriate place in the SRH. The following illustration uses END.OTP SID but the procedures are equally applicable to the END.OP SID.

In an SRv6 network, the user can exercise two flavors of the ping: end-to-end ping or segment-by-segment ping, as outlined in the following.

#### 4.1.2.1. End-to-end ping using END.OP/ END.OTP

The end-to-end ping illustration uses the END.OTP SID but the procedures are equally applicable to the END.OP SID.

Consider the same example where the user wants to ping a remote SID function B:4:C52 , via B:2:C31, from node N1. To force a punt of the ICMPv6 echo request at the node N4, node N1 inserts the END.OTP SID just before the target SID B:4:C52 in the SRH. The ICMPv6 echo request is processed at the individual nodes along the path as follows:

- Node N1 initiates an ICMPv6 ping packet with SRH as follows (A:1::, B:2:C31)(B:4:C52, B:4:OTP, B:2:C31; SL=2; NH=ICMPv6) (ICMPv6 Echo Request).
- Node N2, which is an SRv6 capable node, performs the standard SRH processing. Specifically, it executes the END.X function (B:2:C31) on the echo request packet.
- Node N3 receives the packet as follows (A:1::, B:4:OTP)(B:4:C52, B:4:OTP, B:2:C31 ; SL=1; NH=ICMPv6) (ICMPv6 Echo Request). Node N3, which is a classic IPv6 node, performs the standard IPv6 processing. Specifically, it forwards the echo request based on DA B:4:OTP in the IPv6 header.
- When node N4 receives the packet (A:1::, B:4:OTP)(B:4:C52, B:4:OTP, B:2:C31 ; SL=1; NH=ICMPv6) (ICMPv6 Echo Request), it processes the END.OTP SID, as described in the pseudocode in Section 3. The packet gets punted to the ICMPv6 process for processing. The ICMPv6 process checks if the next SID in SRH (the target SID B:4:C52) is locally programmed.
- If the target SID is not locally programmed, N4 responds with the ICMPv6 message (Type: "SRv6 OAM (TBA)", Code: "SID not locally implemented (TBA)"); otherwise a success is returned.

#### 4.1.2.2. Segment-by-segment ping using O-flag (Proof of Transit)

Consider the same example where the user wants to ping a remote SID function B:4:C52, via B:2:C31, from node N1. However, in this ping, the node N1 wants to get a response from each segment node in the SRH as a "proof of transit". In other words, in the segment-by-segment ping case, the node N1 expects a response from node N2 and node N4 for their respective local SID function. When a response to O-bit is desired from the last SID in a SID-list, it is the responsibility of the ingress node to use USP as the last SID. E.g., in this example, the target SID B:4:C52 is a USP SID.

To force a punt of the ICMPv6 echo request at node N2 and node N4, node N1 sets the O-flag in SRH. The ICMPv6 echo request is processed at the individual nodes along the path as follows: and

- Node N1 initiates an ICMPv6 ping packet with SRH as follows (A:1::, B:2:C31) (B:4:C52, B:2:C31; SL=1, Flags.O=1; NH=ICMPv6) (ICMPv6 Echo Request).
- When node N2 receives the packet (A:1::, B:2:C31) (B:4:C52, B:2:C31; SL=1, Flags.O=1; NH=ICMPv6) (ICMPv6 Echo Request) packet, it processes the O-flag in SRH, as described in the pseudocode in Section 3. A time-stamped copy of the packet gets punted to the ICMPv6 process for processing. Node N2 continues to apply the B:2:C31 SID function on the original packet and forwards it, accordingly. As B:4:C52 is a USP SID, N2 does not remove the SRH. The ICMPv6 process at node N2 checks if its local SID (B:2:C31) is locally programmed or not and responds to the ICMPv6 Echo Request.
- If the target SID is not locally programmed, N4 responds with the ICMPv6 message (Type: "SRv6 OAM (TBA)", Code: "SID not locally implemented (TBA)"); otherwise a success is returned. Please note that, as mentioned in Section 3, if node N2 does not support the O-flag, it simply ignores it and process the local SID, B:2:C31.
- Node N3, which is a classic IPv6 node, performs the standard IPv6 processing. Specifically, it forwards the echo request based on DA B:4:C52 in the IPv6 header.
- When node N4 receives the packet (A:1::, B:4:C52) (B:4:C52, B:2:C31; SL=0, Flags.O=1; NH=ICMPv6) (ICMPv6 Echo Request), it processes the O-flag in SRH, as described in the pseudocode in Section 3. A time-stamped copy of the packet gets punted to the ICMPv6 process for processing. The ICMPv6 process at node N4 checks if its local SID (B:2:C31) is locally programmed or not and responds to the ICMPv6 Echo Request. If the target SID is not locally programmed, N4 responds with the ICMPv6 message (Type: "SRv6 OAM (TBA)", Code: "SID not locally implemented (TBA)"); otherwise a success is returned.

Support for O-flag is part of node capability advertisement. That enables node N1 to know which segment nodes are capable of responding to the ICMPv6 echo request. Node N1 processes the echo responses and presents data to the user, accordingly.

Please note that segment-by-segment ping can be used to address proof of transit use-case.

#### 4.2. Error Reporting

Any IPv6 node can use ICMPv6 control messages to report packet processing errors to the host that originated the datagram packet. To name a few such scenarios:

- If the router receives an undeliverable IP datagram, or
- If the router receives a packet with a Hop Limit of zero, or
- If the router receives a packet such that if the router decrements the packet's Hop Limit it becomes zero, or
- If the router receives a packet with problem with a field in the IPv6 header or the extension headers such that it cannot complete processing the packet, or
- If the router cannot forward a packet because the packet is larger than the MTU of the outgoing link.

In the scenarios listed above, the ICMPv6 response also contains the IP header, IP extension headers and leading payload octets of the "original datagram" to which the ICMPv6 message is a response. Specifically, the "Destination Unreachable Message", "Time Exceeded Message", "Packet Too Big Message" and "Parameter Problem Message" ICMPV6 messages can contain as much of the invoking packet as possible without the ICMPv6 packet exceeding the minimum IPv6 MTU [RFC4443], [RFC4884]. In an SRv6 network, the copy of the invoking packet contains the SR header. The packet originator can use this information for diagnostic purposes. For example, traceroute can use this information as detailed in the following.

#### 4.3. Traceroute

There is no hardware or software change required for traceroute operation at the classic IPv6 nodes in an SRv6 network. That includes the classic IPv6 node with ingress, egress or transit roles. Furthermore, no protocol changes are required to the standard traceroute operations. In other words, existing traceroute mechanisms work seamlessly in the SRv6 networks.

The following subsections outline some use cases of the traceroute in the SRv6 networks.

##### 4.3.1. Classic Traceroute

The existing mechanism to traceroute a remote IP prefix, along the shortest path, continues to work without any modification. The initiator may be an SRv6 node or a classic IPv6 node. Similarly, the egress or transit may be an SRv6 node or a classic IPv6 node.

If an SRv6 capable ingress node wants to traceroute to IPv6 prefix via an arbitrary segment list <S1, S2, S3>, it needs to initiate traceroute probe with an SR header containing the SID list <S1, S2, S3>. That is illustrated using the topology in Figure 1. Assume all the links have IGP metric 10 except both links between node2 and node3, which have IGP metric set to 100. User issues a traceroute from node N1 to a loopback of node 5, via segment list <B:2:C31, B:4:C52>. Figure 3 contains sample output for the traceroute request.

```
> traceroute A:5:: via segment-list B:2:C31, B:4:C52
```

Tracing the route to B5::

```
1 2001:DB8:1:2:21:: 0.512 msec 0.425 msec 0.374 msec
 SRH: (A:5::, B:4:C52, B:2:C31, SL=2)

2 2001:DB8:2:3:31:: 0.721 msec 0.810 msec 0.795 msec
 SRH: (A:5::, B:4:C52, B:2:C31, SL=1)

3 2001:DB8:3:4::41:: 0.921 msec 0.816 msec 0.759 msec
 SRH: (A:5::, B:4:C52, B:2:C31, SL=1)

4 2001:DB8:4:5::52:: 0.879 msec 0.916 msec 1.024 msec
```

Figure 3 A sample traceroute output at an SRv6 capable node

Please note that information for hop2 is returned by N3, which is a classic IPv6 node. Nonetheless, the ingress node is able to display SR header contents as the packet travels through the IPv6 classic node. This is because the "Time Exceeded Message" ICMPv6 message can contain as much of the invoking packet as possible without the ICMPv6 packet exceeding the minimum IPv6 MTU [RFC4443]. The SR header is also included in these ICMPv6 messages initiated by the classic IPv6 transit nodes that are not running SRv6 software. Specifically, a node generating ICMPv6 message containing a copy of the invoking packet does not need to understand the extension header(s) in the invoking packet.

The segment list information returned for hop1 is returned by N2, which is an SRv6 capable node. Just like for hop2, the ingress node is able to display SR header contents for hop1.

There is no difference in processing of the traceroute probe at an IPv6 classic node and an SRv6 capable node. Similarly, both IPv6 classic and SRv6 capable nodes use the address of the interface on which probe was received as the source address in the ICMPv6

response. ICMP extensions defined in [RFC5837] can be used to also display information about the IP interface through which the datagram would have been forwarded had it been forwardable, and the IP next hop to which the datagram would have been forwarded, the IP interface upon which a datagram arrived, the sub-IP component of an IP interface upon which a datagram arrived.

The information about the IP address of the incoming interface on which the traceroute probe was received by the reporting node is very useful. This information can also be used to verify if SID functions B:2:C31 and B:4:C52 are executed correctly by N2 and N4, respectively. Specifically, the information displayed for hop2 contains the incoming interface address 2001:DB8:2:3:31:: at N3. This matches with the expected interface bound to END.X function B:2:C31 (link3). Similarly, the information displayed for hop5 contains the incoming interface address 2001:DB8:4:5::52:: at N5. This matches with the expected interface bound to the END.X function B:4:C52 (link10).

#### 4.3.2. Traceroute to a SID Function

The classic traceroute described in the previous section cannot be used to traceroute a remote SID function, as explained using an example in the following.

Consider the case where the user wants to traceroute the remote SID function B:4:C52, via B:2:C31, from node N1. The trace route fails at N4. This is because the node N4 trace route probe where next header is UDP or ICMPv6, instead of SRH (even though the hop limit is set to 1). To solve this problem, the initiator needs to mark the ICMPv6 echo request as an OAM packet.

The OAM packets are identified either by setting the O-flag in SRH or by inserting the END.OTP SID at an appropriate place in the SRH.

In an SRv6 network, the user can exercise two flavors of the traceroute: hop-by-hop traceroute or overlay traceroute.

- In hop-by-hop traceroute, user gets responses from all nodes including classic IPv6 transit nodes, SRv6 capable transit nodes as well as SRv6 capable segment endpoints. E.g., consider the example where the user wants to traceroute to a remote SID function B:4:C52 , via B:2:C31, from node N1. The traceroute

output will also display information about node3, which is a transit (underlay) node.

- The overlay traceroute, on the other hand, does not trace the underlay nodes. In other words, the overlay traceroute only displays the nodes that acts as SRv6 segments along the route. I.e., in the example where the user wants to traceroute to a remote SID function B:4:C52 , via B:2:C31, from node N1, the overlay traceroute would only display the traceroute information from node N2 and node N2 and will not display information from node 3.

#### 4.3.2.1. Hop-by-hop traceroute using END.OP/ END.OTP

In this section, hop-by-hop traceroute to a SID function is exemplified using UDP probes. However, the procedure is equally applicable to other implementation of traceroute mechanism. Furthermore, the illustration uses the END.OTP SID but the procedures are equally applicable to the END.OP SID

Consider the same example where the user wants to traceroute to a remote SID function B:4:C52 , via B:2:C31, from node N1. To force a punt of the traceroute probe only at the node N4, node N1 inserts the END.OTP SID just before the target SID B:4:C52 in the SRH. The traceroute probe is processed at the individual nodes along the path as follows.

- Node N1 initiates a traceroute probe packet with a monotonically increasing value of hop count and SRH as follows (A:1::, B:2:C31) (B:4:C52, B:4:OTP, B:2:C31; SL=2; NH=UDP) (Traceroute probe).
- When node N2 receives the packet with hop-count = 1, it processes the hop count expiry. Specifically, the node N2 responses with the ICMPv6 message (Type: "Time Exceeded", Code: "Time to Live exceeded in Transit").
- When Node N2 receives the packet with hop-count > 1, it performs the standard SRH processing. Specifically, it executes the END.X function (B:2:C31) on the traceroute probe.
- When node N3, which is a classic IPv6 node, receives the packet (A:1::, B:4:OTP) (B:4:C52, B:4:OTP, B:2:C31 ; HC=1, SL=1; NH=UDP) (Traceroute probe) with hop-count = 1, it processes the hop count expiry. Specifically, the node N3 responses with the ICMPv6 message (Type: "Time Exceeded", Code: "Time to Live exceeded in Transit").
- When node N3, which is a classic IPv6 node, receives the packet with hop-count > 1, it performs the standard IPv6 processing. Specifically, it forwards the traceroute probe based on DA B:4:OTP in the IPv6 header.

- When node N4 receives the packet (A:1::, B:4:OTP) (B:4:C52, B:4:OTP, B:2:C31 ; SL=1; HC=1, NH=UDP) (Traceroute probe), it processes the END.OTP SID, as described in the pseudocode in Section 3. The packet gets punted to the traceroute process for processing. The traceroute process checks if the next SID in SRH (the target SID B:4:C52) is locally programmed. If the target SID B:4:C52 is locally programmed, node N4 responses with the ICMPv6 message (Type: Destination unreachable, Code: Port Unreachable). If the target SID B:4:C52 is not a local SID, node N4 silently drops the traceroute probe.

Figure 4 displays a sample traceroute output for this example.

```
> traceroute srv6 B:4:C52 via segment-list B:2:C31
```

Tracing the route to SID function B:4:C52

```
1 2001:DB8:1:2:21 0.512 msec 0.425 msec 0.374 msec
 SRH: (B:4:C52, B:4:OTP, B:2:C31; SL=2)

2 2001:DB8:2:3:31 0.721 msec 0.810 msec 0.795 msec
 SRH: (B:4:C52, B:4:OTP, B:2:C31; SL=1)

3 2001:DB8:3:4::41 0.921 msec 0.816 msec 0.759 msec
 SRH: (B:4:C52, B:4:OTP, B:2:C31; SL=1)
```

Figure 4 A sample output for hop-by-hop traceroute to a SID function

#### 4.3.2.2. Tracing SRv6 Overlay

The overlay traceroute does not trace the underlay nodes, i.e., only displays the nodes that acts as SRv6 segments along the path. This is achieved by setting the SRH.Flags.0 bit.

In this section, overlay traceroute to a SID function is exemplified using UDP probes. However, the procedure is equally applicable to other implementation of traceroute mechanism.

Consider the same example where the user wants to traceroute to a remote SID function B:4:C52 , via B:2:C31, from node N1.

- Node N1 initiates a traceroute probe with SRH as follows (A:1::, B:2:C31) (B:4:C52, B:2:C31; HC=64, SL=1, Flags.0=1; NH=UDP) (Traceroute Probe). Please note that the hop-count is



set to 64 to skip the underlay nodes from tracing. The O-flag in SRH is set to make the overlay nodes (nodes processing the SRH) respond.

- When node N2 receives the packet (A:1::, B:2:C31) (B:4:C52, B:2:C31; SL=1, HC=64, Flags.O=1; NH=UDP) (Traceroute Probe), it processes the O-flag in SRH, as described in the pseudocode in Section 3. A time-stamped copy of the packet gets punted to the traceroute process for processing. Node N2 continues to apply the B:2:C31 SID function on the original packet and forwards it, accordingly. As SRH.Flags.O=1, Node N2 also disables the PSP flavor, i.e., does not remove the SRH. The traceroute process at node N2 checks if its local SID (B:2:C31) is locally programmed. If the SID is not locally programmed, it silently drops the packet. Otherwise, it performs the egress check by looking at the SL value in SRH.
- As SL is not equal to zero (i.e., it's not egress node), node N2 responds with the ICMPv6 message (Type: "SRv6 OAM (TBA)", Code: "O-flag punt at Transit (TBA)"). Please note that, as mentioned in Section 3, if node N2 does not support the O-flag, it simply ignores it and processes the local SID, B:2:C31.
- When node N3 receives the packet (A:1::, B:4:C52) (B:4:C52, B:2:C31; SL=0, HC=63, Flags.O=1; NH=UDP) (Traceroute Probe), performs the standard IPv6 processing. Specifically, it forwards the traceroute probe based on DA B:4:C52 in the IPv6 header. Please note that there is no hop-count expiration at the transit nodes.
- When node N4 receives the packet (A:1::, B:4:C52) (B:4:C52, B:2:C31; SL=0, HC=62, Flags.O=1; NH=UDP) (Traceroute Probe), it processes the O-flag in SRH, as described in the pseudocode in Section 3. A time-stamped copy of the packet gets punted to the traceroute process for processing. The traceroute process at node N4 checks if its local SID (B:2:C31) is locally programmed. If the SID is not locally programmed, it silently drops the packet. Otherwise, it performs the egress check by looking at the SL value in SRH. As SL is equal to zero (i.e., N4 is the egress node), node N4 tries to consume the UDP probe. As UDP probe is set to access an invalid port, the node N4 responds with the ICMPv6 message (Type: Destination unreachable, Code: Port Unreachable).

Figure 5 displays a sample overlay traceroute output for this example. Please note that the underlay node N3 does not appear in the output.

```
> traceroute srv6 B:4:C52 via segment-list B:2:C31
```

```
Tracing the route to SID function B:4:C52
```

- 1 2001:DB8:1:2:21:: 0.512 msec 0.425 msec 0.374 msec  
SRH: (B:4:C52, B:4:OTP, B:2:C31; SL=2)
- 2 2001:DB8:3:4::41:: 0.921 msec 0.816 msec 0.759 msec  
SRH: (B:4:C52, B:4:OTP, B:2:C31; SL=1)

Figure 5 A sample output for overlay traceroute to a SID function

#### 4.5. Monitoring of SRv6 Paths

In the recent past, network operators are interested in performing network OAM functions in a centralized manner. Various data models like YANG are available to collect data from the network and manage it from a centralized entity.

SR technology enables a centralized OAM entity to perform path monitoring from centralized OAM entity without control plane intervention on monitored nodes. [I.D-draft-ietf-spring-oam-usecase] describes such a centralized OAM mechanism. Specifically, the draft describes a procedure that can be used to perform path continuity check between any nodes within an SR domain from a centralized monitoring system, with minimal or no control plane intervene on the nodes. However, the draft focuses on SR networks with MPLS data plane. The same concept applies to the SRv6 networks. This document describes how the concept can be used to perform path monitoring in an SRv6 network. This document describes how the concept can be used to perform path monitoring in an SRv6 network as follows.

In the above reference topology, N100 is the centralized monitoring system implementing an END function B:100:1::. In order to verify a segment list <B:2:C31, B:4:C52>, N100 generates a probe packet with SRH set to (B:100:1::, B:4:C52, B:2:C31, SL=2). The controller routes the probe packet towards the first segment, which is B:2:C31. N2 performs the standard SRH processing and forward it over link3 with the DA of IPv6 packet set to B:4:C52. N4 also performs the normal SRH processing and forward it over link10 with the DA of IPv6 packet set to B:100:1::. This makes the probe loops back to the centralized monitoring system.

In the reference topology in Figure 1, N100 uses an IGP protocol like OSPF or ISIS to get the topology view within the IGP domain. N100 can also use BGP-LS to get the complete view of an inter-domain topology. In other words, the controller leverages the visibility of the topology to monitor the paths between the various endpoints without control plane intervention required at the monitored nodes.

## 5. Security Considerations

This document does not define any new protocol extensions and relies on existing procedures defined for ICMP. This document does not impose any additional security challenges to be considered beyond security considerations described in [RFC4884], [RFC4443], [RFC792] and RFCs that updates these RFCs.

## 6. IANA Considerations

### 6.1. ICMPv6 type Numbers Registry

This document defines one ICMPv6 Message, a type that has been allocated from the "ICMPv6 'type' Numbers" registry of [RFC4443].

Specifically, it requests to add the following to the "ICMPv6 Type Numbers" registry:

TBA (suggested value: 162) SRv6 OAM Message.

The document also requests the creation of a new IANA registry to the

"ICMPv6 'Code' Fields" against the "ICMPv6 Type Numbers TBA - SRv6 OAM Message" with the following codes:

| Code | Name                           | Reference     |
|------|--------------------------------|---------------|
| 0    | No Error                       | This document |
| 1    | SID is not locally implemented | This document |
| 2    | O-flag punt at Transit         | This document |

### 6.3. SRv6 OAM Endpoint Types

This I-D requests to IANA to allocate, within the "SRv6 Endpoint Behaviors Registry" sub-registry belonging to the top-level "Segment-routing with IPv6 dataplane (SRv6) Parameters" registry [I-D.filsfils-spring-srv6-network-programming], the following allocations:

| Value (Suggested Value) | Endpoint Behavior | Reference |
|-------------------------|-------------------|-----------|
| TBA (30)                | End.OP            | [This.ID] |
| TBA (31)                | End.OTP           | [This.ID] |

## 7. References

### 7.1. Normative References

- [RFC792] J. Postel, "Internet Control Message Protocol", RFC 792, September 1981.
- [RFC4443] A. Conta, S. Deering, M. Gupta, Ed., "Internet Control

Message Protocol (ICMPv6) for the Internet Protocol  
Version 6 (IPv6) Specification", RFC 4443, March 2006.

- [RFC4884] R. Bonica, D. Gan, D. Tappan, C. Pignataro, "Extended ICMP to Support Multi-Part Messages", RFC 4884, April 2007.
- [RFC5837] A. Atlas, Ed., R. Bonica, Ed., C. Pignataro, Ed., N. Shen, JR. Rivers, "Extending ICMP for Interface and Next-Hop Identification", RFC 5837, April 2010.
- [I-D.filsfils-spring-srv6-network-programming] C. Filsfils, et al., "SRv6 Network Programming", draft-filsfils-spring-srv6-network-programming, work in progress.
- [I-D.6man-segment-routing-header] Previdi, S., Filsfils, et al, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header, work in progress.

## 7.2. Informative References

- [I-D.bashandy-isis-srv6-extensions] IS-IS Extensions to Support Routing over IPv6 Dataplane. L. Ginsberg, P. Psenak, C. Filsfils, A. Bashandy, B. Decraene, Z. Hu, draft-bashandy-isis-srv6-extensions, work in progress.
- [I-D.dawra-idr-bgpls-srv6-ext] G. Dawra, C. Filsfils, K. Talaulikar, et al., BGP Link State extensions for IPv6 Segment Routing (SRv6), draft-dawra-idr-bgpls-srv6-ext, work in progress.
- [I-D.ietf-spring-oam-usecase] A Scalable and Topology-Aware MPLS Dataplane Monitoring System. R. Geib, C. Filsfils, C. Pignataro, N. Kumar, draft-ietf-spring-oam-usecase, work in progress.
- [I-D.brockners-inband-oam-data] F. Brockners, et al., "Data Formats for In-situ OAM", draft-brockners-inband-oam-data, work in progress.
- [I-D.brockners-inband-oam-transport] F.Brockners, et al., "Encapsulations for In-situ OAM Data", draft-brockners-inband-oam-transport, work in progress.
- [I-D.brockners-inband-oam-requirements] F.Brockners, et al., "Requirements for In-situ OAM", draft-brockners-inband-oam-requirements, work in progress.
- [I-D.spring-segment-routing-policy] Filsfils, C., et al., "Segment Routing Policy for Traffic Engineering", draft-filsfils-spring-segment-routing-policy, work in progress.

## 8. Acknowledgments

To be added.



Authors' Addresses

Clarence Filsfils  
Cisco Systems, Inc.  
Email: cfilsfil@cisco.com

Zafar Ali  
Cisco Systems, Inc.  
Email: zali@cisco.com

Nagendra Kumar  
Cisco Systems, Inc.  
Email: naikumar@cisco.com

Carlos Pignataro  
Cisco Systems, Inc.  
Email: cpignata@cisco.com

Faisal Iqbal  
Cisco Systems, Inc.  
Email: faiqbal@cisco.com

Rakesh Gandhi  
Cisco Systems, Inc.  
Canada  
Email: rgandhi@cisco.com

John Leddy  
Comcast  
Email: John\_Leddy@cable.comcast.com

Robert Raszuk  
Bloomberg LP  
731 Lexington Ave  
New York City, NY10022, USA  
Email: robert@raszuk.net

Satoru Matsushima  
SoftBank  
Japan  
Email: satoru.matsushima@g.softbank.co.jp

Daniel Voyer  
Bell Canada  
Email: daniel.voyer@bell.ca

Gaurav Dawra  
LinkedIn  
Email: gdawra.ietf@gmail.com

Bart Peirens  
Proximus  
Email: bart.peirens@proximus.com

Mach Chen  
Huawei  
Email: mach.chen@huawei.com

Gaurav Naik  
Drexel University  
United States of America  
Email: gn@drexel.edu





6man  
Internet-Draft  
Intended status: Standards Track  
Expires: 16 May 2022

R. Bonica  
Juniper Networks  
Y. Kamite  
NTT Communications Corporation  
A. Alston  
D. Henriques  
Liquid Telecom  
L. Jalil  
Verizon  
12 November 2021

The IPv6 Compact Routing Header (CRH)  
draft-bonica-6man-comp-rtg-hdr-27

## Abstract

This document defines two new Routing header types. Collectively, they are called the Compact Routing Headers (CRH). Individually, they are called CRH-16 and CRH-32.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 16 May 2022.

## Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components

extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                                                                             |    |
|-----------------------------------------------------------------------------|----|
| 1. Introduction . . . . .                                                   | 2  |
| 2. Requirements Language . . . . .                                          | 3  |
| 3. The Compressed Routing Headers (CRH) . . . . .                           | 3  |
| 4. The CRH Forwarding Information Base (CRH-FIB) . . . . .                  | 4  |
| 5. Processing Rules . . . . .                                               | 6  |
| 5.1. Computing Minimum CRH Length . . . . .                                 | 7  |
| 5.2. CRH Removal Procedure . . . . .                                        | 8  |
| 6. Mutability . . . . .                                                     | 8  |
| 7. Applications And SIDs . . . . .                                          | 8  |
| 8. Management Considerations . . . . .                                      | 9  |
| 9. Security Considerations . . . . .                                        | 9  |
| 10. Implementation and Deployment Status . . . . .                          | 9  |
| 11. IANA Considerations . . . . .                                           | 10 |
| 12. Acknowledgements . . . . .                                              | 10 |
| 13. Contributors . . . . .                                                  | 10 |
| 14. References . . . . .                                                    | 11 |
| 14.1. Normative References . . . . .                                        | 11 |
| 14.2. Informative References . . . . .                                      | 11 |
| Appendix A. CRH Processing Examples . . . . .                               | 12 |
| A.1. The SID List Contains One Entry For Each Segment In The Path . . . . . | 13 |
| A.2. The SID List Omits The First Entry In The Path . . . . .               | 14 |
| Appendix B. A Packet Recycling Use-Case . . . . .                           | 14 |
| Authors' Addresses . . . . .                                                | 15 |

## 1. Introduction

IPv6 [RFC8200] source nodes use Routing headers to specify the path that a packet takes to its destination. The IETF has defined several Routing header types [IANA-RH]. This document defines two new Routing header types. Collectively, they are called the Compact Routing Headers (CRH). Individually, they are called CRH-16 and CRH-32.

The CRH allows IPv6 source nodes to specify the path that a packet takes to its destination. The CRH:

- \* Can be encoded in relatively few bytes.
- \* Is designed to operate within a network domain. (See Section 9).

The following are reasons for encoding the CRH in as few bytes as possible:

- \* Many ASIC-based forwarders copy headers from buffer memory to on-chip memory. As header sizes increase, so does the cost of this copy.
- \* Because Path MTU Discovery (PMTUD) [RFC8201] is not entirely reliable, many IPv6 hosts refrain from sending packets larger than the IPv6 minimum link MTU (i.e., 1280 bytes). When packets are small, the overhead imposed by large Routing Headers is excessive.

## 2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 3. The Compressed Routing Headers (CRH)

Both CRH versions (i.e., CRH-16 and CRH-32) contain the following fields:

- \* Next Header - Defined in [RFC8200].
- \* Hdr Ext Len - Defined in [RFC8200].
- \* Routing Type - Defined in [RFC8200]. Value TBD by IANA. (For CRH-16, the suggested value is 5. For CRH-32, the suggested value is 6.)
- \* Segments Left - Defined in [RFC8200].
- \* Type-specific Data - Described in [RFC8200].

In the CRH, the Type-specific data field contains a list of Segment Identifiers (SIDs). Each SID represents both of the following:

- \* A segment of the path that the packet takes to its destination.
- \* An entry in the CRH Forwarding Information Base (CRH-FIB) (Section 4).

SIDs are listed in reverse order. So, the first SID in the list represents the final segment in the path. Because segments are listed in reverse order, the Segments Left field can be used as an index into the SID list. In this document, the "current SID" is the SID list entry referenced by the Segments Left field.

The first segment in the path can be omitted from the list. See Appendix A for examples.

In the CRH-16 (Figure 1), each SID is encoded in 16-bits. In the CRH-32 (Figure 2), each SID is encoded in 32-bits.

In all cases, the CRH MUST end on a 64-bit boundary. So, the Type-specific data field MUST be padded with zeros if the CRH would otherwise not end on a 64-bit boundary.

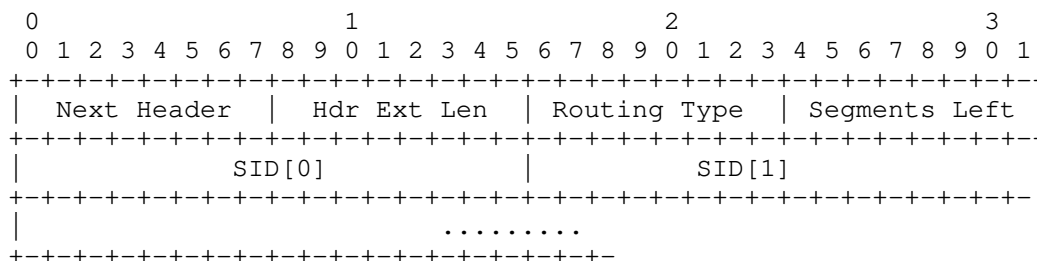


Figure 1: CRH-16

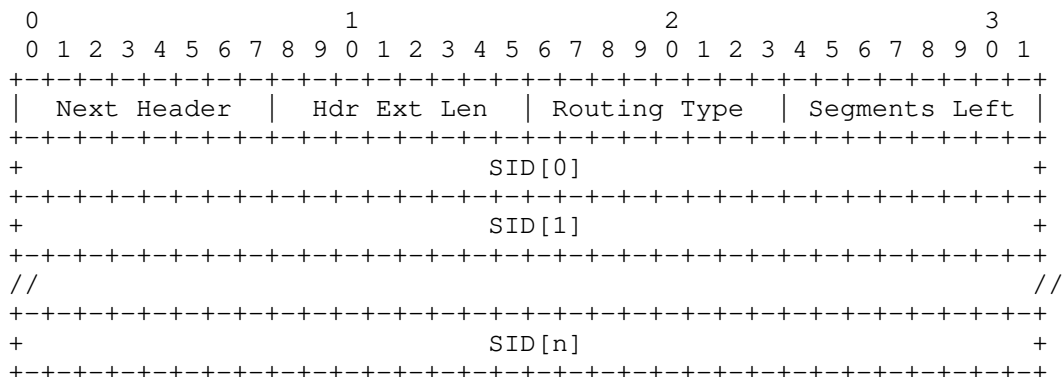


Figure 2: CRH-32

#### 4. The CRH Forwarding Information Base (CRH-FIB)

Each SID identifies a CRH-FIB entry.

Each CRH-FIB entry contains:

- \* An IPv6 address (optional).
- \* A topological function.
- \* Arguments for the topological function (optional).
- \* Flags.
- \* A service function (optional).
- \* Arguments for the service function (optional).

The IPv6 address can represent either:

- \* An interface on the next segment endpoint.
- \* An SRv6 SID [RFC8986], instantiated on the next segment endpoint.

The first ten bits of the IPv6 address MUST NOT be fe00. That prefix is reserved for link-local [RFC6890] addresses.

The topological function specifies how the processing node forwards the packet to the next segment endpoint. The following are examples:

- \* Forward the packet through the least-cost path to the next segment endpoint.
- \* Forward the packet through a specified interface.
- \* Encapsulate the packet in another IPv6 header of any type (e.g., MPLS, IPv6) and forward either through the least cost path or a specified interface.
- \* Recycle the packet, as if the node had forwarded to one of its own interfaces. When recycling is complete, process the next SID. See Appendix B for a packet recycling use-case.

Some topological functions require parameters. For example, a topological function might require a parameter that identifies the interface through which the packet should be forwarded.

The following flags are defined:

- \* The PSP flag indicates whether the penultimate segment endpoint (i.e., the node that sets Segments Left to 0) MAY remove the CRH.

- \* The OAM flag indicates whether the processing node should invoke OAM procedures for which it is configured.

The service function is optional. If present, it invokes a node specific procedure. The following are examples of node specific procedures:

- \* Emit telemetry.
- \* Subject the packet's payload to a firewall rule.
- \* Replicate the packet, forwarding one copy and retaining the other for sampling, analysis, or other purposes.

Node specific procedures are not subject to standardization. A node can support any number of node specific procedures and associate them with any SIDs.

Some service functions require parameters. For example, an instruction to emit telemetry might require an IP address to which telemetry should be sent.

The CRH-FIB can be populated:

- \* By an operator, using a Command Line Interface (CLI).
- \* By a controller, using the Path Computation Element (PCE) Communication Protocol (PCEP) [RFC5440] or the Network Configuration Protocol (NETCONF) [RFC6241].
- \* By a distributed routing protocol [ISO10589-Second-Edition], [RFC5340], [RFC4271].

## 5. Processing Rules

The following rules describe CRH processing:

- \* If Segments Left equals 0, skip over the CRH and process the next header in the packet.
- \* If Hdr Ext Len indicates that the CRH is larger than the implementation can process, discard the packet and send an ICMPv6 [RFC4443] Parameter Problem, Code 0, message to the Source Address, pointing to the Hdr Ext Len field.
- \* Compute L, the minimum CRH length ( Section 5.1).

- \* If L is greater than Hdr Ext Len, discard the packet and send an ICMPv6 Parameter Problem, Code 0, message to the Source Address, pointing to the Segments Left field.
- \* Decrement Segments Left.
- \* Search for the current SID in the CRH-FIB. In this document, the "current SID" is the SID list entry referenced by the Segments Left field.
- \* If the search does not return a CRH-FIB entry, discard the packet and send an ICMPv6 Parameter Problem, Code 0, message to the Source Address, pointing to the current SID.
- \* If Segments Left is greater than 0 and the CRH-FIB entry contains a multicast address, discard the packet and send an ICMPv6 Parameter Problem, Code 0, message to the Source Address, pointing to the current SID.
- \* If present, copy the IPv6 address from the CRH-FIB entry to the Destination Address field in the IPv6 header.
- \* Decrement the IPv6 Hop Limit.
- \* If the CRH-FIB entry contains a service function, execute it.
- \* If Segments Left is equal to zero, and the PSP flag in the CRH-FIB entry is set, execute the CRH removal procedure ( Section 5.2).
- \* Submit the packet, its topological function and its parameters to the IPv6 module. See NOTE.

NOTE: By default, the IPv6 module determines the next-hop and forwards the packet. However, the topological function may elicit another behavior. For example, the IPv6 module may forward the packet through a specified interface.

#### 5.1. Computing Minimum CRH Length

The algorithm described in this section accepts the following CRH fields as its input parameters:

- \* Routing Type (i.e., CRH-16 or CRH-32).
- \* Segments Left.

It yields L, the minimum CRH length. The minimum CRH length is measured in 8-octet units, not including the first 8 octets.



```
<CODE BEGINS>
switch(Routing Type) {
 case CRH-16:
 if (Segments Left <= 2)
 return(0)
 sidsBeyondFirstWord = Segments Left - 2;
 sidPerWord = 4;
 case CRH-32:
 if (Segments Left <= 1)
 return(0)
 sidsBeyondFirstWord = Segments Left - 1;
 sidsPerWord = 2;
 case default:
 return(0xFF);
}

words = sidsBeyondFirstWord div sidsPerWord;
if (sidsBeyondFirstWord mod sidsPerWord)
 words++;

return(words)
<CODE ENDS>
```

## 5.2. CRH Removal Procedure

The processing node SHOULD execute the following procedure, if it is capable of doing so:

- \* Update the Next Header field in the header preceding the CRH using a value taken from the Next Header field in the CRH.
- \* Decrease the Payload Length field in the IPv6 header by  $8 \times (x+1)$ , where value of  $x$  is equal to the value of the Hdr Ext Len field in the CRH.
- \* Remove the CRH from the IPv6 header chain.

## 6. Mutability

In the CRH, the Segments Left field is mutable. All remaining fields are immutable.

## 7. Applications And SIDs

A CRH contains one or more SIDs. Each SID is processed by exactly one node.

Therefore, a SID is not required to have domain-wide significance. Applications can:

- \* Allocate SIDs so that they have domain-wide significance.
- \* Allocate SIDs so that they have node-local significance.

## 8. Management Considerations

PING and TRACEROUTE [RFC2151] both operate correctly in the presence of the CRH.

## 9. Security Considerations

Networks that process the CRH MUST NOT accept packets containing the CRH from untrusted sources. Their border routers SHOULD discard packets that satisfy the following criteria:

- \* The packet contains a CRH
- \* The Segments Left field in the CRH has a value greater than 0
- \* The Destination Address field in the IPv6 header represents an interface that resides inside of the network.

Many border routers cannot filter packets based upon the Segments Left value. These border routers MAY discard packets that satisfy the following criteria:

- \* The packet contains a CRH
- \* The Destination Address field in the IPv6 header represents an interface that resides inside of the network.

## 10. Implementation and Deployment Status

Juniper Networks has produced experimental implementations of the CRH on:

- \* A LINUX-based software platform
- \* The MX-series (ASIC-based) router

Liquid Telecom has deployed the CRH, on a limited basis, in their network. Other experimental deployments are in progress.

## 11. IANA Considerations

This document makes the following registrations in the "Internet Protocol Version 6 (IPv6) Parameters" "Routing Types" subregistry maintained by IANA:

| Value | Description | Reference     |
|-------|-------------|---------------|
| 5     | CRH-16      | This document |
| 6     | CRH-32      | This document |

## 12. Acknowledgements

Thanks to Dr. Vanessa Ameen, Fernando Gont, Naveen Kottapalli, Joel Halpern, Tony Li, Gerald Schmidt, Nancy Shaw, Ketan Talaulikar, and Chandra Venkatraman for their contributions to this document.

## 13. Contributors

Gang Chen

Baidu

No.10 Xibeiwang East Road Haidian District

Beijing 100193 P.R. China

Email: phdgang@gmail.com

Yifeng Zhou

ByteDance

Building 1, AVIC Plaza, 43 N 3rd Ring W Rd Haidian District

Beijing 100000 P.R. China

Email: yifeng.zhou@bytedance.com

Gyan Mishra

Verizon

Silver Spring, Maryland, USA

Email: hayabusagsm@gmail.com

## 14. References

### 14.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <<https://www.rfc-editor.org/info/rfc8201>>.

### 14.2. Informative References

- [IANA-RH] IANA, "Routing Headers", <<https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#ipv6-parameters-3>>.
- [ISO10589-Second-Edition] International Organization for Standardization, "Intermediate system to Intermediate system intra-domain routing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode Network Service (ISO 8473)", ISO/IEC 10589:2002, Second Edition, November 2001.

- [RFC2151] Kessler, G. and S. Shepard, "A Primer On Internet and TCP/IP Tools and Utilities", FYI 30, RFC 2151, DOI 10.17487/RFC2151, June 1997, <<https://www.rfc-editor.org/info/rfc2151>>.
- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <<https://www.rfc-editor.org/info/rfc4271>>.
- [RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, DOI 10.17487/RFC5340, July 2008, <<https://www.rfc-editor.org/info/rfc5340>>.
- [RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <<https://www.rfc-editor.org/info/rfc5440>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <<https://www.rfc-editor.org/info/rfc6890>>.
- [RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <<https://www.rfc-editor.org/info/rfc8986>>.

## Appendix A. CRH Processing Examples

This appendix demonstrates CRH processing in the following scenarios:

- \* The SID list contains one entry for each segment in the path (Appendix A.1).
- \* The SID list omits the first entry in the path (Appendix A.2).

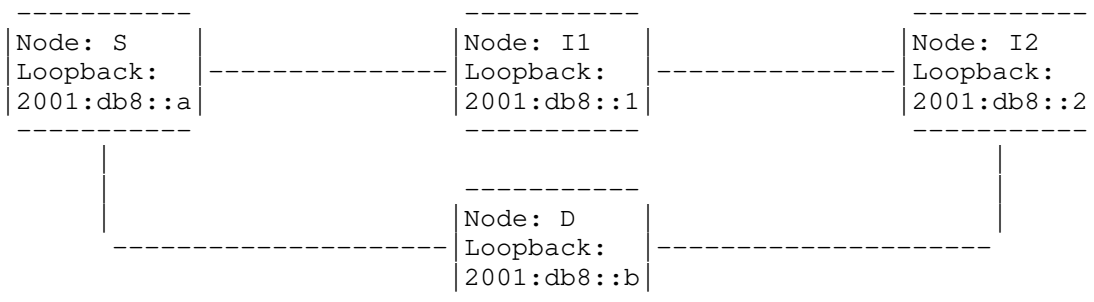


Figure 3: Reference Topology

Figure 3 provides a reference topology that is used in all examples.

| SID | IPv6 Address | Forwarding Method |
|-----|--------------|-------------------|
| 2   | 2001:db8::2  | Least-cost path   |
| 11  | 2001:db8::b  | Least-cost path   |

Table 1: Node SIDs

Table 1 describes two entries that appear in each node's CRH-FIB.

#### A.1. The SID List Contains One Entry For Each Segment In The Path

In this example, Node S sends a packet to Node D, via I2. In this example, I2 appears in the CRH segment list.

|                                     |                   |
|-------------------------------------|-------------------|
| As the packet travels from S to I2: |                   |
| Source Address = 2001:db8::a        | Segments Left = 1 |
| Destination Address = 2001:db8::2   | SID[0] = 11       |
|                                     | SID[1] = 2        |

Table 2

|                                     |                   |
|-------------------------------------|-------------------|
| As the packet travels from I2 to D: |                   |
| Source Address = 2001:db8::a        | Segments Left = 0 |
| Destination Address = 2001:db8::b   | SID[0] = 11       |
|                                     | SID[1] = 2        |

Table 3

#### A.2. The SID List Omits The First Entry In The Path

In this example, Node S sends a packet to Node D, via I2. In this example, I2 does not appear in the CRH segment list.

|                                     |                   |
|-------------------------------------|-------------------|
| As the packet travels from S to I2: |                   |
| Source Address = 2001:db8::a        | Segments Left = 1 |
| Destination Address = 2001:db8::2   | SID[0] = 11       |

Table 4

|                                     |                   |
|-------------------------------------|-------------------|
| As the packet travels from I2 to D: |                   |
| Source Address = 2001:db8::a        | Segments Left = 0 |
| Destination Address = 2001:db8::b   | SID[0] = 11       |

Table 5

#### Appendix B. A Packet Recycling Use-Case

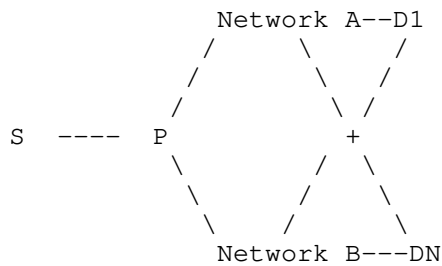


Figure 4: Packet Recycling Use-case

In Figure 4:

- \* The SR domain contains Node S, Node P, and a set of destination nodes (D1 through DN)
- \* S is connected to P
- \* P is connected to Network A and to Network B. Neither of these networks are SR-capable.
- \* The destination nodes connect to both Network A and Network B

S needs to reach each destination node through two SR paths. One SR path traverses Network A while the other traverses Network B.

Uncompressed SRv6 can encode this SR Path in two segments, with one segment instantiated on P and the other on the destination. To support this strategy, P instantiates two END.X SIDs (one per network).

CRH compressed SRv6 can encode this SR Path in two or three segments. When it encodes the path in two segments, one segment instantiated on P and the other on the destination. To support this strategy, P instantiates 2\*N SIDs (one per network per destination). When CRH compressed SRv6 encodes the path in three segments, two segments are instantiated on P and the other on the destination. The first segment on P updates the IPv6 Destination address without forwarding the packet, while the other segment on P forwards the packet without updating the IPv6 destination address. To support this strategy, P instantiates 2+N SIDs (one per network and one per destination).

#### Authors' Addresses

Ron Bonica  
 Juniper Networks  
 2251 Corporate Park Drive



Herndon, Virginia 20171  
United States of America

Email: [rbonica@juniper.net](mailto:rbonica@juniper.net)

Yuji Kamite  
NTT Communications Corporation  
3-4-1 Shibaura, Minato-ku,  
108-8118  
Japan

Email: [y.kamite@ntt.com](mailto:y.kamite@ntt.com)

Andrew Alston  
Liquid Telecom  
Nairobi  
Kenya

Email: [Andrew.Alston@liquidtelecom.com](mailto:Andrew.Alston@liquidtelecom.com)

Daniam Henriques  
Liquid Telecom  
Johannesburg  
South Africa

Email: [daniam.henriques@liquidtelecom.com](mailto:daniam.henriques@liquidtelecom.com)

Luay Jalil  
Verizon  
Richardson, Texas  
United States of America

Email: [luay.jalil@one.verizon.com](mailto:luay.jalil@one.verizon.com)

6man Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: March 19, 2020

R. Bonica  
Juniper Networks  
Y. Kamite  
NTT Communications Corporation  
N. So  
F. Xu  
Reliance Jio  
G. Chen  
Baidu  
Y. Zhu  
G. Yang  
China Telecom  
Y. Zhou  
ByteDance  
September 16, 2019

OAM Capabilities for IPv6  
draft-bonica-6man-oam-04

Abstract

This document defines new IPv6 Operations and Management (OAM) capabilities. In order to support these new capabilities, this document defines an IPv6 OAM Option and an ICMPv6 OAM message.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 19, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                                      |   |
|--------------------------------------|---|
| 1. Problem Statement . . . . .       | 2 |
| 2. Requirements Language . . . . .   | 2 |
| 3. The OAM Option . . . . .          | 2 |
| 3.1. Processing . . . . .            | 4 |
| 4. The ICMPv6 OAM Message . . . . .  | 4 |
| 5. IANA Considerations . . . . .     | 6 |
| 6. Security Considerations . . . . . | 6 |
| 7. Acknowledgements . . . . .        | 6 |
| 8. Normative References . . . . .    | 6 |
| Authors' Addresses . . . . .         | 7 |

## 1. Problem Statement

This document defines new IPv6 [RFC8200] Operations and Management (OAM) capabilities. In order to support these new capabilities, this document defines an IPv6 OAM Option and an ICMPv6 [RFC4443] OAM message.

## 2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 3. The OAM Option

IPv6 source nodes use the OAM option to invoke one or more OAM actions on downstream devices. The OAM option can be included in any of the following:

- o A Hop-by-hop header.
- o A Destination Options header that precedes a Routing header.

- o A Destination Options header that precedes an upper-layer header.

If a Hop-by-hop header includes an OAM option, OAM actions MAY be invoked on every node along the path to the destination, including the destination. If a Destination Options header that precedes a Routing header includes an OAM option, OAM actions are invoked by the first node that appears in the IPv6 Destination Address field plus subsequent nodes listed in the Routing header. If a Destination Options header that precedes an upper-layer header includes an OAM option, OAM actions are invoked on the destination node only.

The OAM option includes the following fields:

- o Option Type (8 bits): OAM. Value TBD by IANA. See Note 1 and Note 2.
- o Opt Data Len (8 bits): Length of Option Data, in bytes. Value MUST be equal to 2.
- o Option Data (16 bits): A bit mask indicating which OAM actions are to be invoked.

| Bit  | Action             | Notes                                                                                                                                                    |
|------|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0    | Log the packet     | The processing node creates a log entry. The log entry reflects the time at which it was created. It also reflects the time at which the packet arrived. |
| 1    | Count the packet   | The processing node increments a counter.                                                                                                                |
| 2    | Send an ICMPv6 OAM | The processing node sends an ICMP OAM message to the packet's source. The OAM message indicates the time at which the packet arrived.                    |
| 3    | Send telemetry     | The processing node sends telemetry to a monitoring station. Telemetry includes the packet and the time at which the packet arrived.                     |
| 4-15 | Reserved           |                                                                                                                                                          |

Table 1: Option Data Bits Mapped to OAM Actions

Table 1 maps Option Data bits to OAM actions.

NOTE 1: As per [RFC8200], the highest-order two bits of the Option Type (i.e., the "act" bits) specify the action taken by a processing node that does not recognize Option Type. The required action is skip over this option and continue processing the header. Therefore, IANA is requested to assign this Option Type with "act" bits "00".

NOTE 2: As per [RFC8200], the third-highest-order bit (i.e., the "chg" bit) of the Option Type specifies whether Option Data can change on route to the packet's destination. Because option data MUST NOT be changed, IANA is requested to assign this Option Type with "chg" bit "0".

### 3.1. Processing

The processing of OAM actions is optional. If a node does not support particular OAM action, it can ignore the corresponding bit in Option Data.

Having processed an OAM option, the processing node should continue to process the packet. If possible, the OAM action should be executed in parallel with the processing of the rest of the packet.

The processing node SHOULD execute the OAM action, even if it can not process the packet further. For example, assume the following:

- o A node receives a packet.
- o The packet contains a Hop-by-hop Options header and the Hop-by-hop Options header includes the OAM option.
- o The node does not maintain a route to the packet's Destination Address

In this case, the node SHOULD execute the requested OAM action. Because the node does not maintain a route to the packet's Destination Address, it should also send an ICMPv6 Destination Unreachable message to the source node and discard the packet.

### 4. The ICMPv6 OAM Message

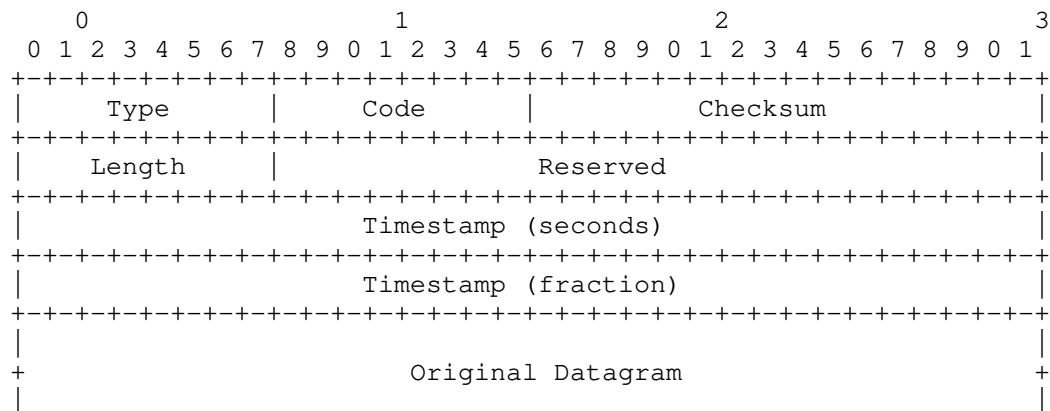


Figure 1

Figure 1 depicts the ICMPv6 OAM message. The ICMPv6 OAM message contains the following fields:

- o Type - OAM. Value TBD by IANA.
- o Code - MUST be set to (0) No Error.
- o Checksum - See [RFC4443]
- o Reserved - MUST be set to 0 and MUST be ignored upon receipt.
- o Length - Represents the length of the padded "original datagram" field, measured in 32-bit words.
- o Timestamp (seconds) - Represents the time at which the original packet arrived in Network Time Protocol (NTP) [RFC5905] format.
- o Timestamp (fraction) - Represents the time at which the original packet arrived in NTP [RFC5905] format.
- o Original Datagram - As much of invoking packet as possible without the ICMPv6 packet exceeding the minimum IPv6 MTU (1280 bytes). The original datagram MUST be zero padded to the nearest 32-bit boundary.

ICMPv6 OAM messages SHOULD be rate limited by the sender.

The Timestamp fields SHOULD be as accurate as possible. They SHOULD reflect the time at which the original packet arrived, not the time at which the ICMPv6 OAM message was sent.

## 5. IANA Considerations

IANA is requested to perform the following actions:

- o Allocate a codepoint from the Destination Options and Hop-by-hop Options registry (<https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#ipv6-parameters-2>). This option is called "OAM". The "act" bits are 00 and the "chg" bit is 0.
- o Create a subregistry in the Destination Options and Hop-by-hop Options registry (<https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#ipv6-parameters-2>). This subregistry is called OAM Option Data Bit Mask. Its contents are defined in Table 1 of this document.
- o Allocate a codepoint from the "ICMPv6 'type' Numbers" registry (<https://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xml>). This type is called "OAM". As it represents an informational message, its value should be greater than 128.
- o Create a "Type x - OAM" subregistry in the "ICMPv6 'type' Numbers" registry (<https://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xml>) registry. This subregistry contains the Code entry (0) No Error.

## 6. Security Considerations

The OAM option can also be used in denial of service attacks. Network devices SHOULD protect themselves against such attacks by limiting the number of OAM options that they process per unit time. If the rate limit is exceeded, the network device MAY either discard the packet or continue to process the packet, ignoring the OAM option.

## 7. Acknowledgements

The authors acknowledge Fred Baker, Shizhang Bi, Ross Callon, Brian Carpenter and Tom Herbert for their helpful comments.

## 8. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.

## Authors' Addresses

Ron Bonica  
Juniper Networks  
Herndon, Virginia 20171  
USA

Email: [rbonica@juniper.net](mailto:rbonica@juniper.net)

Yuji Kamite  
NTT Communications Corporation  
3-4-1 Shibaura, Minato-ku  
Tokyo 108-8118  
Japan

Email: : [y.kamite@ntt.com](mailto:y.kamite@ntt.com)

Ning So  
Reliance Jio  
3010 Gaylord PKWY, Suite 150  
Frisco, Texas 75034  
USA

Email: [Ning.So@ril.com](mailto:Ning.So@ril.com)



Fengman Xu  
Reliance Jio  
3010 Gaylord PKWY, Suite 150  
Frisco, Texas 75034  
USA

Email: Fengman.Xu@ril.com

Gang Chen  
Baidu  
No.10 Xibeiwang East Road Haidian District  
Beijing 100193  
P.R. China

Email: phdgang@gmail.com

Yongqing Zhu  
China Telecom  
109 West Zhongshan Ave, Tianhe District  
Guangzhou  
P.R. China

Email: zhuyq.gd@chinatelecom.cn

Guangming Yang  
China Telecom  
109 West Zhongshan Ave, Tianhe District  
Guangzhou  
P.R. China

Email: yanggm.gd@chinatelecom.cn

Yifeng Zhou  
ByteDance  
Building 1, AVIC Plaza, 43 N 3rd Ring W Rd Haidian District  
Beijing 100000  
P.R. China

Email: yifeng.zhou@bytedance.com

6man  
Internet-Draft  
Intended status: Standards Track  
Expires: April 12, 2021

R. Bonica  
Juniper Networks  
October 9, 2020

The Per-Segment Service Instruction (PSSI) Option  
draft-bonica-6man-seg-end-opt-09

Abstract

This draft has been withdrawn..

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 12, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                                      |   |
|--------------------------------------|---|
| 1. Introduction . . . . .            | 2 |
| 2. Security Considerations . . . . . | 2 |
| 3. IANA Considerations . . . . .     | 2 |
| Author's Address . . . . .           | 2 |

## 1. Introduction

This draft has been withdrawn

## 2. Security Considerations

None.

## 3. IANA Considerations

None.

## Author's Address

Ron Bonica  
Juniper Networks  
2251 Corporate Park Drive  
Herndon, Virginia 20171  
USA

Email: [rbonica@juniper.net](mailto:rbonica@juniper.net)

6man  
Internet-Draft  
Intended status: Standards Track  
Expires: 21 July 2022

R. Bonica  
Juniper Networks  
Y. Kamite  
NTT Communications Corporation  
L. Jalil  
Verizon  
Y. Zhou  
ByteDance  
G. Chen  
Baidu  
17 January 2022

The IPv6 Tunnel Payload Forwarding (TPF) Option  
draft-bonica-6man-vpn-dest-opt-17

Abstract

This document explains how IPv6 options can be used in IPv6 tunnels.  
It also defines the IPv6 Tunnel Payload Forwarding (TPF) option.

Status of This Memo

This Internet-Draft is submitted in full conformance with the  
provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering  
Task Force (IETF). Note that other groups may also distribute  
working documents as Internet-Drafts. The list of current Internet-  
Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months  
and may be updated, replaced, or obsoleted by other documents at any  
time. It is inappropriate to use Internet-Drafts as reference  
material or to cite them other than as "work in progress."

This Internet-Draft will expire on 21 July 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the  
document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal  
Provisions Relating to IETF Documents ([https://trustee.ietf.org/  
license-info](https://trustee.ietf.org/license-info)) in effect on the date of publication of this document.  
Please review these documents carefully, as they describe your rights  
and restrictions with respect to this document. Code Components

extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

|                                                                 |   |
|-----------------------------------------------------------------|---|
| 1. Introduction . . . . .                                       | 2 |
| 2. Requirements Language . . . . .                              | 4 |
| 3. The IPv6 Tunnel Payload Forwarding (TPF) Option . . . . .    | 4 |
| 4. TPF Information Determines Next-Protocol Engine Behavior . . | 5 |
| 5. TPF Information Semantics . . . . .                          | 5 |
| 6. Virtual Private Networking (VPN) Applications . . . . .      | 6 |
| 7. Security Considerations . . . . .                            | 6 |
| 8. IANA Considerations . . . . .                                | 6 |
| 9. Acknowledgements . . . . .                                   | 7 |
| 10. Contributors . . . . .                                      | 7 |
| 11. References . . . . .                                        | 7 |
| 11.1. Normative References . . . . .                            | 7 |
| 11.2. Informative References . . . . .                          | 8 |
| Authors' Addresses . . . . .                                    | 8 |

## 1. Introduction

This document explains how IPv6 options [RFC8200] can be used in IPv6 tunnels. It also defines the IPv6 Tunnel Payload Forwarding (TPF) option.

An IPv6 tunnel [RFC2473] connects two nodes, called the entry-point and the exit-point. The entry-point receives a packet and encapsulates it in a Tunnel IPv6 Header. Figure 1 depicts the encapsulation.

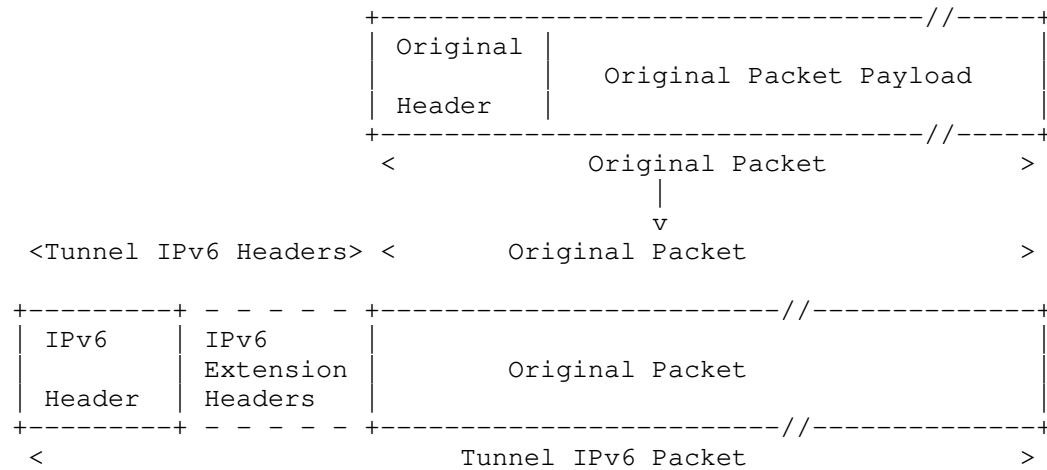


Figure 1: IPv6 Tunnel Encapsulation

The original packet can be any layer-2 or layer-3 packet (e.g., Ethernet, IPv4, IPv6). The Tunnel Header is an IPv6 header followed by zero or more extension headers. The resulting packet is a Tunnel IPv6 Packet.

The entry-point sends the Tunnel IPv6 Packet to the exit-point which then executes the following procedure:

- \* Process the Tunnel IPv6 Header.
- \* Remove the Tunnel IPv6 Header, exposing the original packet.
- \* Submit the original packet to the next-protocol engine.

The exit-point node processes the Tunnel IPv6 Header in strict left-to-right order. It processes the IPv6 header first and then processes extension headers in the order that they appear in the packet. The IPv6 header, and each extension header, includes a Next Header field. The last Next Header field processed identifies the next-protocol engine.

Entry-point nodes can send optional information to the next-protocol engine on the exit-point node. For example, the entry-point can indicate:

- \* The interface through which the next-protocol engine should send the packet.

- \* The routing table that the next-protocol engine should use to process the packet.

To send this information, the entry-point node includes an IPv6 Destination Option header in the Tunnel IPv6 Header. The IPv6 Destination Options header includes an IPv6 TPF option and the IPv6 TPF option includes TPF information. The next-protocol engine on the exit-point node uses TPF information when it forwards the original packet.

## 2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 3. The IPv6 Tunnel Payload Forwarding (TPF) Option

The TPF Option contains the following fields:

- \* Option Type: 8-bit selector. TPF option. Value TBD by IANA. (Suggested value: 0x41). See Note below.
- \* Opt Data Len - 8-bit unsigned integer. Length of the option, in octets, excluding the Option Type and Option Length fields. This field MUST be set to 4.
- \* Option Data - 32-bits. Tunnel Payload Forwarding (TPF) Information.

The TPF option MAY appear in a Destination Options header that precedes an upper-layer header. It MUST NOT appear in a Hop-by-hop Options header or in a Destination Options header that precedes a Routing header.

NOTE : The highest-order two bits of the Option Type (i.e., the "act" bits) are 01. These bits specify the action taken by a destination node that does not recognize the option. The required action is to discard the packet. The third highest-order bit of the Option Type (i.e., the "chg" bit) is 0. This indicates that Option Data cannot be modified along the path between the packet's source and its destination.

#### 4. TPF Information Determines Next-Protocol Engine Behavior

An exit-point node supports one or more next-protocol engines (e.g., Ethernet, IPv4, IPv6). Each next-protocol engine supports a default forwarding procedure and zero or more special forwarding procedures.

When an exit-point node submits a packet to a next-protocol engine without TPF information, the next-protocol engine executes its default forwarding procedure. For example, assume that the exit-point node receives the following Tunnel IPv6 Packet:

- \* The Tunnel IPv6 Packet does not contain TPF information.
- \* The original packet is IPv4.

In this case, the exit-point node processes and removes the Tunnel IPv6 Header. It then submits the original packet, without any TPF information, to the IPv4 protocol engine.

The IPv4 protocol engine executes its default forwarding procedure. It searches its Forwarding Information Base (FIB) for an entry that matches the original packet's destination address. If the search returns a FIB entry, the protocol engine forwards the packet through an interface that the FIB entry identifies.

When an exit-point node submits a packet to a next-protocol engine with TPF information, the next-protocol engine executes a special forwarding procedure. For example, assume that the exit-point node receives the following Tunnel IPv6 packet:

- \* The Tunnel IPv6 Packet contains TPF information that identifies an interface.
- \* The original packet is IPv4.

In this case, the exit-point node processes and removes the Tunnel IPv6 Header. It then submits the original packet, along with TPF information, to the IPv4 protocol engine.

The IPv4 protocol engine executes a special forwarding procedure. It forwards the packet through the interface identified by TPF information, without searching the FIB.

#### 5. TPF Information Semantics

TPF information is opaque. While it must be understood by the entry-point node and the exit-point node, it does not need to be understood by any other node.



## 6. Virtual Private Networking (VPN) Applications

The IPv6 TPF option is useful in deployments where IPv6 tunnels carry:

- \* Layer 3 Virtual Private Network (L3VPN) [RFC4364] traffic.
- \* Ethernet Virtual Private Network (EVPN) [RFC7432] traffic.

When an IPv6 tunnel carries L3VPN traffic, VPN context information can be encoded in an IPv6 TPF option. Therefore, the MPLS service label that is normally present in an L3VPN packet can be eliminated.

When an IPv6 tunnel carries EVPN traffic, VPN context information can be encoded in an IPv6 TPF option. Therefore, the UDP and VXLAN headers that might otherwise be present can be eliminated.

## 7. Security Considerations

TPF information MUST NOT be accepted from untrusted sources. The following are acceptable methods of risk mitigation:

- \* Authenticate the IPv6 TPF option using the IPv6 Authentication Header (AH) [RFC4302] or the IPv6 Encapsulating Security Payload (ESP) Header [RFC4303].
- \* Maintain a secure TPF domain.

All nodes at the edge of a secure TPF domain discard packets that satisfy the following criteria:

- \* Contain an IPv6 TPF option.
- \* Contain an IPv6 Destination Address that represents an interface inside of the secure TPF domain.

## 8. IANA Considerations

IANA is requested to allocate a code point from the Destination Options and Hop-by-hop Options registry (<https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#ipv6-parameters-2>). This option is called "Tunnel Payload Forwarding Option". The "act" bits are 01 and the "chg" bit is 0. The suggested value is 0x41.

## 9. Acknowledgements

Thanks to Dr. Vanessa Ameen, Brian Carpenter, Adrian Farrel, Ishaan Gandhi, Tom Herbert, John Leddy, Srihari Sangli and Tony Li for their comments.

## 10. Contributors

Chris Lenart

Verizon

22001 Loudoun County Parkway

Ashburn, Virginia 20147 USA

Email: [chris.lenart@verizon.com](mailto:chris.lenart@verizon.com)

Greg Presbury

Hughes Network Systems

11717 Exploration Lane

Germantown, Maryland 20876 USA

Email: [greg.presbury@hughes.com](mailto:greg.presbury@hughes.com)

## 11. References

### 11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <<https://www.rfc-editor.org/info/rfc2473>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<https://www.rfc-editor.org/info/rfc4302>>.

- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<https://www.rfc-editor.org/info/rfc4303>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.

## 11.2. Informative References

- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <<https://www.rfc-editor.org/info/rfc7432>>.

## Authors' Addresses

Ron Bonica  
Juniper Networks  
2251 Corporate Park Drive  
Herndon, Virginia 20171  
United States of America

Email: [rbonica@juniper.net](mailto:rbonica@juniper.net)

Yuji Kamite  
NTT Communications Corporation  
3-4-1 Shibaura, Minato-ku,  
108-8118  
Japan

Email: [y.kamite@ntt.com](mailto:y.kamite@ntt.com)

Luay Jalil  
Verizon  
Richardson, Texas  
United States of America

Email: [luay.jalil@one.verizon.com](mailto:luay.jalil@one.verizon.com)

Yifeng Zhou  
ByteDance  
Building 1, AVIC Plaza, 43 N 3rd Ring W Rd Haidian District  
Beijing  
100000  
P.R. China

Email: [yifeng.zhou@bytedance.com](mailto:yifeng.zhou@bytedance.com)

Gang Chen  
Baidu  
No.10 Xibeiwang East Road Haidian District  
Beijing  
100193  
P.R. China

Email: [phdgang@gmail.com](mailto:phdgang@gmail.com)

Network Working Group  
Internet-Draft  
Intended status: Experimental  
Expires: January 6, 2020

R. Hinden  
Check Point Software  
G. Fairhurst  
University of Aberdeen  
July 5, 2019

IPv6 Minimum Path MTU Hop-by-Hop Option  
draft-hinden-6man-mtu-option-02

Abstract

This document specifies a new Hop-by-Hop IPv6 option that is used to record the minimum Path MTU along the forward path between a source to a destination host. This collects a minimum recorded MTU along the path to the destination. The value can then be communicated back to the source using the return Path MTU field in the option.

This Hop-by-Hop option is intended to be used in environments like Data Centers and on paths between Data Centers, to allow them to better take advantage of paths able to support a large Path MTU.

Status of This Memo

This Internet-Draft is submitted to IETF 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 6, 2020.

Copyright Notice

Copyright (c) 2019 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.

## Table of Contents

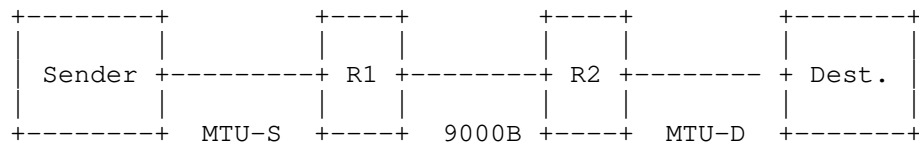
|                                                      |    |
|------------------------------------------------------|----|
| 1. Introduction . . . . .                            | 2  |
| 2. Motivation and Problem Solved . . . . .           | 4  |
| 3. Requirements Language . . . . .                   | 5  |
| 4. Applicability Statements . . . . .                | 5  |
| 5. IPv6 Minimum Path MTU Hop-by-Hop Option . . . . . | 5  |
| 6. Router, Host, and Transport Behaviors . . . . .   | 6  |
| 6.1. Router Behaviour . . . . .                      | 6  |
| 6.2. Host Behavior . . . . .                         | 7  |
| 6.3. Transport Behavior . . . . .                    | 9  |
| 7. IANA Considerations . . . . .                     | 10 |
| 8. Security Considerations . . . . .                 | 11 |
| 9. Acknowledgments . . . . .                         | 11 |
| 10. Change log [RFC Editor: Please remove] . . . . . | 11 |
| 11. References . . . . .                             | 12 |
| 11.1. Normative References . . . . .                 | 12 |
| 11.2. Informative References . . . . .               | 13 |
| Appendix A. Planned Experiments . . . . .            | 13 |
| Authors' Addresses . . . . .                         | 14 |

## 1. Introduction

This draft proposes a new Hop-by-Hop Option to be used to record the minimum MTU along the forward path between the source and destination nodes. The source node creates a packet with this Hop-by-Hop Option and fills the Reported PMTU Field in the option with the value of the MTU for the outbound link that will be used to forward the packet towards the destination.

At each subsequent hop where the option is processed, the router compares the value of the Reported PMTU in the option and the MTU of its outgoing link. If the MTU of the outgoing link is less than the Reported PMTU specified in the option, it rewrites the value in the Option Data with the smaller value. When the packet arrives at the Destination node, the Destination node can send the minimum reported PMTU value back to the Source Node using the Return PMTU field in the option.

The figure below can be used to illustrate the operation of the method. In this case, the path between the Sender and Destination nodes comprises three links, the sender has a link MTU of size MTU-S, the link between routers R1 and R2 has an MTU of size 8 KBytes, and the final link to the destination has an MTU of size MTU-D.



The scenarios are described:

Scenario 1, considers all links to have an 9000 Byte MTU and the method is supported by both routers.

Scenario 2, considers the destination link to have an MTU of 1500 Byte. This is the smallest MTU, router R2 resets the reported PMTU to 1500 Byte and this is detected by the method. Had there been another smaller MTU at a link further along the path that supports the method, the lower PMTU would also have been detected.

Scenario 3, considers the case where the router preceding the smallest link does not support the method, and the method then fails to detect the actual PMTU. These scenarios are summarized in the table below. This scenario would also arise if the PTB message was not delivered to the sender.

|   | MTU-S | MTU-D | R1 | R2 | Rec PMTU | Note                                                                                                    |
|---|-------|-------|----|----|----------|---------------------------------------------------------------------------------------------------------|
| 1 | 9000B | 9000B | H  | H  | 9000 B   | Endpoints attempt to use an 9000 B PMTU.                                                                |
| 2 | 9000B | 1500B | H  | H  | 1500 B   | Endpoints attempt to use a 1500 B PMTU.                                                                 |
| 3 | 9000B | 1500B | H  | -  | 9000 B   | Endpoints attempt to use an 9000 B PMTU, but need to implement a method to fall back use a 1500 B PMTU. |

IPv6 as specified in [RFC8200] allows nodes to optionally process Hop-by-Hop headers. Specifically from Section 4:

- o The Hop-by-Hop Options header is not inserted or deleted, but may be examined or processed by any node along a packet's delivery path, until the packet reaches the node (or each of the set of

nodes, in the case of multicast) identified in the Destination Address field of the IPv6 header. The Hop-by-Hop Options header, when present, must immediately follow the IPv6 header. Its presence is indicated by the value zero in the Next Header field of the IPv6 header.

- o NOTE: While [RFC2460] required that all nodes must examine and process the Hop-by-Hop Options header, it is now expected that nodes along a packet's delivery path only examine and process the Hop-by-Hop Options header if explicitly configured to do so.

The Hop-by-Hop Option defined in this document is designed to take advantage of this property of how Hop-by-Hop options are processed. Nodes that do not support this Option SHOULD ignore them. This can mean that the value returned in the response message does not account for all links along a path.

## 2. Motivation and Problem Solved

The current state of Path MTU Discovery on the Internet is problematic. The problems with the mechanisms defined in [RFC8201] are known to not work well in all environments. Nodes in the middle of the network may not send ICMP Packet Too Big messages or they are rate limited to the point of not making them a useful mechanism.

This results in many connection defaulting to 1280 octets and makes it very difficult to take advantage of links with larger MTU where they exist. Applications that need to send large packets over UDP are forced to use IPv6 Fragmentation.

Transport encapsulations and network-layer tunnels reduce the PMTU available for a transport to use. For example, Network Virtualization Using Generic Routing Encapsulation (NVGRE) [RFC7637] encapsulates L2 packets in an outer IP header and does not allow IP Fragmentation.

The use of 10G Ethernet will not achieve it's potential because the packet per second rate will exceed what most nodes can send to achieve multi-gigabit rates if the packet size limited to 1280 octets. For example, the packet per second rate required to reach wire speed on a 10G Ethernet link with 1280 octet packets is about 977K packets per second (pps), vs. 139K pps for 9,000 octet packets. A significant difference.

The purpose of the this draft is to improve the situation by defining a mechanism that does not rely on nodes in the middle of the network to send ICMPv6 Packet Too Big messages, instead it provides the destination host information on the minimum Path MTU and it can send



this information back to the source host. This is expected to work better than the current RFC8201 based mechanisms.

### 3. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

### 4. Applicability Statements

This Hop-by-Hop Option header is intended to be used in environments such as Data Centers and on paths between Data Centers, to allow them to better take advantage of a path that is able to support a large PMTU. For example, it helps inform a sender that the path includes links that have a MTU of 9,000 Bytes. This has many performance advantages compared to the current practice of limiting packets to 1280 Bytes.

The design of the option is sufficiently simple that it could be executed on a router's fast path. To create critical mass for this to happen will have to be a strong pull from router vendors customers. This could be the case for connections within and between Data Centers.

The method could also be useful in other environments, including the general Internet.

### 5. IPv6 Minimum Path MTU Hop-by-Hop Option

The Minimum Path MTU Hop-by-Hop Option has the following format:

| Option<br>Type | Option<br>Data Len | Option<br>Data          |
|----------------|--------------------|-------------------------|
| BBCTTTTT       | 00000100           | Min-PMTU   Rtn-PMTU   R |

## Option Type:

- BB 00 Skip over this option and continue processing.
- C 1 Option data can change en route to the packet's final destination.
- TTTT 11110 Experimental Option Type from [IANA-HBH].
- Length: 4 The size of the each value field in Option Data field supports Path MTU values from 0 to 65,535 octets.
- Min-PMTU: n 16-bits. The minimum PMTU in octets, reflecting the smallest link MTU that the packet experienced across the path. This is called the Reported PMTU. A value less than the IPv6 minimum link MTU [RFC8200] should be ignored.
- Rtn-PMTU: n 15-bits. The returned minimum PMTU, carrying the 15 most significant bits of the latest received Min-PMTU field. The value zero means that no Reported MTU is being returned.
- R n 1-bit. R-Flag. Set by the source to signal that the destination should include the received Reported PMTU in Rtn-PMTU field.

NOTE: The encoding of the final two octets (Rtn-PMTU and R-Flag) could be implemented by a mask of the latest received Min-MTU value with 0xFFFE, discarding the right-most bit and then performing a logical 'OR' with the R-Flag value of the sender.

## 6. Router, Host, and Transport Behaviors

## 6.1. Router Behaviour

Routers that do not support Hop-by-Hop options SHOULD ignore this option and SHOULD forward the packet.

Routers that support Hop-by-Hop Options, but do not recognize this option SHOULD ignore the option and SHOULD forward the packet.

Routers that recognize this option SHOULD compare the Reported PMTU in the Min-PMTU field and the MTU configured for the outgoing link. If the MTU of the outgoing link is less than the Reported PMTU, the router rewrites the Reported PMTU in the Option to use the smaller value.

The router MUST ignore and not change the Rtn-PMTU field and R-Flag in the option.

Discussion:

- o The design of this Hop-by-Hop Option makes it feasible to be implemented within the fast path of a router, because the required processing is simple.

## 6.2. Host Behavior

The source host that supports this option SHOULD create a packet with this Hop-by-Hop Option and fill the Min-PMTU field of the option with the MTU of configured for the link over which it will send the packet on the next hop towards the destination.

The source host may request that the destination host return the received minimum MTU value by setting the R-Flag in the option. This will cause the destination host to include a PMTU option in an outgoing packet.

Discussion:

- o This option does not need to be sent in all packets belonging to a flow. A transport protocol (or packetization layer) can set this option only on specific packets used to test the path.
- o In the case of TCP, the option could be included in packets carrying a SYN segment as part of the connection set up, or can periodically be sent in packets carrying other segments. Including this packet in a SYN could increase the probability that SYN segment is lost, when routers on the path drop packets with this option. Including this option in a large packet is not likely to be useful, since the large packet might itself also be dropped by a link along the path with a smaller MTU, preventing the Reported PMTU information from reaching the Destination node.
- o The use with datagram transport protocols (e.g. UDP) is harder to characterize because applications using datagram transports range from very short-lived (low data-volume applications) exchanges, to longer (bulk) exchanges of packets between the Source and Destination nodes [RFC8085].

- o For applications that use Anycast, this option should be included in all packets as the actual destination will vary due to the nature of Anycast.
- o Simple-exchange protocols (i.e low data-volume applications [RFC8085] that only send one or a few packets per transaction, could be optimized by assuming that the Path MTU is symmetrical, that is where the Path MTU is the same in both directions, or at least not smaller in the return path. This optimisation does not hold when the paths are not symmetric.
- o The use of this option with DNS and DNSSEC over UDP ought to work as long as the paths are symmetric. The DNS server will learn the Path MTU from the DNS query messages. If the return Path MTU is smaller, then the large DNSSEC response may be dropped and the known problems with PMTUD will occur. DNS and DNSSEC over transport protocols that can carry the Path MTU should work.

The Source Host can request the destination host to send a packet carrying the PMTU Option using the R-Flag.

A Destination Host SHOULD respond to each packet received with the R-Flag set, by setting the PMTU Option in the next packet that it sends to the Source Host by the same upper layer protocol instance.

The upper layer protocol MAY generate a packet when any of these conditions is met when the R Flag is set in the PMTU Option and either:

- o It is the first Reported PMTU value it has received from the Source.
- o The Reported PMTU value is lower than previously received.

The R-Flag SHOULD NOT be set when the PMTU Option was sent solely to carry the feedback of a Reported PMTU.

The PMTU Option sent back to the source SHOULD contain the outgoing link MTU in Min-PMTU field and SHOULD set the last Received PMTU in the Rtn-PMTU field. If these values are not present the field MUST be set to zero.

For a connection-oriented upper layer protocol, this could be implemented by saving the value of the last received option within the connection context. This last received value is then used to set the return Path MTU field for all packets belonging to this flow that carry the IPv6 Minimum Path MTU Hop-by-Hop Option.

A connection-less protocol, e.g., based on UDP, requires the application to be updated to cache the Received PMTU value, and to ensure that this corresponding value is used to set the last Received PMTU in the Rtn-PMTU field of any PMTU Option that it sends.

NOTE: The Rtn-PMTU value is specific to the instance of the upper layer protocol (i.e. matching the IPv6 flow ID, port-fields in UDP or the SPI in IPsec, etc), not the protocol itself, because network devices can make forwarding decisions that impact the PMTU based on the presence and values of these upper layer fields, and therefore these fields need to correspond to those of the packets for the flow received by the Destination Host set to ensure feedback is provided to the corresponding Source Host.

NOTE: An upper layer protocol that send packets from the Destination Host towards the Source Host less frequently than the Destination Host receives packets from the Source Host, provides less frequent feedback of the received Min-PMTU value. However, it will always needs to send the most recent value.

#### Discussion:

- o A simple mechanism could only send an MTU Option with the Rtn-PMTU field filled in the first time this option is received or when the Received PMTU is reduced. This is good because it limits the number sent, but there is no provision for retransmission of the PMTU Option fails to reach the sender, or the sender loses state.
- o The Reported PMTU value could increase or decrease over time. For instance, it would increase when the path changes and the packets become then forwarded over a link with a MTU larger than the link previously used.

### 6.3. Transport Behavior

A transport endpoint using this option needs to use a method to verify the information provided by this option.

The Received PMTU does not necessarily reflect the actual PMTU between the sender and destination. Care therefore needs to be exercised in using this value at the sender. Specifically:

- o If the Received PMTU value returned by the Destination is the same as the initial Reported PMTU value, there could still be a router or layer 2 device on the path that does not support this PMTU. The usable PMTU therefore needs to be confirmed.

- o If the Received PMTU value returned by the Destination is smaller than the initial Reported PMTU value, this is an indication that there is at least one router in the path with a smaller MTU. There could still be another router or layer 2 device on the path that does not support this MTU.
- o If the Received PMTU value returned by the Destination is larger than the initial Reported PMTU value, this may be a corrupted, delayed or mis-ordered response, and SHOULD be ignored.

A sender needs to discriminate between the Received PMTU value in a PTB message generated in response to a Hop-by-Hop option requesting this, and a PTB message received from a router on the path.

A PMTUD or PLPMTUD method could use the Received PMTU value as an initial target size to probe the path. This can significantly decrease the number of probe attempts (and hence time taken) to arrive at a workable PMTU. It has the potential to complete discovery of the correct value in a single Round Trip Time (RTT), even over paths that may have successive links configured with lower MTUs.

Since the method can delay notification of an increase in the actual PMTU, a sender with a link MTU larger than the current PMTU SHOULD periodically probe for a PMTU value that is larger than the Received PMTU value. This specification does not define an interval for the time between probes.

Since the option consumes less capacity than a full probe packet, there may be advantage in using this to detect a change in the path characteristics.

NOTE: Further details to be included in next version.

NOTE: A future version of the document will consider more the impact of Equal Cost Multipath (ECMP). Specifically, whether a Received PMTU value should be maintained by the method for each transport endpoint, or for each network address, and how these are best used by methods such as PLPMTUD or DPLPMTUD.

## 7. IANA Considerations

No IANA assignments are requested. Document uses experimental option from [IANA-HBH].

## 8. Security Considerations

The method has no way to protect the destination from off-path attack using this option in packets that do not originate from the source. This attack could be used to inflate or reduce the size of the reported PMTU. Mechanisms to provide this protection can be provided at a higher layer (e.g., the transport packetization layer using PLPMTUD or DPLPMTUD), where more information is available about the size of packet that has successfully traversed a path.

The method solicits a response from the destination, which should be used to generate a response to the IPv6 node originating the option packet. A malicious attacker could generate a packet to the destination for a previously inactive flow or one that advertises a change in the size of the MTU for an active flow. This would create additional work at the destination, and could induce creation of state when a new flow is created. It could potentially result in additional traffic on the return path to the sender, which could be mitigated by limiting the rate at which responses are generated.

A sender **MUST** check the quoted packet within the PTB message to validate that the message is in response to a packet that was originated by the sender. This is intended to provide protection against off-path insertion of ICMP PTB messages by an attacker trying to disrupt the service. Messages that fail this check **MAY** be logged, but the information they contain **MUST** be discarded.

TBD

## 9. Acknowledgments

A somewhat similar mechanism was proposed for IPv4 in 1988 in [RFC1063] by Jeff Mogul, C. Kent, Craig Partridge, and Keith McCloghrie. It was later obsoleted in 1990 by [RFC1191] the current deployed approach to Path MTU Discovery.

Helpful comments were received from Tom Herbert, Tom Jones, Fred Templin, Ole Troan, [Your name here], and other members of the 6MAN working group.

## 10. Change log [RFC Editor: Please remove]

draft-hinden-6man-mtu-option-02, 2019-July-5

- o Changed option format to also include the Returned MTU value and Return flag and made related text changes in Section 6.2 to describe this behaviour.

- o ICMP Packet Too Big messages are no longer used for feedback to the Source host.
- o Added to Acknowledgements Section that a similar mechanism was proposed for IPv4 in 1988 in [RFC1063].
- o Editorial changes.

draft-hinden-6man-mtu-option-01, 2019-March-05

- o Changed requested status from Standards Track to Experimental to allow use of experimental option type (11110) to allow for experimentation. Removed request for IANA Option assignment.
- o Added Section 2 "Motivation and Problem Solved" section to better describe what the purpose of this document is.
- o Added Appendix A describing planned experiments and how the results will be measured.
- o Editorial changes.

draft-hinden-6man-mtu-option-00, 2018-Oct-16

- o Initial draft.

## 11. References

### 11.1. Normative References

[IANA-HBH]

"Destination Options and Hop-by-Hop Options",  
<<https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#ipv6-parameters-2>>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.

[RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017, <<https://www.rfc-editor.org/info/rfc8201>>.



## 11.2. Informative References

- [RFC1063] Mogul, J., Kent, C., Partridge, C., and K. McCloghrie, "IP MTU discovery options", RFC 1063, DOI 10.17487/RFC1063, July 1988, <<https://www.rfc-editor.org/info/rfc1063>>.
- [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <<https://www.rfc-editor.org/info/rfc1191>>.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<https://www.rfc-editor.org/info/rfc2460>>.
- [RFC7637] Garg, P., Ed. and Y. Wang, Ed., "NVGRE: Network Virtualization Using Generic Routing Encapsulation", RFC 7637, DOI 10.17487/RFC7637, September 2015, <<https://www.rfc-editor.org/info/rfc7637>>.
- [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <<https://www.rfc-editor.org/info/rfc8085>>.

## Appendix A. Planned Experiments

TBD

This section will describe a set of experiments planned for the use of the option defined in this document. There are many aspects of the design that require experimental data or experience to evaluate this experimental specification.

This includes experiments to understand the pathology of packets sent with the specified option to determine the likelihood that they are lost within specific types of network segment.

This includes consideration of the cost and alternatives for providing the feedback required by the mechanism and how to effectively limit the rate of transmission.

This includes consideration of the potential for integration in frameworks such as that offered by DPLPMTUD.

There are also security-related topics to be understood as described in the Security Considerations (Section 8).

Authors' Addresses

Robert M. Hinden  
Check Point Software  
959 Skyway Road  
San Carlos, CA 94070  
USA

Email: bob.hinden@gmail.com

Godred Fairhurst  
University of Aberdeen  
School of Engineering  
Fraser Noble Building  
Aberdeen AB24 3UE  
UK

Email: gorry@erg.abdn.ac.uk

Network Working Group  
Internet-Draft  
Updates: 4861, 5175 (if approved)  
Intended status: Standards Track  
Expires: September 8, 2019

R. Hinden  
Check Point Software  
B. Carpenter  
Univ. of Auckland  
B. Zeeb  
March 7, 2019

IPv6 Router Advertisement IPv6-Only Flag  
draft-ietf-6man-ipv6only-flag-05

Abstract

This document specifies a Router Advertisement Flag to indicate to hosts that the administrator has configured the router to advertise that the link is IPv6-Only. This document updates RFC4861 and RFC5175.

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 September 8, 2019.

Copyright Notice

Copyright (c) 2019 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 Language . . . . .                               | 4  |
| 3. Applicability Statements . . . . .                            | 4  |
| 4. IPv6-Only Definition . . . . .                                | 5  |
| 5. IPv6-Only Flag . . . . .                                      | 5  |
| 6. Router and Operational Considerations . . . . .               | 6  |
| 7. Host Behavior Considerations . . . . .                        | 7  |
| 8. IANA Considerations . . . . .                                 | 8  |
| 9. Security Considerations . . . . .                             | 8  |
| 10. Acknowledgments . . . . .                                    | 9  |
| 11. Change log [RFC Editor: Please remove] . . . . .             | 9  |
| 12. References . . . . .                                         | 12 |
| 12.1. Normative References . . . . .                             | 12 |
| 12.2. Informative References . . . . .                           | 13 |
| Appendix A. Implementaton Status [RFC Editor: Please remove] . . | 14 |
| A.1. FreeBSD Implementation . . . . .                            | 14 |
| A.2. Test using Scapy . . . . .                                  | 14 |
| Authors' Addresses . . . . .                                     | 15 |

## 1. Introduction

This document specifies a Router Advertisement Flag to indicate to hosts that the administrator has configured the router to advertise that the link is IPv6-Only. The flag only applies to IPv6 default routers.

Hosts that support IPv4 and IPv6, usually called dual stack hosts, need to also work efficiently on IPv6-Only links, i.e, links where there are no IPv4 routers and/or IPv4 services. Dual stack is the default configuration for most current host operating systems such as Windows 10, iOS, Android, Linux, and BSD, as well as devices such as some printers. Monitoring of an IPv6-Only link, for example at the IETF 100 meeting in Singapore, shows that current dual stack hosts will create local auto-configured IPv4 addresses and attempt to reach IPv4 services, even though they cannot configure a normal address using DHCP. This may be a problem for several reasons, depending on the equipment in use and its configuration, especially on large wireless networks:

- o It may result in an undesirable level of wasted Layer 2 broadcast traffic.

- o Switches in multi-segment wireless networks may create IPv4 state for dual stack hosts (in particular, ARP cache entries to support ARP proxying).
- o Such traffic may drain battery power on wireless hosts that have no interest in link-local IPv4, ARP, and DHCPv4 relay traffic, but receive unwanted IPv4 packets. [RFC7772] indicates how this risk might be quantified.
- o Similarly, hosts may waste battery power on futile attempts to access services by sending IPv4 packets.
- o On an IPv6-Only link, IPv4 might be used for malicious purposes and pass unnoticed by IPv6-Only monitoring mechanisms.

In networks with managed infrastructure whose equipment allows it, these problems could be mitigated by configuring the Layer 2 infrastructure to drop IPv4 and ARP traffic by filtering Ethertypes 0x0800 and 0x0806 [IANA-Ethertype]. IPv6 uses a different EtherType, 0x86DD, so this filtering will not interfere with IPv6 traffic. Depending on the equipment details, this would limit the traffic to the link from an IPv4 sender to the switch, and would drop all IPv4 and ARP broadcast packets at the switch. This document recommends using such mechanisms when available.

However, hosts transmitting IPv4 packets would still do so, consuming their own battery power and some radio bandwidth. The intent of this specification is to provide a mechanism that prevents such traffic, and also works on networks without the ability to filter L2 traffic, or where there are portions of a network without the ability to filter L2 traffic. It may also be valuable on unmanaged networks using routers pre-configured for IPv6-Only operations and where Layer 2 filtering is unavailable.

An assumption of this document is that because it is an IPv6-Only link there is no IPv4 DHCP server or relay active on the link. This further means that the DHCP option to disable IPv4 stateless auto-configuration [RFC2563] can not be used.

The remainder of this document therefore assumes that neither effective Layer 2 filtering nor the RFC 2563 DHCP option is applicable to the link concerned.

Because there is no IPv4 support on an IPv6-Only link, the only way to notify the dual stack hosts that this link is IPv6-Only is to use an IPv6 mechanism. An active notification will be much more precise than attempting to deduce this fact by the lack of IPv4 responses or traffic.

This document therefore defines a mechanism that a router administrator can use to inform hosts that this is an IPv6-Only link on their default routers such that they can disable IPv4 on this link, mitigating all of the above problems. The mechanism is based on the IPv6 Router Advertisement message because this is a type of message that is certain to be received by every dual stack host, regardless of what network management protocols may or may not be in use.

IPv4-only hosts, and dual-stack hosts that do not recognize the new flag, may continue to attempt IPv4 operations, in particular IPv4 discovery protocols typically sent as link-layer broadcasts. This legacy traffic cannot be prevented by any IPv6 mechanism. The value of the new flag is limited to hosts that recognize it.

A possible subsidiary use of the IPv6-Only flag is using it to trigger IPv6-Only testing and validation on a link.

This document specifies a new flag for Router Advertisement Flag [RFC5175]. It updates [RFC5175] to add this flag. It also updates [RFC4861] to add an additional item to check and report.

## 2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 3. Applicability Statements

This OPTIONAL mechanism is designed to allow administrators to notify hosts that the link is IPv6-Only. It SHOULD be only used in IPv6-Only links (see Section 4 for definition of IPv6-Only). For a VLAN, the IPv6-Only flag only applies to the specific VLAN on which it was received.

Dual stack hosts that have IPv4 active configuration obtained from the network (e.g., via DHCP), can ignore the flag and continue to use IPv4.

Administrators MUST only use this mechanism if they are certain that the link is IPv6-Only. For example, in cases where there is a need to continue to use IPv4, when there are intended to be IPv4-only hosts or IPv4 routers on the link, setting this flag to 1 is a configuration error.

This mechanism is intended to be compatible with link-layer solutions that filter out IPv4 traffic.

#### 4. IPv6-Only Definition

IPv6-Only is defined to mean that no other versions of Internet Protocol than IPv6 are intentionally in use directly on the link. Today this effectively simply means that IPv4 is not intentionally in use on the link, and it includes:

- \* No IPv4 traffic on the link.
- \* No IPv4 routers on the link.
- \* No DHCPv4 servers on the link.
- \* No IPv4 accessible services on the link.
- \* All IPv4 and ARP traffic may be blocked at Layer 2 by the administrator.

It is expected that on IPv6-Only networks it will be common to access to IPv4 external services by techniques such as NAT64 [RFC6146] and DNS64 [RFC6147] at the edge of the network. This is beyond the scope of this document.

Note that IPv6-Only provides no information about other network protocols than IP (and ARP) in use directly over the link layer. It is out of scope of this specification whether any such protocol is in use on the link or whether any protocol is tunneled over IPv6.

#### 5. IPv6-Only Flag

RFC5175 currently defines the flags in the NDP Router Advertisement message and these flags are registered in the IANA IPv6 ND Router Advertisement flags Registry [IANA-RF]. This currently contains the following one-bit flags defined in published RFCs:

```

 0 1 2 3 4 5 6 7
+---+---+---+---+---+---+
|M|O|H|Prf|P|R|R|
+---+---+---+---+---+---+

```

|     |                                              |
|-----|----------------------------------------------|
| M   | Managed Address Configuration Flag [RFC4861] |
| O   | Other Configuration Flag [RFC4861]           |
| H   | Mobile IPv6 Home Agent Flag [RFC3775]        |
| Prf | Router Selection Preferences [RFC4191]       |
| P   | Neighbor Discovery Proxy Flag [RFC4389]      |

R     Reserved

This document defines bit 6 to be the IPv6-Only Flag:

S     IPv6-Only Flag

This flag has two values. These are:

0     This is not an IPv6-Only link  
1     This is an IPv6-Only link

RFC 5175 requires that unused flag bits be set to zero. Therefore, a router that does not support the new flag will not appear to assert that this is an IPv6-Only link.

Hosts receiving the Router Advertisement SHOULD only process this flag if the advertising router is a Default Router. Specifically, if the Lifetime field in the Router Advertisement is not zero, otherwise it SHOULD be ignored. This is done to allow some IPv6 routers to advertise information without being a Default Router and providing IPv6 connectivity.

Note that although this mechanism uses one of only two reserved flag bits in the RA, an extension mechanism is defined in Section 4 of [RFC5175] in case additional flags are ever required for future extensions. It should be noted that since RFC5175 was published in 2008, no new RA flags have been assigned in the IANA registry.

## 6. Router and Operational Considerations

Default IPv6 routers that are on an IPv6-Only link SHOULD be configured by the administrator to set the IPv6-Only flag to 1 on interfaces on this link. In all other cases the flag SHOULD NOT be set to 1.

The intent is that the administrator of the router configures the router to set the IPv6-Only flag if and only if she/he wants to tell the hosts on the link that the link is IPv6-Only. This is a configuration flag, it is not something that the router decides on its own. Routers MAY log a configuration error if the flag is set and IPv4 is still active on the router's interface to the link.



Routers implementing this document SHOULD log to system or network management inconsistent setting of the IPv6-Only flag. This extends the behaviour specified in Section 6.2.7 of [RFC4861].

Operators of large IPv6-Only wireless links are advised to also use Layer 2 techniques to drop IPv4 and ARP packets (Ethertypes 0x0800 and 0x0806) at all switches, and to ensure that IPv4 and ARP features are disabled in all switches.

## 7. Host Behavior Considerations

Hosts that support the IPv6-Only RA flag MUST have a configuration option to ignore or process the flag. The motivation for this configuration option is for hosts that are capable of processing the IPv6-Only flag to only act on the flag if they are configured to do so.

If there are multiple IPv6 default routers on a link, they might send different values of the flag. If at least one IPv6 default router sends the flag with value 0, a dual stack host MUST NOT assume that the link is IPv6-Only. If all IPv6 default routers send the flag with value 1, a dual stack host SHOULD assume that this is an IPv6-Only link.

A host that receives only RAs with the flag set to 1 SHOULD NOT attempt any IPv4 operations, unless it subsequently receives at least one RA with the flag set to zero. As soon as such an RA is received, IPv4 operations MAY be started.

If the host has active IPv4 configuration information obtained from the network (e.g., via DHCP), the flag can be ignored and IPv4 operations can continue. The host MAY implement a policy overriding these default behaviors.

In the event that the host subsequently receives at least one RA with the flag set to zero IPv4 operations MAY be started.

A host MAY delay all IPv4 operations at start-up or reconnection until a reasonable time has elapsed for RA messages to arrive. If all RAs received have the flag set to 1, a host SHOULD NOT attempt IPv4 operations.

In all of the above, the flag's value is considered valid for the lifetime of the default router concerned, unless a subsequent RA delivers a different flag value. If a default router expires (i.e., no RA is received that refreshes its lifetime), the host must remove this router's flag value from consideration. If the result is that all surviving default routers have the flag set to 1, the host SHOULD

assume that the link is IPv6-Only. In other words, at any given time, the state of the flag as seen by the host is the logical AND of the flags sent by all unexpired default IPv6 routers on the link.

This also means that if all default routers on the link have set the flag, the resulting host state for the link is IPv6-Only. If the lifetimes of all the routers on the link subsequently expire, then the host state for the link is not IPv6-Only.

## 8. IANA Considerations

IANA is requested to assign the new Router Advertisement flag defined in Section 5 of this document. Bit 6 is the next available bit in this registry, IANA is requested to use this bit unless there is a reason to use another bit in this registry.

IANA is also requested to register this new flag bit in the IANA IPv6 ND Router Advertisement flags Registry [IANA-RF].

## 9. Security Considerations

This document shares the security issues with other parts of IPv6 Neighbor Discovery. [RFC6104] discusses certain attacks and mitigations. General techniques to protect Router Advertisement traffic such as Router Guard [RFC6105] are useful in protecting against these vulnerabilities.

A bad actor could use this mechanism to attempt turn off IPv4 service on a link that is intentionally using IPv4, by sending Router Advertisements with the IPv6-Only flag set to 1. There are several protections to reduce this attack. These are:

- o There is configuration setting that controls if the host should process the IPv6-Only flag. This gives local control over using the flag and reduces the ability of a bad actor to turn off IPv4 for hosts that support the flag.
- o As long as there are one or more routers sending Router Advertisements with this flag set to 0, they would override this attack given the mechanism in Section 5. Specifically a host would only turn off IPv4 service if it wasn't hearing any Router Advertisement with the flag set to 0. If the advice in Section 6 is followed, this attack will fail.
- o An attack would not succeed if the dual stack hosts had active IPv4 configuration. As specified in Section 7, a dual stack host will ignore the flag if it has active IPv4 configuration.

In a situation where the bad actor has control of all routers on the link and sends Router Advertisements with the IPv6-Only flag set to 1 from all of them and if the hosts don't have assigned IPv4 addresses, the attack will succeed, but so will many other forms of router-based attack.

Conversely, a bad actor could use this mechanism to turn on, or pretend to turn on, IPv4 service on an IPv6-Only link, by sending Router Advertisements with the flag set to 0. However, this is really no different than what such a bad actor can do anyway, if they have the ability to configure a bogus router in the first place. The advice in Section 6 will minimize such an attack by limiting it to a single link.

Note that manipulating the Router Preference [RFC4191] will not affect either of these attacks: any IPv6-Only flag of 0 will always override all flags set to 1.

The new flag is neutral from an IPv6 privacy viewpoint, since it does not affect IPv6 operations in any way. From an IPv4 privacy viewpoint, it has the potential benefit of suppressing unnecessary traffic that might reveal the existence of a host and the correlation between its hardware and IPv4 addresses. It should be noted that hosts that don't support this flag are not protected from IPv4-based attacks.

## 10. Acknowledgments

A closely related proposal was published earlier as [I-D.ietf-sunset4-noipv4].

Helpful comments were received from Lorenzo Colitti, David Farmer, Fernando Gont, Nick Hilliard, Lee Howard, Erik Kline, Jen Linkova, Veronika McKillop, George Michaelson, Alexandre Petrescu, Michael Richardson, Mark Smith, Barbara Stark, Tatuya Jinmei, Ole Troan, James Woodyatt, and other members of the 6MAN working group.

Bjoern Zeeb has also produced a variant of this proposal and proposed an IPv6 transition plan in [I-D.bz-v4goawayflag].

## 11. Change log [RFC Editor: Please remove]

draft-ietf-6man-ipv6only-flag-05, 2019-March-7:

- \* Added a host configuration option to Section 7 that controls if the host should process the IPv6-Only flag. This provides local control over using the use of flag and reduces the

ability of a bad actor to turn off IPv4 for hosts that support the flag.

- \* Changed Section 7 to specify that the host can ignore flag set to 1 if it has active IPv4 configuration obtained from the network (e.g., via DHCP). Similar changes to Section 3 and Section 9
- \* Clarification in Section 6 to strengthen the text about the administrators intent.
- \* Added Bjoern Zeeb as an author.
- \* Updated information on FreeBSD implementation in Appendix A.1
- \* Editorial changes.

draft-ietf-6man-ipv6only-flag-04, 2018-November-4:

- \* Added text to Section 1 explaining why the mechanism is based on Router Advertisements.
- \* Added text to Section 3 that for a VLAN, the IPv6-Only flag only applies to the specific VLAN on which it was received.
- \* Changed Section 3 that administrators MUST only use this mechanism if they are certain that the link is IPv6-Only, instead of SHOULD.
- \* Added ARP to Section 4 protocols that the IPv6-Only flag applies to.
- \* Renamed the IPv6-Only flag label from "6" to "S".
- \* Added pointers to Section 7.2.7 of RFC4861 in Section 6.
- \* Added that RFC4861 is also updated by Section 6 for routers implementing this flag.
- \* Changed Section 7 from SHOULD NOT to MUST NOT.
- \* Added Appendix A on implementations and testing.
- \* Many small clarifications based on IPv6 list discussion and editorial changes.

draft-ietf-6man-ipv6only-flag-03, 2018-October-16:

- \* Reorganized text about problem statement and applicability
- \* Added note about shortage of flag bits
- \* Clarified text about logging configuration error in Section 6
- \* Editorial changes.

draft-ietf-6man-ipv6only-flag-02, 2018-August-14:

- \* Added text to Section 9 to clarify that hosts not supporting this flag are not protected from IPv4-based attacks.
- \* Editorial changes.

draft-ietf-6man-ipv6only-flag-01, 2018-June-29:

- \* Added text to section that defines what IPv6-Only includes to clarify that only other version of the Internet Protocol are in scope.
- \* Added clarification if the lifetime of all routers expire.
- \* Editorial changes.

draft-ietf-6man-ipv6only-flag-00, 2018-May-21:

- \* Changed the file name to draft-ietf-6man-ipv6only-flag to match the current title and that it is a w.g. draft.
- \* Added new section that defines what IPv6-Only includes.
- \* Expanded description of using Layer 2 filter to block IPv4 and ARP traffic.
- \* Editorial changes.

draft-hinden-ipv4flag-04, 2018-April-16:

- \* Changed the name of the document and flag to be the IPv6-Only flag.
- \* Rewrote text to make it affirmative that this is used by an administrator to tell the hosts that the link is IPv6-Only.
- \* Added an Applicability Statements section to scope the intended use.
- \* Changed requirement language to upper case, added Requirements Language section with references to [RFC2119] and [RFC8174].
- \* Editorial changes.

draft-hinden-ipv4flag-03, 2018-Feb-15:

- \* Changed terminology to use "link" instead of "network".
- \* Improved text in Section 4. "Host Behavior Considerations" and added suggestion to only perform IPv4 if an application requests it.
- \* Added clarification that the bit is set because an administrator configured the router to send it.
- \* Editorial changes.

draft-hinden-ipv4flag-02, 2018-Feb-15:

- \* Improved text in introduction.
- \* Added reference to current IANA registry in Section 2.

- \* Editorial changes.

draft-hinden-ipv4flag-01, 2017-Dec-12

- \* Inverted name of flag from "Available" to "Unavailable".
- \* Added problem description and clarified scope.
- \* Added router and operational considerations.
- \* Added host behavior considerations.
- \* Extended security considerations.
- \* Added Acknowledgment section, including reference to prior sunset4 draft.

draft-hinden-ipv4flag-00, 2017-Nov-17:

- \* Original version.

## 12. References

### 12.1. Normative References

- [IANA-Ethertype] "Ether Types", <<https://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xhtml#ieee-802-numbers-1>>.
- [IANA-RF] "IPv6 ND Router Advertisement flags", <<https://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xhtml#icmpv6-parameters-11>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <<https://www.rfc-editor.org/info/rfc4191>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [RFC5175] Haberman, B., Ed. and R. Hinden, "IPv6 Router Advertisement Flags Option", RFC 5175, DOI 10.17487/RFC5175, March 2008, <<https://www.rfc-editor.org/info/rfc5175>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

## 12.2. Informative References

- [I-D.bz-v4goawayflag]  
Zeeb, B., "IPv6 Router Advertisement IPv4 GoAway Flag", draft-bz-v4goawayflag-00 (work in progress), March 2018.
- [I-D.ietf-sunset4-noipv4]  
Perreault, S., George, W., Tsou, T., Yang, T., and J. Tremblay, "Turning off IPv4 Using DHCPv6 or Router Advertisements", draft-ietf-sunset4-noipv4-01 (work in progress), December 2014.
- [RFC2563] Troll, R., "DHCP Option to Disable Stateless Auto-Configuration in IPv4 Clients", RFC 2563, DOI 10.17487/RFC2563, May 1999, <<https://www.rfc-editor.org/info/rfc2563>>.
- [RFC6104] Chown, T. and S. Venaas, "Rogue IPv6 Router Advertisement Problem Statement", RFC 6104, DOI 10.17487/RFC6104, February 2011, <<https://www.rfc-editor.org/info/rfc6104>>.
- [RFC6105] Levy-Abegnoli, E., Van de Velde, G., Popoviciu, C., and J. Mohacsi, "IPv6 Router Advertisement Guard", RFC 6105, DOI 10.17487/RFC6105, February 2011, <<https://www.rfc-editor.org/info/rfc6105>>.
- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <<https://www.rfc-editor.org/info/rfc6146>>.
- [RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, DOI 10.17487/RFC6147, April 2011, <<https://www.rfc-editor.org/info/rfc6147>>.
- [RFC7772] Yourtchenko, A. and L. Colitti, "Reducing Energy Consumption of Router Advertisements", BCP 202, RFC 7772, DOI 10.17487/RFC7772, February 2016, <<https://www.rfc-editor.org/info/rfc7772>>.

[Scapy\_RA]

"Router Advertisements with scapy (NETLAB)",  
<<https://samsclass.info/124/proj11/proj9xN-scapy-ra.html>>.

#### Appendix A. Implementaton Status [RFC Editor: Please remove]

At the time this document was written there is one implementation and a few comparability tests.

##### A.1. FreeBSD Implementation

A FreeBSD implementation was written by Bjoern A. Zeeb.

###### Summary:

Changes for the IPv6-Only flag include updates of user space utilities to announce the "S" (IPv6-Only) flag to the network and to show it in management utilities.

The kernel logic includes a global flag to disable processing of the IPv6-Only flag even if the logic to act upon the IPv6-Only flag is compiled in. There are checks for IPv4 configuration on a receiving interface, which if found, will also force the IPv6-Only flag to be ignored.

Further there are input and output filters for IPv4, ARP, and REVARP in place for when the flag passes the aforementioned checks and is enabled.

In addition to the draft there is a manual option to enable the IPv6-Only filtering logic manually to observe the system behaviour on links without router(s) advertising the IPv6-Only flag.

The code was tested with 2 FreeBSD IPv6 routers, a FreeBSD laptop on ethernet as well as wifi, and with Win10 and OSX clients (which did not fall over with the "S" flag set but not understood).

More information and updates can be found at:

<https://wiki.freebsd.org/IPv6/IPv6OnlyRAFlag>

##### A.2. Test using Scapy

Independent tests have been done using [Scapy\_RA] by Alexandre Petrescu and Brian Carpenter to verify that setting the IPv6-Only



Flag did not break legacy hosts. Both verified that setting this flag did not cause any adverse effects on Windows 10 and Android.

Authors' Addresses

Robert M. Hinden  
Check Point Software  
959 Skyway Road  
San Carlos, CA 94070  
USA

Email: bob.hinden@gmail.com

Brian Carpenter  
Department of Computer Science  
University of Auckland  
PB 92019  
Auckland 1142  
New Zealand

Email: brian.e.carpenter@gmail.com

Bjoern A. Zeeb  
Bruehlstr. 19  
Bondorf 71149  
Germany

Email: bzeeb+ietf@zabbadoz.net

IPv6 Maintenance (6man) Working Group  
Internet-Draft  
Obsoletes: 4941 (if approved)  
Intended status: Standards Track  
Expires: May 6, 2021

F. Gont  
SI6 Networks  
S. Krishnan  
Kaloom  
T. Narten

R. Draves  
Microsoft Research  
November 2, 2020

Temporary Address Extensions for Stateless Address Autoconfiguration in  
IPv6  
draft-ietf-6man-rfc4941bis-12

Abstract

This document describes an extension to IPv6 Stateless Address Autoconfiguration that causes hosts to generate global scope addresses with randomized interface identifiers that change over time. Changing global scope addresses over time limits the window of time during which eavesdroppers and other information collectors may trivially perform address-based network activity correlation when the same address is employed for multiple transactions by the same host. Additionally, it reduces the window of exposure of a host as being accessible via an address that becomes revealed as a result of active communication. This document obsoletes RFC4941.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 6, 2021.

## Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                                                                            |    |
|----------------------------------------------------------------------------|----|
| 1. Introduction . . . . .                                                  | 3  |
| 1.1. Terminology . . . . .                                                 | 3  |
| 1.2. Problem Statement . . . . .                                           | 4  |
| 2. Background . . . . .                                                    | 4  |
| 2.1. Extended Use of the Same Identifier . . . . .                         | 4  |
| 2.2. Possible Approaches . . . . .                                         | 6  |
| 3. Protocol Description . . . . .                                          | 6  |
| 3.1. Design Guidelines . . . . .                                           | 7  |
| 3.2. Assumptions . . . . .                                                 | 7  |
| 3.3. Generation of Randomized Interface Identifiers . . . . .              | 8  |
| 3.3.1. Simple Randomized Interface Identifiers . . . . .                   | 8  |
| 3.3.2. Hash-based Generation of Randomized Interface Identifiers . . . . . | 9  |
| 3.4. Generating Temporary Addresses . . . . .                              | 11 |
| 3.5. Expiration of Temporary Addresses . . . . .                           | 12 |
| 3.6. Regeneration of Temporary Addresses . . . . .                         | 13 |
| 3.7. Implementation Considerations . . . . .                               | 14 |
| 3.8. Defined Constants and Configuration Variables . . . . .               | 14 |
| 4. Implications of Changing Interface Identifiers . . . . .                | 15 |
| 5. Significant Changes from RFC4941 . . . . .                              | 17 |
| 6. Future Work . . . . .                                                   | 18 |
| 7. Implementation Status . . . . .                                         | 19 |
| 8. IANA Considerations . . . . .                                           | 19 |
| 9. Security Considerations . . . . .                                       | 19 |
| 10. Acknowledgments . . . . .                                              | 20 |
| 11. References . . . . .                                                   | 21 |
| 11.1. Normative References . . . . .                                       | 21 |
| 11.2. Informative References . . . . .                                     | 22 |
| Authors' Addresses . . . . .                                               | 24 |

## 1. Introduction

[RFC4862] specifies "Stateless Address Autoconfiguration (SLAAC) for IPv6", which typically results in hosts configuring one or more "stable" IPv6 addresses composed of a network prefix advertised by a local router and a locally-generated Interface Identifier (IID). The security and privacy implications of such addresses have been discussed in detail in [RFC7721], [RFC7217], and [RFC7707]. This document specifies an extension for SLAAC to generate temporary addresses, that can help mitigate some of the aforementioned issues. This is a revision of RFC4941, and formally obsoletes RFC4941. Section 5 describes the changes from [RFC4941].

The default address selection for IPv6 has been specified in [RFC6724]. The determination as to whether to use stable versus temporary addresses can in some cases only be made by an application. For example, some applications may always want to use temporary addresses, while others may want to use them only in some circumstances or not at all. An Application Programming Interface (API) such as that specified in [RFC5014] can enable individual applications to indicate a preference for the use of temporary addresses.

Section 2 provides background information. Section 3 describes a procedure for generating temporary addresses. Section 4 discusses implications of changing interface identifiers (IIDs). Section 5 describes the changes from [RFC4941].

### 1.1. 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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The terms "public address", "stable address", "temporary address", "constant IID", "stable IID", and "temporary IID" are to be interpreted as specified in [RFC7721].

The term "global scope addresses" is used in this document to collectively refer to "Global unicast addresses" as defined in [RFC4291] and "Unique local addresses" as defined in [RFC4193], and not to "globally reachable" addresses, as defined in [RFC8190].

## 1.2. Problem Statement

Addresses generated using stateless address autoconfiguration [RFC4862] contain an embedded interface identifier, which may remain stable over time. Anytime a fixed identifier is used in multiple contexts, it becomes possible to correlate seemingly unrelated activity using this identifier.

The correlation can be performed by

- o An attacker who is in the path between the host in question and the peer(s) to which it is communicating, and who can view the IPv6 addresses present in the datagrams.
- o An attacker who can access the communication logs of the peers with which the host has communicated.

Since the identifier is embedded within the IPv6 address, it cannot be hidden. This document proposes a solution to this issue by generating interface identifiers that vary over time.

Note that an attacker, who is on path, may be able to perform significant correlation based on:

- o The payload contents of unencrypted packets on the wire
- o The characteristics of the packets such as packet size and timing

Use of temporary addresses will not prevent such correlation, nor will it prevent an on-link observer (e.g. the host's default router) from tracking all the host's addresses.

## 2. Background

This section discusses the problem in more detail, provides context for evaluating the significance of the concerns in specific environments, and makes comparisons with existing practices.

### 2.1. Extended Use of the Same Identifier

The use of a non-changing interface identifier to form addresses is a specific instance of the more general case where a constant identifier is reused over an extended period of time and in multiple independent activities. Any time the same identifier is used in multiple contexts, it becomes possible for that identifier to be used to correlate seemingly unrelated activity. For example, a network sniffer placed strategically on a link across which all traffic to/from a particular host crosses could keep track of which destinations

a host communicated with and at what times. Such information can in some cases be used to infer things, such as what hours an employee was active, when someone is at home, etc. Although it might appear that changing an address regularly in such environments would be desirable to lessen privacy concerns, it should be noted that the network prefix portion of an address also serves as a constant identifier. All hosts at, say, a home, would have the same network prefix, which identifies the topological location of those hosts. This has implications for privacy, though not at the same granularity as the concern that this document addresses. Specifically, all hosts within a home could be grouped together for the purposes of collecting information. If the network contains a very small number of hosts, say, just one, changing just the interface identifier will not enhance privacy, since the prefix serves as a constant identifier.

One of the requirements for correlating seemingly unrelated activities is the use (and reuse) of an identifier that is recognizable over time within different contexts. IP addresses provide one obvious example, but there are more. For example,

- o Many hosts also have DNS names associated with their addresses, in which case the DNS name serves as a similar identifier. Although the DNS name associated with an address is more work to obtain (it may require a DNS query), the information is often readily available. In such cases, changing the address on a host over time would do little to address the concerns raised in this document, unless the DNS name is changed at the same time as well (see Section 4).
- o Web browsers and servers typically exchange "cookies" with each other [RFC6265]. Cookies allow web servers to correlate a current activity with a previous activity. One common usage is to send back targeted advertising to a user by using the cookie supplied by the browser to identify what earlier queries had been made (e.g., for what type of information). Based on the earlier queries, advertisements can be targeted to match the (assumed) interests of the end-user.

The use of a constant identifier within an address is of special concern because addresses are a fundamental requirement of communication and cannot easily be hidden from eavesdroppers and other parties. Even when higher layers encrypt their payloads, addresses in packet headers appear in the clear. Consequently, if a mobile host (e.g., laptop) accessed the network from several different locations, an eavesdropper might be able to track the movement of that mobile host from place to place, even if the upper layer payloads were encrypted.

Changing global scope addresses over time limits the time window over which eavesdroppers and other information collectors may trivially correlate network activity when the same address is employed for multiple transactions by the same host. Additionally, it reduces the window of exposure of a host as being accessible via an address that becomes revealed as a result of active communication.

The security and privacy implications of IPv6 addresses are discussed in detail in [RFC7721], [RFC7707], and [RFC7217].

## 2.2. Possible Approaches

One approach, compatible with the stateless address autoconfiguration architecture, would be to change the interface identifier portion of an address over time. Changing the interface identifier can make it more difficult to look at the IP addresses in independent transactions and identify which ones actually correspond to the same host, both in the case where the routing prefix portion of an address changes and when it does not.

Many hosts function as both clients and servers. In such cases, the host would need a name (e.g. a DNS domain name) for its use as a server. Whether the address stays fixed or changes has little privacy implication since the name remains constant and serves as a constant identifier. When acting as a client (e.g., initiating communication), however, such a host may want to vary the addresses it uses. In such environments, one may need multiple addresses: a stable address associated with the name, that is used to accept incoming connection requests from other hosts, and a temporary address used to shield the identity of the client when it initiates communication.

On the other hand, a host that functions only as a client may want to employ only temporary addresses for public communication.

To make it difficult to make educated guesses as to whether two different interface identifiers belong to the same host, the algorithm for generating alternate identifiers must include input that has an unpredictable component from the perspective of the outside entities that are collecting information.

## 3. Protocol Description

The following subsections define the procedures for the generation of IPv6 temporary addresses.

### 3.1. Design Guidelines

Temporary addresses observe the following properties:

1. Temporary addresses are typically employed for initiating outgoing sessions.
2. Temporary addresses are used for a short period of time (typically hours to days) and are subsequently deprecated. Deprecated addresses can continue to be used for established connections, but are not used to initiate new connections.
3. New temporary addresses are generated over time to replace temporary addresses that expire.
4. Temporary addresses must have a limited lifetime (limited "valid lifetime" and "preferred lifetime" from [RFC4862]). The lifetime of an address should be further reduced when privacy-meaningful events (such as a host attaching to a different network, or the regeneration of a new randomized MAC address) takes place. The lifetime of temporary addresses must be statistically different for different addresses, such that it is hard to predict or infer when a new temporary address is generated, or correlate a newly-generated address with an existing one.
5. By default, one address is generated for each prefix advertised by stateless address autoconfiguration. The resulting Interface Identifiers must be statistically different when addresses are configured for different prefixes or different network interfaces. This means that, given two addresses, it must be difficult for an outside entity to infer whether the addresses correspond to the same host or network interface.
6. It must be difficult for an outside entity to predict the Interface Identifiers that will be employed for temporary addresses, even with knowledge of the algorithm/method employed to generate them and/or knowledge of the Interface Identifiers previously employed for other temporary addresses. These Interface Identifiers must be semantically opaque [RFC7136] and must not follow any specific patterns.

### 3.2. Assumptions

The following algorithm assumes that for a given temporary address, an implementation can determine the prefix from which it was generated. When a temporary address is deprecated, a new temporary address is generated. The specific valid and preferred lifetimes for



the new address are dependent on the corresponding lifetime values set for the prefix from which it was generated.

Finally, this document assumes that when a host initiates outgoing communication, temporary addresses can be given preference over stable addresses (if available), when the device is configured to do so. [RFC6724] mandates implementations to provide a mechanism, which allows an application to configure its preference for temporary addresses over stable addresses. It also allows for an implementation to prefer temporary addresses by default, so that the connections initiated by the host can use temporary addresses without requiring application-specific enablement. This document also assumes that an API will exist that allows individual applications to indicate whether they prefer to use temporary or stable addresses and override the system defaults (see e.g. [RFC5014]).

### 3.3. Generation of Randomized Interface Identifiers

The following subsections specify example algorithms for generating temporary interface identifiers that follow the guidelines in Section 3.1 of this document. The algorithm specified in Section 3.3.1 benefits from a Pseudo-Random Number Generator (PRNG) available on the system. The algorithm specified in Section 3.3.2 allows for code reuse by hosts that implement [RFC7217].

#### 3.3.1. Simple Randomized Interface Identifiers

One approach is to select a pseudorandom number of the appropriate length. A host employing this algorithm should generate IIDs as follows:

1. Obtain a random number from a pseudo-random number generator (PRNG) that can produce random numbers of at least as many bits as required for the Interface Identifier (please see the next step). [RFC4086] specifies randomness requirements for security.
2. The Interface Identifier is obtained by taking as many bits from the random number obtained in the previous step as necessary. See [RFC7136] for the necessary number of bits, that is, the length of the IID. See also [RFC7421] for a discussion of the privacy implications of the IID length. Note: there are no special bits in an Interface Identifier [RFC7136].
3. The resulting Interface Identifier MUST be compared against the reserved IPv6 Interface Identifiers [RFC5453] [IANA-RESERVED-IID] and against those Interface Identifiers already employed in an address of the same network interface and the same network prefix. In the event that an unacceptable identifier has been

generated, a new interface identifier should be generated, by repeating the algorithm from the first step.

### 3.3.2. Hash-based Generation of Randomized Interface Identifiers

The algorithm in [RFC7217] can be augmented for the generation of temporary addresses. The benefit of this would be that a host could employ a single algorithm for generating stable and temporary addresses, by employing appropriate parameters.

Hosts would employ the following algorithm for generating the temporary IID:

1. Compute a random identifier with the expression:

$$RID = F(\text{Prefix}, \text{Net\_Iface}, \text{Network\_ID}, \text{Time}, \text{DAD\_Counter}, \text{secret\_key})$$

Where:

RID:

Random Identifier

F():

A pseudorandom function (PRF) that MUST NOT be computable from the outside (without knowledge of the secret key). F() MUST also be difficult to reverse, such that it resists attempts to obtain the secret\_key, even when given samples of the output of F() and knowledge or control of the other input parameters. F() SHOULD produce an output of at least as many bits as required for the Interface Identifier. F() could be the result of applying a cryptographic hash over an encoded version of the function parameters. While this document does not recommend a specific mechanism for encoding the function parameters (or a specific cryptographic hash function), a cryptographically robust construction will ensure that the mapping from parameters to the hash function input is an injective map, as might be attained by using fixed-width encodings and/or length-prefixing variable-length parameters. SHA-256 [FIPS-SHS] is one possible option for F(). Note: MD5 [RFC1321] is considered unacceptable for F() [RFC6151].

Prefix:

The prefix to be used for SLAAC, as learned from an ICMPv6 Router Advertisement message.

Net\_Iface:

The MAC address corresponding to the underlying network interface card, in the case the link uses IEEE802 link-layer identifiers. Employing the MAC address for this parameter (over the other suggested options in RFC7217) means that the re-generation of a randomized MAC address will result in a different temporary address.

**Network\_ID:**

Some network-specific data that identifies the subnet to which this interface is attached -- for example, the IEEE 802.11 Service Set Identifier (SSID) corresponding to the network to which this interface is associated. Additionally, "Simple Procedures for Detecting Network Attachment in IPv6" ("Simple DNA") [RFC6059] describes ideas that could be leveraged to generate a Network\_ID parameter. This parameter SHOULD be employed if some form of "Network\_ID" is available.

**Time:**

An implementation-dependent representation of time. One possible example is the representation in UNIX-like systems [OPEN-GROUP], that measure time in terms of the number of seconds elapsed since the Epoch (00:00:00 Coordinated Universal Time (UTC), 1 January 1970). The addition of the "Time" argument results in (statistically) different interface identifiers over time.

**DAD\_Counter:**

A counter that is employed to resolve Duplicate Address Detection (DAD) conflicts.

**secret\_key:**

A secret key that is not known by the attacker. The secret key SHOULD be of at least 128 bits. It MUST be initialized to a pseudo-random number (see [RFC4086] for randomness requirements for security) when the operating system is "bootstrapped". The secret\_key MUST NOT be employed for any other purpose than the one discussed in this section. For example, implementations MUST NOT employ the same secret\_key for the generation of stable addresses [RFC7217] and the generation of temporary addresses via this algorithm.

2. The Interface Identifier is finally obtained by taking as many bits from the RID value (computed in the previous step) as necessary, starting from the least significant bit. See [RFC7136] for the necessary number of bits, that is, the length of the IID. See also [RFC7421] for a discussion of the privacy implications of the IID length. Note: there are no special bits in an Interface Identifier [RFC7136].

3. The resulting Interface Identifier MUST be compared against the reserved IPv6 Interface Identifiers [RFC5453] [IANA-RESERVED-IID] and against those Interface Identifiers already employed in an address of the same network interface and the same network prefix. In the event that an unacceptable identifier has been generated, the value DAD\_Counter should be incremented by 1, and the algorithm should be restarted from the first step.

#### 3.4. Generating Temporary Addresses

[RFC4862] describes the steps for generating a link-local address when an interface becomes enabled as well as the steps for generating addresses for other scopes. This document extends [RFC4862] as follows. When processing a Router Advertisement with a Prefix Information option carrying a prefix for the purposes of address autoconfiguration (i.e., the A bit is set), the host MUST perform the following steps:

1. Process the Prefix Information Option as defined in [RFC4862], adjusting the lifetimes of existing temporary addresses, with the overall constraint that no temporary addresses should ever remain "valid" or "preferred" for a time longer than (TEMP\_VALID\_LIFETIME) or (TEMP\_PREFERRED\_LIFETIME - DESYNC\_FACTOR) respectively. The configuration variables TEMP\_VALID\_LIFETIME and TEMP\_PREFERRED\_LIFETIME correspond to maximum target lifetimes for temporary addresses.
2. One way an implementation can satisfy the above constraints is to associate with each temporary address a creation time (called CREATION\_TIME) that indicates the time at which the address was created. When updating the preferred lifetime of an existing temporary address, it would be set to expire at whichever time is earlier: the time indicated by the received lifetime or (CREATION\_TIME + TEMP\_PREFERRED\_LIFETIME - DESYNC\_FACTOR). A similar approach can be used with the valid lifetime.
3. If the host has not configured any temporary address for the corresponding prefix, the host SHOULD create a new temporary address for such prefix.

Note:

For example, a host might implement prefix-specific policies such as not configuring temporary addresses for the Unique Local IPv6 Unicast Addresses (ULA) [RFC4193] prefix.

4. When creating a temporary address, the DESYNC\_FACTOR MUST be computed for this prefix, and the lifetime values MUST be derived from the corresponding prefix as follows:

- \* Its Valid Lifetime is the lower of the Valid Lifetime of the prefix and TEMP\_VALID\_LIFETIME.
  - \* Its Preferred Lifetime is the lower of the Preferred Lifetime of the prefix and TEMP\_PREFERRED\_LIFETIME - DESYNC\_FACTOR.
5. A temporary address is created only if this calculated Preferred Lifetime is greater than REGEN\_ADVANCE time units. In particular, an implementation MUST NOT create a temporary address with a zero Preferred Lifetime.
  6. New temporary addresses MUST be created by appending a randomized interface identifier to the prefix that was received. Section 3.3 of this document specifies some sample algorithms for generating the randomized interface identifier.
  7. The host MUST perform duplicate address detection (DAD) on the generated temporary address. If DAD indicates the address is already in use, the host MUST generate a new randomized interface identifier, and repeat the previous steps as appropriate up to TEMP\_IDGEN\_RETRIES times. If after TEMP\_IDGEN\_RETRIES consecutive attempts no non-unique address was generated, the host MUST log a system error and SHOULD NOT attempt to generate a temporary address for the given prefix for the duration of the host's attachment to the network via this interface. This allows hosts to recover from occasional DAD failures, or otherwise log the recurrent address collisions.

### 3.5. Expiration of Temporary Addresses

When a temporary address becomes deprecated, a new one MUST be generated. This is done by repeating the actions described in Section 3.4, starting at step 4). Note that, except for the transient period when a temporary address is being regenerated, in normal operation at most one temporary address per prefix should be in a non-deprecated state at any given time on a given interface. Note that if a temporary address becomes deprecated as result of processing a Prefix Information Option with a zero Preferred Lifetime, then a new temporary address MUST NOT be generated. To ensure that a preferred temporary address is always available, a new temporary address SHOULD be regenerated slightly before its predecessor is deprecated. This is to allow sufficient time to avoid race conditions in the case where generating a new temporary address is not instantaneous, such as when duplicate address detection must be run. The host SHOULD start the address regeneration process REGEN\_ADVANCE time units before a temporary address would actually be deprecated.

As an optional optimization, an implementation MAY remove a deprecated temporary address that is not in use by applications or upper layers as detailed in Section 6.

### 3.6. Regeneration of Temporary Addresses

The frequency at which temporary addresses change depends on how a device is being used (e.g., how frequently it initiates new communication) and the concerns of the end user. The most egregious privacy concerns appear to involve addresses used for long periods of time (weeks to months to years). The more frequently an address changes, the less feasible collecting or coordinating information keyed on interface identifiers becomes. Moreover, the cost of collecting information and attempting to correlate it based on interface identifiers will only be justified if enough addresses contain non-changing identifiers to make it worthwhile. Thus, having large numbers of clients change their address on a daily or weekly basis is likely to be sufficient to alleviate most privacy concerns.

There are also client costs associated with having a large number of addresses associated with a host (e.g., in doing address lookups, the need to join many multicast groups, etc.). Thus, changing addresses frequently (e.g., every few minutes) may have performance implications.

Hosts following this specification SHOULD generate new temporary addresses on a periodic basis. This can be achieved by generating a new temporary address at least once every  $(\text{TEMP\_PREFERRED\_LIFETIME} - \text{REGEN\_ADVANCE} - \text{DESYNC\_FACTOR})$  time units. As described above, generating a new temporary address  $\text{REGEN\_ADVANCE}$  time units before a temporary address becomes deprecated produces addresses with a preferred lifetime no larger than  $\text{TEMP\_PREFERRED\_LIFETIME}$ . The value  $\text{DESYNC\_FACTOR}$  is a random value computed for a prefix when a temporary address is generated, that ensures that clients do not generate new addresses with a fixed frequency, and that clients do not synchronize with each other and generate new addresses at exactly the same time. When the preferred lifetime expires, a new temporary address MUST be generated using the algorithm specified in Section 3.4.

Because the precise frequency at which it is appropriate to generate new addresses varies from one environment to another, implementations SHOULD provide end users with the ability to change the frequency at which addresses are regenerated. The default value is given in  $\text{TEMP\_PREFERRED\_LIFETIME}$  and is one day. In addition, the exact time at which to invalidate a temporary address depends on how applications are used by end users. Thus, the suggested default value of two days ( $\text{TEMP\_VALID\_LIFETIME}$ ) may not be appropriate in all

environments. Implementations SHOULD provide end users with the ability to override both of these default values.

Finally, when an interface connects to a new (different) link, existing temporary addresses for the corresponding interface MUST be eliminated, and new temporary addresses MUST be generated immediately for use on the new link. If a device moves from one link to another, generating new temporary addresses ensures that the device uses different randomized interface identifiers for the temporary addresses associated with the two links, making it more difficult to correlate addresses from the two different links as being from the same hosts. The host MAY follow any process available to it, to determine that the link change has occurred. One such process is described by Simple DNA [RFC6059]. Detecting link changes would prevent link down/up events from causing temporary addresses to be (unnecessarily) regenerated.

### 3.7. Implementation Considerations

Devices implementing this specification MUST provide a way for the end user to explicitly enable or disable the use of temporary addresses. In addition, a site might wish to disable the use of temporary addresses in order to simplify network debugging and operations. Consequently, implementations SHOULD provide a way for trusted system administrators to enable or disable the use of temporary addresses.

Additionally, sites might wish to selectively enable or disable the use of temporary addresses for some prefixes. For example, a site might wish to disable temporary address generation for "Unique local" [RFC4193] prefixes while still generating temporary addresses for all other global prefixes. Another site might wish to enable temporary address generation only for the prefixes 2001:db8:1::/48 and 2001:db8:2::/48 while disabling it for all other prefixes. To support this behavior, implementations SHOULD provide a way to enable and disable generation of temporary addresses for specific prefix subranges. This per-prefix setting SHOULD override the global settings on the host with respect to the specified prefix subranges. Note that the per-prefix setting can be applied at any granularity, and not necessarily on a per subnet basis.

### 3.8. Defined Constants and Configuration Variables

Constants and configuration variables defined in this document include:

TEMP\_VALID\_LIFETIME -- Default value: 2 days. Users should be able to override the default value.

TEMP\_PREFERRED\_LIFETIME -- Default value: 1 day. Users should be able to override the default value.

Note:

The TEMP\_PREFERRED\_LIFETIME value MUST be less than the TEMP\_VALID\_LIFETIME value, to avoid the pathological case where an address is employed for new communications, but becomes invalid in less than 1 second, disrupting those communications

REGEN\_ADVANCE --  $2 + (\text{TEMP\_IDGEN\_RETRIES} * \text{DupAddrDetectTransmits} * \text{RetransTimer} / 1000)$

Notes:

This parameter is specified as a function of other protocol parameters, to account for the time possibly spent in Duplicate Address Detection (DAD) in the worst-case scenario of TEMP\_IDGEN\_RETRIES. This prevents the pathological case where the generation of a new temporary address is not started with enough anticipation such that a new preferred address is generated before the currently-preferred temporary address becomes deprecated.

RetransTimer is specified in [RFC4861], while DupAddrDetectTransmits is specified in [RFC4862]. Since RetransTimer is specified in units of milliseconds, this expression employs the constant "1000" such that REGEN\_ADVANCE is expressed in seconds.

MAX\_DESYNC\_FACTOR --  $0.4 * \text{TEMP\_PREFERRED\_LIFETIME}$ . Upper bound on DESYNC\_FACTOR.

Note:

Setting MAX\_DESYNC\_FACTOR to  $0.4 * \text{TEMP\_PREFERRED\_LIFETIME}$  results in addresses that have statistically different lifetimes, and a maximum of 3 concurrent temporary addresses when the default parameters specified in this section are employed.

DESYNC\_FACTOR -- A random value within the range 0 - MAX\_DESYNC\_FACTOR. It is computed for a prefix each time a temporary address is generated, and must be smaller than  $(\text{TEMP\_PREFERRED\_LIFETIME} - \text{REGEN\_ADVANCE})$ .

TEMP\_IDGEN\_RETRIES -- Default value: 3

#### 4. Implications of Changing Interface Identifiers

The desire to protect individual privacy can conflict with the desire to effectively maintain and debug a network. Having clients use addresses that change over time will make it more difficult to track



down and isolate operational problems. For example, when looking at packet traces, it could become more difficult to determine whether one is seeing behavior caused by a single errant host, or by a number of them.

Network deployments are currently recommended to provide multiple IPv6 addresses from each prefix to general-purpose hosts [RFC7934]. However, in some scenarios, use of a large number of IPv6 addresses may have negative implications on network devices that need to maintain entries for each IPv6 address in some data structures (e.g., [RFC7039]). For example, concurrent active use of multiple IPv6 addresses will increase neighbor discovery traffic if Neighbor Caches in network devices are not large enough to store all addresses on the link. This can impact performance and energy efficiency on networks on which multicast is expensive (e.g. [I-D.ietf-mboned-ieee802-mcast-problems]). Additionally, some network security devices might incorrectly infer IPv6 address forging if temporary addresses are regenerated at a high rate.

The use of temporary addresses may cause unexpected difficulties with some applications. For example, some servers refuse to accept communications from clients for which they cannot map the IP address into a DNS name. That is, they perform a DNS PTR query to determine the DNS name, and may then also perform an AAAA query on the returned name to verify that the returned DNS name maps back into the address being used. Consequently, clients not properly registered in the DNS may be unable to access some services. However, a host's DNS name (if non-changing) would serve as a constant identifier. The wide deployment of the extension described in this document could challenge the practice of inverse-DNS-based "validation", which has little validity, though it is widely implemented. In order to meet server challenges, hosts could register temporary addresses in the DNS using random names (for example, a string version of the random address itself), albeit at the expense of increased complexity.

In addition, some applications may not behave robustly if temporary addresses are used and an address expires before the application has terminated, or if it opens multiple sessions, but expects them to all use the same addresses.

[RFC4941] employed a randomized temporary Interface Identifier for generating a set of temporary addresses, such that temporary addresses configured at a given time for multiple SLAAC prefixes would employ the same Interface Identifier. Sharing the same IID among multiple address allowed host to join only one solicited-node multicast group per temporary address set.

This document requires that the Interface Identifiers of all temporary addresses on a host are statistically different from each other. This means that when a network employs multiple prefixes, each temporary address of a set will result in a different solicited-node multicast address, and thus the number of multicast groups that a host must join becomes a function of the number of SLAAC prefixes employed for generating temporary addresses.

Thus, a network that employs multiple prefixes may require hosts to join more multicast groups than for an RFC4941 implementation. If the number of multicast groups were large enough, a node might need to resort to setting the network interface card to promiscuous mode. This could cause the node to process more packets than strictly necessary, and might have a negative impact on battery-life, and on system performance in general.

We note that since this document reduces the default TEMP\_VALID\_LIFETIME from 7 days (in [RFC4941]) to 2 days, the number of concurrent temporary addresses per SLAAC prefix will be smaller than for RFC4941 implementations, and thus the number of multicast groups for a network that employs, say, between 1 and three prefixes will be similar than of RFC4941 implementations.

Implementations concerned with the maximum number of multicast groups that would be required to join as a result of configured addresses, or the overall number of configured addresses, should consider enforcing implementation-specific limits on e.g. the maximum number of configured addresses, the maximum number of SLAAC prefixes that are employed for auto-configuration, and/or the maximum ratio for TEMP\_VALID\_LIFETIME/TEMP\_PREFERRED\_LIFETIME (that ultimately controls the approximate number of concurrent temporary addresses per SLAAC prefix). Many of these configuration limits are readily available in SLAAC and RFC4941 implementations. We note that these configurable limits are meant to prevent pathological behaviors (as opposed to simply limiting the usage of IPv6 addresses), since IPv6 implementations are expected to leverage the usage of multiple addresses [RFC7934].

## 5. Significant Changes from RFC4941

This section summarizes the substantive changes in this document relative to RFC 4941.

Broadly speaking, this document introduces the following changes:

- o Addresses a number of flaws in the algorithm for generating temporary addresses: The aforementioned flaws include the use of MD5 for computing the temporary IIDs, and reusing the same IID for

multiple prefixes (see [RAID2015] and [RFC7721] for further details).

- o Allows hosts to employ only temporary addresses: [RFC4941] assumed that temporary addresses were configured in addition to stable addresses. This document does not imply or require the configuration of stable addresses, and thus implementations can now configure both stable and temporary addresses, or temporary addresses only.
- o Removes the recommendation that temporary addresses be disabled by default:  
This is in line with BCP188 ([RFC7258]), and also with BCP204 ([RFC7934]).
- o Reduces the default maximum Valid Lifetime for temporary addresses: The default Valid Lifetime for temporary addresses has been reduced from 1 week to 2 days, decreasing the typical number of concurrent temporary addresses from 7 to 3. This reduces the possible stress on network elements (see Section 4 for further details).
- o DESYNC\_FACTOR is computed on a per-prefix basis each time a temporary address is generated, such that each temporary address has a statistically different preferred lifetime, and that temporary addresses are not generated at a constant frequency.
- o Changes the requirement to not try to regenerate temporary addresses upon DAD failures from "MUST NOT" to "SHOULD NOT".
- o The discussion about the security and privacy implications of different address generation techniques has been replaced with references to recent work in this area ([RFC7707], [RFC7721], and [RFC7217]).
- o Addresses all errata submitted for [RFC4941].

## 6. Future Work

An implementation might want to keep track of which addresses are being used by upper layers so as to be able to remove a deprecated temporary address from internal data structures once no upper layer protocols are using it (but not before). This is in contrast to current approaches where addresses are removed from an interface when they become invalid [RFC4862], independent of whether or not upper layer protocols are still using them. For TCP connections, such information is available in control blocks. For UDP-based applications, it may be the case that only the applications have

knowledge about what addresses are actually in use. Consequently, an implementation generally will need to use heuristics in deciding when an address is no longer in use.

## 7. Implementation Status

[The RFC-Editor should remove this section before publishing this document as an RFC]

The following are known implementations of this document:

- o FreeBSD kernel: There is a FreeBSD kernel implementation of this document, albeit not yet committed. The implementation has been done in April 2020 by Fernando Gont <fgont@si6networks.com>. The corresponding patch can be found at:  
<<https://www.gont.com.ar/code/fgont-patch-freebsd-rfc4941bis.txt>>
- o Linux kernel: A Linux kernel implementation of this document has been committed to the net-next tree. The implementation has been produced in April 2020 by Fernando Gont <fgont@si6networks.com>. The corresponding patch can be found at:  
<<https://patchwork.ozlabs.org/project/netdev/patch/20200501035147.GA1587@archlinux-current.localdomain/>>
- o slaacd(8): slaacd(8) has traditionally used different randomized interface identifiers for each prefix, and it has recently reduced the Valid Lifetime of temporary addresses as specified in Section 3.8, thus fully implementing this document. The implementation has been done by Florian Obser <florian@openbsd.org>, with the update to the temporary address Valid Lifetime applied in March 2020. The implementation can be found at: <<https://github.com/openbsd/src/tree/master/sbin/slaacd>>

## 8. IANA Considerations

There are no IANA registries within this document. The RFC-Editor can remove this section before publication of this document as an RFC.

## 9. Security Considerations

If a very small number of hosts (say, only one) use a given prefix for extended periods of time, just changing the interface identifier part of the address may not be sufficient to mitigate address-based network activity correlation, since the prefix acts as a constant identifier. The procedures described in this document are most effective when the prefix is reasonably non static or is used by a fairly large number of hosts. Additionally, if a temporary address

is used in a session where the user authenticates, any notion of "privacy" for that address is compromised for the part(ies) that receive the authentication information.

While this document discusses ways to limit the lifetime of Interface Identifiers to reduce the ability of attackers to perform address-based network activity correlation, the method described is believed to be ineffective against sophisticated forms of traffic analysis. To increase effectiveness, one may need to consider the use of more advanced techniques, such as Onion Routing [ONION].

Ingress filtering has been and is being deployed as a means of preventing the use of spoofed source addresses in Distributed Denial of Service (DDoS) attacks. In a network with a large number of hosts, new temporary addresses are created at a fairly high rate. This might make it difficult for ingress filtering mechanisms to distinguish between legitimately changing temporary addresses and spoofed source addresses, which are "in-prefix" (using a topologically correct prefix and non-existent interface ID). This can be addressed by using access control mechanisms on a per-address basis on the network egress point, though as noted in Section 4 there are corresponding costs for doing so.

## 10. Acknowledgments

The authors would like to thank (in alphabetical order) Fred Baker, Brian Carpenter, Tim Chown, Lorenzo Colitti, Roman Danyliw, David Farmer, Tom Herbert, Bob Hinden, Christian Huitema, Benjamin Kaduk, Erik Kline, Gyan Mishra, Dave Plonka, Alvaro Retana, Michael Richardson, Mark Smith, Dave Thaler, Pascal Thubert, Ole Troan, Johanna Ullrich, Eric Vyncke, and Timothy Winters, for providing valuable comments on earlier versions of this document.

This document incorporates errata submitted for [RFC4941] by Jiri Bohac and Alfred Hoenes.

This document is based on [RFC4941] (a revision of RFC3041). Suresh Krishnan was the sole author of RFC4941. He would like to acknowledge the contributions of the IPv6 working group and, in particular, Jari Arkko, Pekka Nikander, Pekka Savola, Francis Dupont, Brian Haberman, Tatuya Jinmei, and Margaret Wasserman for their detailed comments.

Rich Draves and Thomas Narten were the authors of RFC 3041. They would like to acknowledge the contributions of the IPv6 working group and, in particular, Ran Atkinson, Matt Crawford, Steve Deering, Allison Mankin, and Peter Bieringer.

## 11. References

### 11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<https://www.rfc-editor.org/info/rfc4193>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<https://www.rfc-editor.org/info/rfc4291>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.
- [RFC5453] Krishnan, S., "Reserved IPv6 Interface Identifiers", RFC 5453, DOI 10.17487/RFC5453, February 2009, <<https://www.rfc-editor.org/info/rfc5453>>.
- [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <<https://www.rfc-editor.org/info/rfc6724>>.
- [RFC7136] Carpenter, B. and S. Jiang, "Significance of IPv6 Interface Identifiers", RFC 7136, DOI 10.17487/RFC7136, February 2014, <<https://www.rfc-editor.org/info/rfc7136>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

## 11.2. Informative References

[FIPS-SHS]

NIST, "Secure Hash Standard (SHS)", FIPS Publication 180-4, August 2015, <<https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>>.

[I-D.ietf-mboned-ieee802-mcast-problems]

Perkins, C., McBride, M., Stanley, D., Kumari, W., and J. Zuniga, "Multicast Considerations over IEEE 802 Wireless Media", draft-ietf-mboned-ieee802-mcast-problems-12 (work in progress), October 2020.

[IANA-RESERVED-IID]

IANA, "Reserved IPv6 Interface Identifiers", <<http://www.iana.org/assignments/ipv6-interface-ids>>.

[ONION]

Reed, MGR., Syverson, PFS., and DMG. Goldschlag, "Proxies for Anonymous Routing", Proceedings of the 12th Annual Computer Security Applications Conference, San Diego, CA, December 1996.

[OPEN-GROUP]

The Open Group, "The Open Group Base Specifications Issue 7 / IEEE Std 1003.1-2008, 2016 Edition", Section 4.16 Seconds Since the Epoch, 2016, <<http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/contents.html>>.

[RAID2015]

Ullrich, J. and E. Weippl, "Privacy is Not an Option: Attacking the IPv6 Privacy Extension", International Symposium on Recent Advances in Intrusion Detection (RAID), 2015, <<https://www.sba-research.org/wp-content/uploads/publications/Ullrich2015Privacy.pdf>>.

[RFC1321]

Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <<https://www.rfc-editor.org/info/rfc1321>>.

[RFC4941]

Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<https://www.rfc-editor.org/info/rfc4941>>.

- [RFC5014] Nordmark, E., Chakrabarti, S., and J. Laganier, "IPv6 Socket API for Source Address Selection", RFC 5014, DOI 10.17487/RFC5014, September 2007, <<https://www.rfc-editor.org/info/rfc5014>>.
- [RFC6059] Krishnan, S. and G. Daley, "Simple Procedures for Detecting Network Attachment in IPv6", RFC 6059, DOI 10.17487/RFC6059, November 2010, <<https://www.rfc-editor.org/info/rfc6059>>.
- [RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <<https://www.rfc-editor.org/info/rfc6151>>.
- [RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <<https://www.rfc-editor.org/info/rfc6265>>.
- [RFC7039] Wu, J., Bi, J., Bagnulo, M., Baker, F., and C. Vogt, Ed., "Source Address Validation Improvement (SAVI) Framework", RFC 7039, DOI 10.17487/RFC7039, October 2013, <<https://www.rfc-editor.org/info/rfc7039>>.
- [RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <<https://www.rfc-editor.org/info/rfc7217>>.
- [RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <<https://www.rfc-editor.org/info/rfc7258>>.
- [RFC7421] Carpenter, B., Ed., Chown, T., Gont, F., Jiang, S., Petrescu, A., and A. Yourtchenko, "Analysis of the 64-bit Boundary in IPv6 Addressing", RFC 7421, DOI 10.17487/RFC7421, January 2015, <<https://www.rfc-editor.org/info/rfc7421>>.
- [RFC7707] Gont, F. and T. Chown, "Network Reconnaissance in IPv6 Networks", RFC 7707, DOI 10.17487/RFC7707, March 2016, <<https://www.rfc-editor.org/info/rfc7707>>.
- [RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <<https://www.rfc-editor.org/info/rfc7721>>.



- [RFC7934] Colitti, L., Cerf, V., Cheshire, S., and D. Schinazi,  
"Host Address Availability Recommendations", BCP 204,  
RFC 7934, DOI 10.17487/RFC7934, July 2016,  
<<https://www.rfc-editor.org/info/rfc7934>>.
- [RFC8190] Bonica, R., Cotton, M., Haberman, B., and L. Vegoda,  
"Updates to the Special-Purpose IP Address Registries",  
BCP 153, RFC 8190, DOI 10.17487/RFC8190, June 2017,  
<<https://www.rfc-editor.org/info/rfc8190>>.

Authors' Addresses

Fernando Gont  
SI6 Networks  
Evaristo Carriego 2644  
Haedo, Provincia de Buenos Aires 1706  
Argentina

Phone: +54 11 4650 8472  
Email: [fgont@si6networks.com](mailto:fgont@si6networks.com)  
URI: <https://www.si6networks.com>

Suresh Krishnan  
Kaloom

Email: [suresh@kaloom.com](mailto:suresh@kaloom.com)

Thomas Narten

Email: [narten@cs.duke.edu](mailto:narten@cs.duke.edu)

Richard Draves  
Microsoft Research  
One Microsoft Way  
Redmond, WA  
USA

Email: [richdr@microsoft.com](mailto:richdr@microsoft.com)

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: April 24, 2020

C. Filsfils, Ed.  
D. Dukes, Ed.  
Cisco Systems, Inc.  
S. Previdi  
Huawei  
J. Leddy  
Individual  
S. Matsushima  
Softbank  
D. Voyer  
Bell Canada  
October 22, 2019

IPv6 Segment Routing Header (SRH)  
draft-ietf-6man-segment-routing-header-26

Abstract

Segment Routing can be applied to the IPv6 data plane using a new type of Routing Extension Header called the Segment Routing Header. This document describes the Segment Routing Header and how it is used by Segment Routing capable nodes.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 24, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                                                                                 |    |
|---------------------------------------------------------------------------------|----|
| 1. Introduction . . . . .                                                       | 3  |
| 1.1. Requirements Language . . . . .                                            | 3  |
| 2. Segment Routing Header . . . . .                                             | 4  |
| 2.1. SRH TLVs . . . . .                                                         | 6  |
| 2.1.1. Padding TLVs . . . . .                                                   | 8  |
| 2.1.2. HMAC TLV . . . . .                                                       | 9  |
| 3. SR Nodes . . . . .                                                           | 12 |
| 3.1. Source SR Node . . . . .                                                   | 12 |
| 3.2. Transit Node . . . . .                                                     | 12 |
| 3.3. SR Segment Endpoint Node . . . . .                                         | 12 |
| 4. Packet Processing . . . . .                                                  | 13 |
| 4.1. Source SR Node . . . . .                                                   | 13 |
| 4.1.1. Reduced SRH . . . . .                                                    | 13 |
| 4.2. Transit Node . . . . .                                                     | 14 |
| 4.3. SR Segment Endpoint Node . . . . .                                         | 14 |
| 4.3.1. FIB Entry Is Locally Instantiated SRv6 SID . . . . .                     | 14 |
| 4.3.2. FIB Entry Is A Local Interface . . . . .                                 | 16 |
| 4.3.3. FIB Entry Is A Non-Local Route . . . . .                                 | 17 |
| 4.3.4. FIB Entry Is A No Match . . . . .                                        | 17 |
| 5. Intra SR Domain Deployment Model . . . . .                                   | 17 |
| 5.1. Securing the SR Domain . . . . .                                           | 17 |
| 5.2. SR Domain as A Single System with Delegation Among<br>Components . . . . . | 18 |
| 5.3. MTU Considerations . . . . .                                               | 19 |
| 5.4. ICMP Error Processing . . . . .                                            | 19 |
| 5.5. Load Balancing and ECMP . . . . .                                          | 19 |
| 5.6. Other Deployments . . . . .                                                | 20 |
| 6. Illustrations . . . . .                                                      | 20 |
| 6.1. Abstract Representation of an SRH . . . . .                                | 20 |
| 6.2. Example Topology . . . . .                                                 | 21 |
| 6.3. Source SR Node . . . . .                                                   | 22 |
| 6.3.1. Intra SR Domain Packet . . . . .                                         | 22 |
| 6.3.2. Inter SR Domain Packet - Transit . . . . .                               | 22 |
| 6.3.3. Inter SR Domain Packet - Internal to External . . . . .                  | 23 |
| 6.4. Transit Node . . . . .                                                     | 23 |
| 6.5. SR Segment Endpoint Node . . . . .                                         | 23 |
| 6.6. Delegation of Function with HMAC Verification . . . . .                    | 23 |
| 6.6.1. SID List Verification . . . . .                                          | 24 |

|       |                                                 |    |
|-------|-------------------------------------------------|----|
| 7.    | Security Considerations . . . . .               | 24 |
| 7.1.  | Source Routing Attacks . . . . .                | 25 |
| 7.2.  | Service Theft . . . . .                         | 25 |
| 7.3.  | Topology Disclosure . . . . .                   | 25 |
| 7.4.  | ICMP Generation . . . . .                       | 26 |
| 7.5.  | Applicability of AH . . . . .                   | 26 |
| 8.    | IANA Considerations . . . . .                   | 26 |
| 8.1.  | Segment Routing Header Flags Registry . . . . . | 27 |
| 8.2.  | Segment Routing Header TLVs Registry . . . . .  | 27 |
| 9.    | Implementation Status . . . . .                 | 27 |
| 9.1.  | Linux . . . . .                                 | 28 |
| 9.2.  | Cisco Systems . . . . .                         | 28 |
| 9.3.  | FD.io . . . . .                                 | 28 |
| 9.4.  | Barefoot . . . . .                              | 28 |
| 9.5.  | Juniper . . . . .                               | 28 |
| 9.6.  | Huawei . . . . .                                | 29 |
| 10.   | Contributors . . . . .                          | 29 |
| 11.   | Acknowledgements . . . . .                      | 29 |
| 12.   | References . . . . .                            | 29 |
| 12.1. | Normative References . . . . .                  | 29 |
| 12.2. | Informative References . . . . .                | 31 |
|       | Authors' Addresses . . . . .                    | 32 |

## 1. Introduction

Segment Routing can be applied to the IPv6 data plane using a new type of Routing Header called the Segment Routing Header. This document describes the Segment Routing Header and how it is used by Segment Routing capable nodes.

The Segment Routing Architecture [RFC8402] describes Segment Routing and its instantiation in two data planes; MPLS and IPv6.

The encoding of IPv6 segments in the Segment Routing Header is defined in this document.

This document uses the terms Segment Routing, SR Domain, SRv6, Segment ID (SID), SRv6 SID, Active Segment, and SR Policy as defined in [RFC8402].

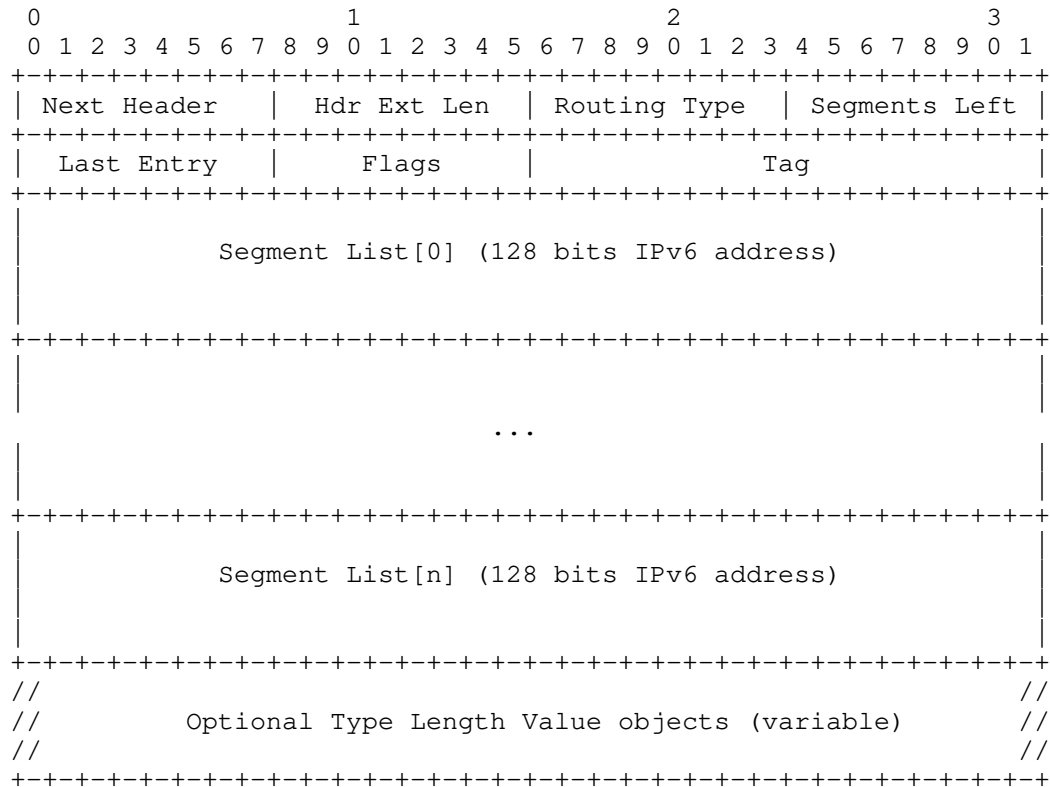
### 1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 2. Segment Routing Header

Routing Headers are defined in [RFC8200]. The Segment Routing Header has a new Routing Type (suggested value 4) to be assigned by IANA.

The Segment Routing Header (SRH) is defined as follows:



where:

- o Next Header: Defined in [RFC8200] Section 4.4
- o Hdr Ext Len: Defined in [RFC8200] Section 4.4
- o Routing Type: TBD, to be assigned by IANA (suggested value: 4).
- o Segments Left: Defined in [RFC8200] Section 4.4
- o Last Entry: contains the index (zero based), in the Segment List, of the last element of the Segment List.

- o Flags: 8 bits of flags. Section 8.1 creates an IANA registry for new flags to be defined. The following flags are defined:

```

 0 1 2 3 4 5 6 7
+-----+
|U U U U U U U U|
+-----+

```

U: Unused and for future use. MUST be 0 on transmission and ignored on receipt.

- o Tag: tag a packet as part of a class or group of packets, e.g., packets sharing the same set of properties. When tag is not used at source it MUST be set to zero on transmission. When tag is not used during SRH Processing it SHOULD be ignored. Tag is not used when processing the SID defined in Section 4.3.1. It may be used when processing other SIDs that are not defined in this document. The allocation and use of tag is outside the scope of this document.
- o Segment List[n]: 128 bit IPv6 addresses representing the nth segment in the Segment List. The Segment List is encoded starting from the last segment of the SR Policy. I.e., the first element of the segment list (Segment List [0]) contains the last segment of the SR Policy, the second element contains the penultimate segment of the SR Policy and so on.
- o Type Length Value (TLV) are described in Section 2.1.

In the SRH, the Next Header, Hdr Ext Len, Routing Type, and Segments Left fields are defined in Section 4.4 of [RFC8200]. Based on the constraints in that section, Next Header, Header Ext Len, and Routing Type are not mutable while Segments Left is mutable.

The mutability of the TLV value is defined by the most significant bit in the type, as specified in Section 2.1.

Section 4.3 defines the mutability of the remaining fields in the SRH (Flags, Tag, Segment List) in the context of the SID defined in this document.

New SIDs defined in the future MUST specify the mutability properties of the Flags, Tag, and Segment List and indicate how the HMAC TLV (Section 2.1.2) verification works. Note, that in effect these fields are mutable.

Consistent with the source routing model, the source of the SRH always knows how to set the segment list, Flags, Tag and TLVs of the SRH for use within the SR Domain. How it achieves this is outside the scope of this document, but may be based on topology, available SIDs and their mutability properties, the SRH mutability requirements of the destination, or any other information.

## 2.1. SRH TLVs

This section defines TLVs of the Segment Routing Header.

A TLV provides meta-data for segment processing. The only TLVs defined in this document are the HMAC (Section 2.1.2) and PAD (Section 2.1.1) TLVs. While processing the SID defined in Section 4.3.1, all TLVs are ignored unless local configuration indicates otherwise (Section 4.3.1.1.1). Thus, TLV and HMAC support is optional for any implementation, however, an implementation adding or parsing TLVs MUST support PAD TLVs. Other documents may define additional TLVs and processing rules for them.

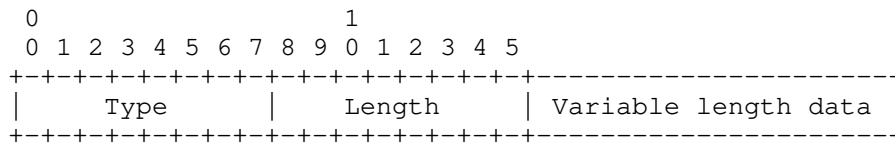
TLVs are present when the Hdr Ext Len is greater than (Last Entry+1)\*2.

While processing TLVs at a segment endpoint, TLVs MUST be fully contained within the SRH as determined by the Hdr Ext Len. Detection of TLVs exceeding the boundary of the SRH Hdr Ext Len results in an ICMP Parameter Problem, Code 0, message to the Source Address, pointing to the Hdr Ext Len field of the SRH, and the packet being discarded.

An implementation MAY limit the number and/or length of TLVs it processes based on local configuration. It MAY:

- o Limit the number of consecutive Pad1 (Section 2.1.1.1) options to 1. If padding of more than one byte is required, then PadN (Section 2.1.1.2) should be used.
- o Limit the length in PadN to 5.
- o Limit the maximum number of non-Pad TLVs to be processed.
- o Limit the maximum length of all TLVs to be processed.

The implementation MAY stop processing additional TLVs in the SRH when these configured limits are exceeded.



Type: An 8 bit codepoint from Segment Routing Header TLVs Registry TBD IANA Reference. Unrecognized Types MUST be ignored on receipt.

Length: The length of the Variable length data in bytes.

Variable length data: Length bytes of data that is specific to the Type.

Type Length Value (TLV) entries contain OPTIONAL information that may be used by the node identified in the Destination Address (DA) of the packet.

Each TLV has its own length, format and semantic. The codepoint allocated (by IANA) to each TLV Type defines both the format and the semantic of the information carried in the TLV. Multiple TLVs may be encoded in the same SRH.

The highest-order bit of the TLV type (bit 0) specifies whether or not the TLV data of that type can change en route to the packet's final destination:

0: TLV data does not change en route

1: TLV data does change en route

All TLVs specify their alignment requirements using an  $xn+y$  format. The  $xn+y$  format is defined as per [RFC8200]. The SR Source nodes use the  $xn+y$  alignment requirements of TLVs and Padding TLVs when constructing an SRH.

The "Length" field of the TLV is used to skip the TLV while inspecting the SRH in case the node doesn't support or recognize the Type. The "Length" defines the TLV length in octets, not including the "Type" and "Length" fields.

The following TLVs are defined in this document:

Padding TLVs

HMAC TLV

Additional TLVs may be defined in the future.



## 2.1.1.1. Padding TLVs

There are two types of Padding TLVs, pad1 and padN, the following applies to both:

Padding TLVs are used for meeting the alignment requirement of the subsequent TLVs.

Padding TLVs are used to pad the SRH to a multiple of 8 octets.

Padding TLVs are ignored by a node processing the SRH TLV.

Multiple Padding TLVs MAY be used in one SRH

## 2.1.1.1.1. PAD1

Alignment requirement: none

```

0 1 2 3 4 5 6 7
+---+---+---+---+
| Type |
+---+---+---+---+

```

Type: to be assigned by IANA (Suggested value 0)

A single Pad1 TLV MUST be used when a single byte of padding is required. A Pad1 TLV MUST NOT be used if more than one consecutive byte of padding is required.

## 2.1.1.1.2. PADN

Alignment requirement: none

```

0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Type | Length | Padding (variable) |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
// Padding (variable) //
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Type: to be assigned by IANA (suggested value 4).

Length: 0 to 5

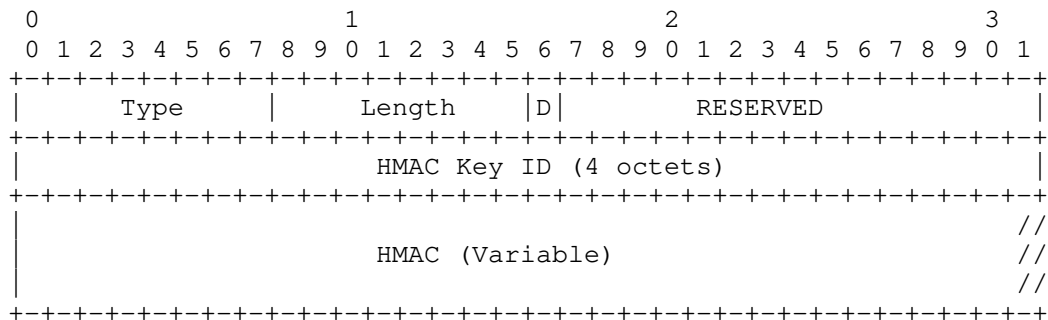
Padding: Length octets of padding. Padding bits have no semantic. They MUST be set to 0 on transmission and ignored on receipt.

The PadN TLV MUST be used when more than one byte of padding is required.

#### 2.1.1.2. HMAC TLV

Alignment requirement: 8n

The keyed Hashed Message Authentication Code (HMAC) TLV is OPTIONAL and has the following format:



where:

- o Type: to be assigned by IANA (suggested value 5).
- o Length: The length of the variable length data in bytes.
- o D: 1 bit. 1 indicates the Destination Address verification is disabled due to use of reduced segment list, Section 4.1.1.
- o RESERVED: 15 bits. MUST be 0 on transmission.
- o HMAC Key ID: A 4 octet opaque number which uniquely identifies the pre-shared key and algorithm used to generate the HMAC.
- o HMAC: Keyed HMAC, in multiples of 8 octets, at most 32 octets.

The HMAC TLV is used to verify that the SRH applied to a packet was selected by an authorized party, and to ensure that the segment list is not modified after generation. This also allows for verification that the current segment (by virtue of being in the authorized segment list) is authorized for use. The SR Domain ensures the source node is permitted to use the source address in the packet via ingress filtering mechanisms as defined in BCP 84 [RFC3704], or other strategies as appropriate.

#### 2.1.2.1. HMAC Generation and Verification

Local configuration determines when to check for an HMAC. This local configuration is outside the scope of this document. It may be based on the active segment at an SR Segment endpoint node, the result of an ACL that considers incoming interface, HMAC Key ID, or other packet fields.

An implementation that supports the generation and verification of the HMAC supports the following default behavior, as defined in the remainder of this section.

The HMAC verification begins by checking the current segment is equal to the destination address of the IPv6 header. The check is successful when either

- o HMAC D bit is 1 and Segments Left is greater than Last Entry.
- o HMAC Segments Left is less than or equal to Last Entry and destination address is equal to Segment List [Segments Left].

The HMAC field is the output of the HMAC computation as defined in [RFC2104], using:

- o key: the pre-shared key identified by HMAC Key ID
- o HMAC algorithm: identified by the HMAC Key ID
- o Text: a concatenation of the following fields from the IPv6 header and the SRH, as it would be received at the node verifying the HMAC:
  - \* IPv6 header: source address (16 octets)
  - \* SRH: Last Entry (1 octet)
  - \* SRH: Flags (1 octet)
  - \* SRH: HMAC 16 bits following Length
  - \* SRH: HMAC Key ID (4 octets)
  - \* SRH: all addresses in the Segment List (variable octets)

The HMAC digest is truncated to 32 octets and placed in the HMAC field of the HMAC TLV.

For HMAC algorithms producing digests less than 32 octets, the digest is placed in the lowest order octets of the HMAC field. Subsequent octets MUST be set to zero such that the HMAC length is a multiple of 8 octets.

If HMAC verification is successful, processing proceeds as normal.

If HMAC verification fails, an ICMP error message (parameter problem, error code 0, pointing to the HMAC TLV) SHOULD be generated (but rate limited) and SHOULD be logged and the packet discarded.

#### 2.1.2.2. HMAC Pre-Shared Key Algorithm

The HMAC Key ID field allows for the simultaneous existence of several hash algorithms (SHA-256, SHA3-256 ... or future ones) as well as pre-shared keys.

The HMAC Key ID field is opaque, i.e., it has neither syntax nor semantic except as an identifier of the right combination of pre-shared key and hash algorithm.

At the HMAC TLV generating and verification nodes, the Key ID uniquely identifies the pre-shared key and HMAC algorithm.

At the HMAC TLV generating node, the Text for the HMAC computation is set to the IPv6 header fields and SRH fields as they would appear at the verification node(s), not necessarily the same as the source node sending a packet with the HMAC TLV.

Pre-shared key roll-over is supported by having two key IDs in use while the HMAC TLV generating node and verifying node converge to a new key.

The HMAC TLV generating node may need to revoke an SRH for which it previously generated an HMAC. Revocation is achieved by allocating a new key and key ID, then rolling over the key ID associated with the SRH to be revoked. The HMAC TLV verifying node drops packets with the revoked SRH.

An implementation supporting HMAC can support multiple hash functions. An implementation supporting HMAC MUST implement SHA-2 [FIPS180-4] in its SHA-256 variant.

The selection of pre-shared key and algorithm, and their distribution is outside the scope of this document. Some options may include:

- o in the configuration of the HMAC generating or verifying nodes, either by static configuration or any SDN oriented approach

- o dynamically using a trusted key distribution protocol such as [RFC6407]

While key management is outside the scope of this document, the recommendations of BCP 107 [RFC4107] should be considered when choosing the key management system.

### 3. SR Nodes

There are different types of nodes that may be involved in segment routing networks: source SR nodes originate packets with a segment in the destination address of the IPv6 header, transit nodes that forward packets destined to a remote segment, and SR segment endpoint nodes that process a local segment in the destination address of an IPv6 header.

#### 3.1. Source SR Node

A Source SR Node is any node that originates an IPv6 packet with a segment (i.e. SRv6 SID) in the destination address of the IPv6 header. The packet leaving the source SR Node may or may not contain an SRH. This includes either:

A host originating an IPv6 packet.

An SR domain ingress router encapsulating a received packet in an outer IPv6 header, followed by an optional SRH.

The mechanism through which a segment in the destination address of the IPv6 header and the Segment List in the SRH, is derived is outside the scope of this document.

#### 3.2. Transit Node

A transit node is any node forwarding an IPv6 packet where the destination address of that packet is not locally configured as a segment nor a local interface. A transit node is not required to be capable of processing a segment nor SRH.

#### 3.3. SR Segment Endpoint Node

A SR segment endpoint node is any node receiving an IPv6 packet where the destination address of that packet is locally configured as a segment or local interface.

#### 4. Packet Processing

This section describes SRv6 packet processing at the SR source, Transit and SR segment endpoint nodes.

##### 4.1. Source SR Node

A Source node steers a packet into an SR Policy. If the SR Policy results in a segment list containing a single segment, and there is no need to add information to the SRH flag or to add TLV, the DA is set to the single segment list entry and the SRH MAY be omitted.

When needed, the SRH is created as follows:

Next Header and Hdr Ext Len fields are set as specified in [RFC8200].

Routing Type field is set as TBD (to be allocated by IANA, suggested value 4).

The DA of the packet is set with the value of the first segment.

The first element of the SRH Segment List is the ultimate segment. The second element is the penultimate segment, and so on.

The Segments Left field is set to  $n-1$  where  $n$  is the number of elements in the SR Policy.

The Last Entry field is set to  $n-1$  where  $n$  is the number of elements in the SR Policy.

TLVs (including HMAC) may be set according to their specification.

The packet is forwarded toward the packet's Destination Address (the first segment).

##### 4.1.1. Reduced SRH

When a source does not require the entire SID list to be preserved in the SRH, a reduced SRH may be used.

A reduced SRH does not contain the first segment of the related SR Policy (the first segment is the one already in the DA of the IPv6 header), and the Last Entry field is set to  $n-2$  where  $n$  is the number of elements in the SR Policy.

#### 4.2. Transit Node

As specified in [RFC8200], the only node allowed to inspect the Routing Extension Header (and therefore the SRH), is the node corresponding to the DA of the packet. Any other transit node MUST NOT inspect the underneath routing header and MUST forward the packet toward the DA according to its IPv6 routing table.

When a SID is in the destination address of an IPv6 header of a packet, it's routed through an IPv6 network as an IPv6 address. SIDs, or the prefix(es) covering SIDs, and their reachability may be distributed by means outside the scope of this document. For example, [RFC5308] or [RFC5340] may be used to advertise a prefix covering the SIDs on a node.

#### 4.3. SR Segment Endpoint Node

Without constraining the details of an implementation, the SR segment endpoint node creates Forwarding Information Base (FIB) entries for its local SIDs.

When an SRv6-capable node receives an IPv6 packet, it performs a longest-prefix-match lookup on the packets destination address. This lookup can return any of the following:

- \* A FIB entry that represents a locally instantiated SRv6 SID
- \* A FIB entry that represents a local interface, not locally instantiated as an SRv6 SID
- \* A FIB entry that represents a non-local route
- \* No Match

##### 4.3.1. FIB Entry Is Locally Instantiated SRv6 SID

This document, and section, defines a single SRv6 SID. Future documents may define additional SRv6 SIDs. In which case, the entire content of this section will be defined in that document.

If the FIB entry represents a locally instantiated SRv6 SID, process the next header chain of the IPv6 header as defined in section 4 of [RFC8200]. Section 4.3.1.1 describes how to process an SRH, Section 4.3.1.2 describes how to process an upper layer header or no next header.

Processing this SID modifies the Segments Left and, if configured to process TLVs, it may modify the "variable length data" of TLV types that change en route. Therefore Segments Left is mutable and TLVs that change en route are mutable. The remainder of the SRH (Flags,

Tag, Segment List, and TLVs that do not change en route) are immutable while processing this SID.

#### 4.3.1.1. SRH Processing

```
S01. When an SRH is processed {
S02. If Segments Left is equal to zero {
S03. Proceed to process the next header in the packet,
 whose type is identified by the Next Header field in
 the Routing header.
S04. }
S05. Else {
S06. If local configuration requires TLV processing {
S07. Perform TLV processing (see TLV Processing)
S08. }
S09. max_last_entry = (Hdr Ext Len / 2) - 1
S10. If ((Last Entry > max_last_entry) or
S11. (Segments Left is greater than (Last Entry+1)) {
S12. Send an ICMP Parameter Problem, Code 0, message to
 the Source Address, pointing to the Segments Left
 field, and discard the packet.
S13. }
S14. Else {
S15. Decrement Segments Left by 1.
S16. Copy Segment List[Segments Left] from the SRH to the
 destination address of the IPv6 header.
S17. If the IPv6 Hop Limit is less than or equal to 1 {
S18. Send an ICMP Time Exceeded -- Hop Limit Exceeded in
 Transit message to the Source Address and discard
 the packet.
S19. }
S20. Else {
S21. Decrement the Hop Limit by 1
S22. Resubmit the packet to the IPv6 module for transmission
 to the new destination.
S23. }
S24. }
S25. }
S26. }
```

#### 4.3.1.1.1. TLV Processing

Local configuration determines how TLVs are to be processed when the Active Segment is a local SID defined in this document. The definition of local configuration is outside the scope of this document.



For illustration purpose only, two example local configurations that may be associated with a SID are provided below.

Example 1:

For any packet received from interface I2  
    Skip TLV processing

Example 2:

For any packet received from interface I1  
    If first TLV is HMAC {  
        Process the HMAC TLV  
    }  
    Else {  
        Discard the packet  
    }

#### 4.3.1.2. Upper-layer Header or No Next Header

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 SID defined in this document.

```
IF (Upper-layer Header is IPv4 or IPv6) and
 local configuration permits {
 Perform IPv6 decapsulation
 Resubmit the decapsulated packet to the IPv4 or IPv6 module
}
ELSE {
 Send an ICMP parameter problem message to the Source Address and
 discard the packet. Error code (TBD by IANA) "SR Upper-layer
 Header Error", pointer set to the offset of the upper-layer
 header.
}
```

A unique error code allows an SR Source node to recognize an error in SID processing at an endpoint.

#### 4.3.2. FIB Entry Is A Local Interface

If the FIB entry represents a local interface, not locally instantiated as an SRv6 SID, the SRH is processed as follows:

    If Segments Left is zero, the node must ignore the Routing header and proceed to process the next header in the packet, whose type is identified by the Next Header field in the Routing Header.

    If Segments Left is non-zero, the node must discard the packet and send an ICMP Parameter Problem, Code 0, message to the packet's Source Address, pointing to the unrecognized Routing Type.

#### 4.3.3. FIB Entry Is A Non-Local Route

Processing is not changed by this document.

#### 4.3.4. FIB Entry Is A No Match

Processing is not changed by this document.

### 5. Intra SR Domain Deployment Model

The use of the SIDs exclusively within the SR Domain and solely for packets of the SR Domain is an important deployment model.

This enables the SR Domain to act as a single routing system.

This section covers:

- o securing the SR Domain from external attempt to use its SIDs
- o SR Domain as a single system with delegation between components
- o handling packets of the SR Domain

#### 5.1. Securing the SR Domain

Nodes outside the SR Domain are not trusted: they cannot directly use the SIDs of the domain. This is enforced by two levels of access control lists:

1. Any packet entering the SR Domain and destined to a SID within the SR Domain is dropped. This may be realized with the following logic. Other methods with equivalent outcome are considered compliant:
  - \* allocate all the SID's from a block S/s
  - \* configure each external interface of each edge node of the domain with an inbound infrastructure access list (IACL) which drops any incoming packet with a destination address in S/s
  - \* Failure to implement this method of ingress filtering exposes the SR Domain to source routing attacks as described and referenced in [RFC5095]
2. The distributed protection in #1 is complemented with per node protection, dropping packets to SIDs from source addresses outside the SR Domain. This may be realized with the following

logic. Other methods with equivalent outcome are considered compliant:

- \* assign all interface addresses from prefix A/a
- \* at node k, all SIDs local to k are assigned from prefix Sk/sk
- \* configure each internal interface of each SR node k in the SR Domain with an inbound IACL which drops any incoming packet with a destination address in Sk/sk if the source address is not in A/a.

## 5.2. SR Domain as A Single System with Delegation Among Components

All intra SR Domain packets are of the SR Domain. The IPv6 header is originated by a node of the SR Domain, and is destined to a node of the SR Domain.

All inter domain packets are encapsulated for the part of the packet journey that is within the SR Domain. The outer IPv6 header is originated by a node of the SR Domain, and is destined to a node of the SR Domain.

As a consequence, any packet within the SR Domain is of the SR Domain.

The SR Domain is a system in which the operator may want to distribute or delegate different operations of the outer most header to different nodes within the system.

An operator of an SR domain may choose to delegate SRH addition to a host node within the SR domain, and validation of the contents of any SRH to a more trusted router or switch attached to the host. Consider a top of rack switch (T) connected to host (H) via interface (I). H receives an SRH (SRH1) with a computed HMAC via some SDN method outside the scope of this document. H classifies traffic it sources and adds SRH1 to traffic requiring a specific SLA. T is configured with an IACL on I requiring verification of the SRH for any packet destined to the SID block of the SR Domain (S/s). T checks and verifies that SRH1 is valid, contains an HMAC TLV and verifies the HMAC.

An operator of the SR Domain may choose to have all segments in the SR Domain verify the HMAC. This mechanism would verify that the SRH segment list is not modified while traversing the SR Domain.

### 5.3. MTU Considerations

An SR Domain ingress edge node encapsulates packets traversing the SR Domain, and needs to consider the MTU of the SR Domain. Within the SR Domain, well known mitigation techniques are RECOMMENDED, such as deploying a greater MTU value within the SR Domain than at the ingress edges.

Encapsulation with an outer IPv6 header and SRH share the same MTU and fragmentation considerations as IPv6 tunnels described in [RFC2473]. Further investigation on the limitation of various tunneling methods (including IPv6 tunnels) are discussed in [I-D.ietf-intarea-tunnels] and SHOULD be considered by operators when considering MTU within the SR Domain.

### 5.4. ICMP Error Processing

ICMP error packets generated within the SR Domain are sent to source nodes within the SR Domain. The invoking packet in the ICMP error message may contain an SRH. Since the destination address of a packet with an SRH changes as each segment is processed, it may not be the destination used by the socket or application that generated the invoking packet.

For the source of an invoking packet to process the ICMP error message, the ultimate destination address of the IPv6 header may be required. The following logic is used to determine the destination address for use by protocol error handlers.

- o Walk all extension headers of the invoking IPv6 packet to the routing extension header preceding the upper layer header.
  - \* If routing header is type TBD IANA (SRH)
    - + The SID at Segment List[0] may be used as the destination address of the invoking packet.

ICMP errors are then processed by upper layer transports as defined in [RFC4443].

For IP packets encapsulated in an outer IPv6 header, ICMP error handling is as defined in [RFC2473].

### 5.5. Load Balancing and ECMP

For any inter domain packet, the SR Source node MUST impose a flow label computed based on the inner packet. The computation of the

flow label is as recommended in [RFC6438] for the sending Tunnel End Point.

For any intra domain packet, the SR Source node SHOULD impose a flow label computed as described in [RFC6437] to assist ECMP load balancing at transit nodes incapable of computing a 5-tuple beyond the SRH.

At any transit node within an SR domain, the flow label MUST be used as defined in [RFC6438] to calculate the ECMP hash toward the destination address. If flow label is not used, the transit node would likely hash all packets between a pair of SR Edge nodes to the same link.

At an SR segment endpoint node, the flow label MUST be used as defined in [RFC6438] to calculate any ECMP hash used to forward the processed packet to the next segment.

## 5.6. Other Deployments

Other deployment models and their implications on security, MTU, HMAC, ICMP error processing and interaction with other extension headers are outside the scope of this document.

## 6. Illustrations

This section provides illustrations of SRv6 packet processing at SR source, transit and SR segment endpoint nodes.

### 6.1. Abstract Representation of an SRH

For a node  $k$ , its IPv6 address is represented as  $A_k$ , its SRv6 SID is represented as  $S_k$ .

IPv6 headers are represented as the tuple of (source, destination). For example, a packet with source address  $A_1$  and destination address  $A_2$  is represented as  $(A_1, A_2)$ . The payload of the packet is omitted.

An SR Policy is a list of segments. A list of segments is represented as  $\langle S_1, S_2, S_3 \rangle$  where  $S_1$  is the first SID to visit,  $S_2$  is the second SID to visit and  $S_3$  is the last SID to visit.

$(SA, DA) (S_3, S_2, S_1; SL)$  represents an IPv6 packet with:

- o Source Address is SA, Destination Addresses is DA, and next-header is SRH.
- o SRH with SID list  $\langle S_1, S_2, S_3 \rangle$  with SegmentsLeft = SL.

- o Note the difference between the <> and () symbols. <S1, S2, S3> represents a SID list where the leftmost segment is the first segment. Whereas, (S3, S2, S1; SL) represents the same SID list but encoded in the SRH Segment List format where the leftmost segment is the last segment. When referring to an SR policy in a high-level use-case, it is simpler to use the <S1, S2, S3> notation. When referring to an illustration of detailed behavior, the (S3, S2, S1; SL) notation is more convenient.

At its SR Policy headend, the Segment List <S1,S2,S3> results in SRH (S3,S2,S1; SL=2) represented fully as:

```
Segments Left=2
Last Entry=2
Flags=0
Tag=0
Segment List[0]=S3
Segment List[1]=S2
Segment List[2]=S1
```

## 6.2. Example Topology

The following topology is used in examples below:

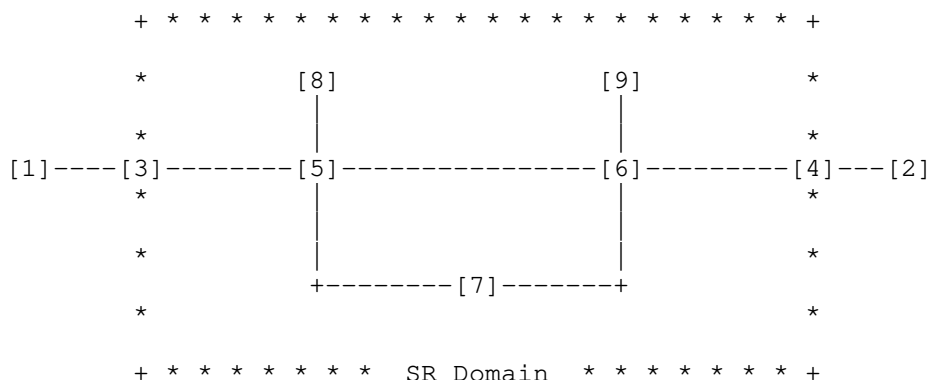


Figure 3

- o 3 and 4 are SR Domain edge routers
- o 5, 6, and 7 are all SR Domain routers
- o 8 and 9 are hosts within the SR Domain
- o 1 and 2 are hosts outside the SR Domain

- o The SR domain implements ingress filtering as per Section 5.1 and no external packet can enter the domain with a destination address equal to a segment of the domain.

### 6.3. Source SR Node

#### 6.3.1. Intra SR Domain Packet

When host 8 sends a packet to host 9 via an SR Policy <S7,A9> the packet is

P1: (A8,S7) (A9,S7; SL=1)

##### 6.3.1.1. Reduced Variant

When host 8 sends a packet to host 9 via an SR Policy <S7,A9> and it wants to use a reduced SRH, the packet is

P2: (A8,S7) (A9; SL=1)

#### 6.3.2. Inter SR Domain Packet - Transit

When host 1 sends a packet to host 2, the packet is

P3: (A1,A2)

The SR Domain ingress router 3 receives P3 and steers it to SR Domain egress router 4 via an SR Policy <S7, S4>. Router 3 encapsulates the received packet P3 in an outer header with an SRH. The packet is

P4: (A3, S7) (S4, S7; SL=1) (A1, A2)

If the SR Policy contains only one segment (the egress router 4), the ingress Router 3 encapsulates P3 into an outer header (A3, S4) without SRH. The packet is

P5: (A3, S4) (A1, A2)

##### 6.3.2.1. Reduced Variant

The SR Domain ingress router 3 receives P3 and steers it to SR Domain egress router 4 via an SR Policy <S7, S4>. If router 3 wants to use a reduced SRH, Router 3 encapsulates the received packet P3 in an outer header with a reduced SRH. The packet is

P6: (A3, S7) (S4; SL=1) (A1, A2)

### 6.3.3. Inter SR Domain Packet - Internal to External

When host 8 sends a packet to host 1, the packet is encapsulated for the portion of its journey within the SR Domain. From 8 to 3 the packet is

P7: (A8,S3) (A8,A1)

In the opposite direction, the packet generated from 1 to 8 is

P8: (A1,A8)

At node 3 P8 is encapsulated for the portion of its journey within the SR domain, with the outer header destined to segment S8. Resulting in

P9: (A3,S8) (A1,A8)

At node 8 the outer IPv6 header is removed by S8 processing, then processed again when received by A8.

### 6.4. Transit Node

Nodes 5 acts as transit nodes for packet P1, and sends packet

P1: (A8,S7) (A9,S7;SL=1)

on the interface toward node 7.

### 6.5. SR Segment Endpoint Node

Node 7 receives packet P1 and, using the logic in Section 4.3.1, sends packet

P7: (A8,A9) (A9,S7; SL=0)

on the interface toward router 6.

### 6.6. Delegation of Function with HMAC Verification

This section describes how a function may be delegated within the SR Domain. In the following sections consider a host 8 connected to a top of rack 5.



#### 6.6.1. SID List Verification

An operator may prefer to apply the SRH at source 8, while 5 verifies the SID list is valid.

For illustration purpose, an SDN controller provides 8 an SRH terminating at node 9, with segment list <S5,S7,S6,A9>, and HMAC TLV computed for the SRH. The HMAC key ID and key associated with the HMAC TLV is shared with 5. Node 8 does not know the key. Node 5 is configured with an IACL applied to the interface connected to 8, requiring HMAC verification for any packet destined to S/s.

Node 8 originates packets with the received SRH including HMAC TLV.

P15: (A8,S5) (A9,S6,S7,S5;SL=3;HMAC)

Node 5 receives and verifies the HMAC for the SRH, then forwards the packet to the next segment

P16: (A8,S7) (A9,S6,S7,S5;SL=2;HMAC)

Node 6 receives

P17: (A8,S6) (A9,S6,S7,S5;SL=1;HMAC)

Node 9 receives

P18: (A8,A9) (A9,S6,S7,S5;SL=0;HMAC)

This use of an HMAC is particularly valuable within an enterprise based SR Domain [SRN].

### 7. Security Considerations

This section reviews security considerations related to the SRH, given the SRH processing and deployment models discussed in this document.

As described in Section 5, it is necessary to filter packets ingress to the SR Domain, destined to SIDs within the SR Domain (i.e., bearing a SID in the destination address). This ingress filtering is via an IACL at SR Domain ingress border nodes. Additional protection is applied via an IACL at each SR Segment Endpoint node, filtering packets not from within the SR Domain, destined to SIDs in the SR Domain. ACLs are easily supported for small numbers of prefixes, making summarization important, and when the prefixes requiring filtering is kept to a seldom changing set.

Additionally, ingress filtering of IPv6 source addresses as recommended in BCP38 [RFC2827] SHOULD be used.

#### 7.1. Source Routing Attacks

An SR domain implements distributed and per node protection as described in section 5.1. Additionally, domains deny traffic with spoofed addresses by implementing the recommendations in BCP 84 [RFC3704].

Full implementation of the recommended protection blocks the attacks documented in [RFC5095] from outside the SR domain, including bypassing filtering devices, reaching otherwise unreachable Internet systems, network topology discovery, bandwidth exhaustion, and defeating anycast.

Failure to implement distributed and per node protection allows attackers to bypass filtering devices and exposes the SR Domain to these attacks.

Compromised nodes within the SR Domain may mount the attacks listed above along with other known attacks on IP networks (e.g. DOS/DDOS, topology discovery, man-in-the-middle, traffic interception/siphoning).

#### 7.2. Service Theft

Service theft is defined as the use of a service offered by the SR Domain by a node not authorized to use the service.

Service theft is not a concern within the SR Domain as all SR Source nodes and SR segment endpoint nodes within the domain are able to utilize the services of the Domain. If a node outside the SR Domain learns of segments or a topological service within the SR domain, IACL filtering denies access to those segments.

#### 7.3. Topology Disclosure

The SRH is unencrypted and may contain SIDs of some intermediate SR-nodes in the path towards the destination within the SR Domain. If packets can be snooped within the SR Domain, the SRH may reveal topology, traffic flows, and service usage.

This is applicable within an SR Domain, but the disclosure is less relevant as an attacker has other means of learning topology, flows, and service usage.

#### 7.4. ICMP Generation

The generation of ICMPv6 error messages may be used to attempt denial-of-service attacks by sending an error-causing destination address or SRH in back-to-back packets. An implementation that correctly follows Section 2.4 of [RFC4443] would be protected by the ICMPv6 rate-limiting mechanism.

#### 7.5. Applicability of AH

The SR Domain is a trusted domain, as defined in [RFC8402] Section 2 and Section 8.2. The SR Source is trusted to add an SRH (optionally verified as having been generated by a trusted source via the HMAC TLV in this document), and segments advertised within the domain are trusted to be accurate and advertised by trusted sources via a secure control plane. As such the SR Domain does not rely on the Authentication Header (AH) as defined in [RFC4302] to secure the SRH.

The use of SRH with AH by an SR source node, and processing at a SR segment endpoint node is not defined in this document. Future documents may define use of SRH with AH and its processing.

#### 8. IANA Considerations

This document makes the following registrations in the Internet Protocol Version 6 (IPv6) Parameters "Routing Type" registry maintained by IANA:

| Suggested Value | Description                  | Reference     |
|-----------------|------------------------------|---------------|
| 4               | Segment Routing Header (SRH) | This document |

This document makes the following registrations in "Type 4 - Parameter Problem" message of the "Internet Control Message Protocol version 6 (ICMPv6) Parameters" registry maintained by IANA:

| CODE     | NAME/DESCRIPTION            |
|----------|-----------------------------|
| TBD IANA | SR Upper-layer Header Error |

This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding registration of values related to the SRH, in accordance with BCP 26, [RFC8126].

The following terms are used here with the meanings defined in BCP 26: "namespace", "assigned value", "registration".

The following policies are used here with the meanings defined in BCP 26 [RFC8126]: "IETF Review".

#### 8.1. Segment Routing Header Flags Registry

This document requests the creation of a new IANA managed registry to identify SRH Flags Bits. The registration procedure is "IETF Review". Suggested registry name is "Segment Routing Header Flags". Flags is 8 bits.

#### 8.2. Segment Routing Header TLVs Registry

This document requests the creation of a new IANA managed registry to identify SRH TLVs. The registration procedure is "IETF Review". Suggested registry name is "Segment Routing Header TLVs". A TLV is identified through an unsigned 8 bit codepoint value, with assigned values 0-127 for TLVs that do not change en route, and 128-255 for TLVs that may change en route. The following codepoints are defined in this document:

| Assigned Value | Description              | Reference     |
|----------------|--------------------------|---------------|
| 0              | Pad1 TLV                 | This document |
| 1              | Reserved                 | This document |
| 2              | Reserved                 | This document |
| 3              | Reserved                 | This document |
| 4              | PadN TLV                 | This document |
| 5              | HMAC TLV                 | This document |
| 6              | Reserved                 | This document |
| 124-126        | Experimentation and Test | This document |
| 127            | Reserved                 | This document |
| 252-254        | Experimentation and Test | This document |
| 255            | Reserved                 | This document |

Values 1,2,3,6 were defined in draft versions of this specification and are Reserved for backwards compatibility with early implementations and should not be reassigned. Values 127 and 255 are Reserved to allow for expansion of the Type field in future specifications if needed.

#### 9. Implementation Status

This section is to be removed prior to publishing as an RFC.

See [I-D.matsushima-spring-srv6-deployment-status] for updated deployment and interoperability reports.

### 9.1. Linux

Name: Linux Kernel v4.14

Status: Production

Implementation: adds SRH, performs END processing, supports HMAC TLV

Details: <https://irtf.org/anrw/2017/anrw17-final3.pdf> and  
[I-D.filsfils-spring-srv6-interop]

### 9.2. Cisco Systems

Name: IOS XR and IOS XE

Status: Production (IOS XR), Pre-production (IOS XE)

Implementation: adds SRH, performs END processing, no TLV processing

Details: [I-D.filsfils-spring-srv6-interop]

### 9.3. FD.io

Name: VPP/Segment Routing for IPv6

Status: Production

Implementation: adds SRH, performs END processing, no TLV processing

Details: [https://wiki.fd.io/view/VPP/Segment\\_Routing\\_for\\_IPv6](https://wiki.fd.io/view/VPP/Segment_Routing_for_IPv6) and  
[I-D.filsfils-spring-srv6-interop]

### 9.4. Barefoot

Name: Barefoot Networks Tofino NPU

Status: Prototype

Implementation: performs END processing, no TLV processing

Details: [I-D.filsfils-spring-srv6-interop]

### 9.5. Juniper

Name: Juniper Networks Trio and vTrio NPU's

Status: Prototype & Experimental

Implementation: SRH insertion mode, Process SID where SID is an interface address, no TLV processing

#### 9.6. Huawei

Name: Huawei Systems VRP Platform

Status: Production

Implementation: adds SRH, performs END processing, no TLV processing

#### 10. Contributors

Kamran Raza, Zafar Ali, Brian Field, Daniel Bernier, Ida Leung, Jen Linkova, Ebben Aries, Tomoya Kosugi, Eric Vyncke, David Lebrun, Dirk Steinberg, Robert Raszuk, Dave Barach, John Brzozowski, Pierre Francois, Nagendra Kumar, Mark Townsley, Christian Martin, Roberta Maglione, James Connolly, Aloys Augustin contributed to the content of this document.

#### 11. Acknowledgements

The authors would like to thank Ole Troan, Bob Hinden, Ron Bonica, Fred Baker, Brian Carpenter, Alexandru Petrescu, Punit Kumar Jaiswal, David Lebrun, Benjamin Kaduk, Frank Xialiang, Mirja Kuhlewind, Roman Danyliw, Joe Touch, and Magnus Westerlund for their comments to this document.

#### 12. References

##### 12.1. Normative References

[FIPS180-4]

National Institute of Standards and Technology, "FIPS 180-4 Secure Hash Standard (SHS)", March 2012, <<http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>>.

[RFC2104]

Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <<https://www.rfc-editor.org/info/rfc2104>>.

[RFC2119]

Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <<https://www.rfc-editor.org/info/rfc2473>>.
- [RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <<https://www.rfc-editor.org/info/rfc2827>>.
- [RFC3704] Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks", BCP 84, RFC 3704, DOI 10.17487/RFC3704, March 2004, <<https://www.rfc-editor.org/info/rfc3704>>.
- [RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, DOI 10.17487/RFC4107, June 2005, <<https://www.rfc-editor.org/info/rfc4107>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<https://www.rfc-editor.org/info/rfc4302>>.
- [RFC5095] Abley, J., Savola, P., and G. Neville-Neil, "Deprecation of Type 0 Routing Headers in IPv6", RFC 5095, DOI 10.17487/RFC5095, December 2007, <<https://www.rfc-editor.org/info/rfc5095>>.
- [RFC6407] Weis, B., Rowles, S., and T. Hardjono, "The Group Domain of Interpretation", RFC 6407, DOI 10.17487/RFC6407, October 2011, <<https://www.rfc-editor.org/info/rfc6407>>.
- [RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <<https://www.rfc-editor.org/info/rfc6437>>.
- [RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <<https://www.rfc-editor.org/info/rfc6438>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.

- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

## 12.2. Informative References

- [I-D.filsfils-spring-srv6-interop]  
Filsfils, C., Clad, F., Camarillo, P., Abdelsalam, A., Salsano, S., Bonaventure, O., Horn, J., and J. Liste, "SRv6 interoperability report", draft-filsfils-spring-srv6-interop-02 (work in progress), March 2019.
- [I-D.ietf-intarea-tunnels]  
Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", draft-ietf-intarea-tunnels-10 (work in progress), September 2019.
- [I-D.matsushima-spring-srv6-deployment-status]  
Matsushima, S., Filsfils, C., Ali, Z., and Z. Li, "SRv6 Implementation and Deployment Status", draft-matsushima-spring-srv6-deployment-status-02 (work in progress), October 2019.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC5308] Hopps, C., "Routing IPv6 with IS-IS", RFC 5308, DOI 10.17487/RFC5308, October 2008, <<https://www.rfc-editor.org/info/rfc5308>>.
- [RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, DOI 10.17487/RFC5340, July 2008, <<https://www.rfc-editor.org/info/rfc5340>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [SRN] Lebrun, D., Jadin, M., Clad, F., Filsfils, C., and O. Bonaventure, "Software Resolved Networks: Rethinking Enterprise Networks with IPv6 Segment Routing", 2018, <<https://inl.info.ucl.ac.be/system/files/sosr18-final15-embedfonts.pdf>>.



## Authors' Addresses

Clarence Filsfils (editor)  
Cisco Systems, Inc.  
Brussels  
BE

Email: cfilsfil@cisco.com

Darren Dukes (editor)  
Cisco Systems, Inc.  
Ottawa  
CA

Email: ddukes@cisco.com

Stefano Previdi  
Huawei  
Italy

Email: stefano@previdi.net

John Leddy  
Individual  
US

Email: john@leddy.net

Satoru Matsushima  
Softbank

Email: satoru.matsushima@g.softbank.co.jp

Daniel Voyer  
Bell Canada

Email: daniel.voyer@bell.ca

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

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

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

Abstract

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

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 3, 2020.

## Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                                                                                      |    |
|--------------------------------------------------------------------------------------|----|
| 1. Introduction . . . . .                                                            | 3  |
| 1.1. Specification of Requirements . . . . .                                         | 4  |
| 2. Terminology . . . . .                                                             | 4  |
| 3. Provisioning Domain Identification using Router Advertisements . . . . .          | 5  |
| 3.1. PvD ID Option for Router Advertisements . . . . .                               | 5  |
| 3.2. Router Behavior . . . . .                                                       | 8  |
| 3.3. Non-PvD-aware Host Behavior . . . . .                                           | 9  |
| 3.4. PvD-aware Host Behavior . . . . .                                               | 9  |
| 3.4.1. DHCPv6 configuration association . . . . .                                    | 10 |
| 3.4.2. DHCPv4 configuration association . . . . .                                    | 11 |
| 3.4.3. Connection Sharing by the Host . . . . .                                      | 11 |
| 3.4.4. Usage of DNS Servers . . . . .                                                | 12 |
| 4. Provisioning Domain Additional Information . . . . .                              | 13 |
| 4.1. Retrieving the PvD Additional Information . . . . .                             | 13 |
| 4.2. Operational Consideration to Providing the PvD Additional Information . . . . . | 16 |
| 4.3. PvD Additional Information Format . . . . .                                     | 16 |
| 4.3.1. Example . . . . .                                                             | 18 |
| 4.4. Detecting misconfiguration and misuse . . . . .                                 | 18 |
| 5. Operational Considerations . . . . .                                              | 19 |
| 5.1. Exposing Extra RA Options to PvD-Aware Hosts . . . . .                          | 19 |
| 5.2. Different RAs for PvD-Aware and Non-PvD-Aware Hosts . . . . .                   | 19 |
| 5.3. Enabling Multi-homing for PvD-Aware Hosts . . . . .                             | 21 |
| 5.4. Providing Additional Information to PvD-Aware Hosts . . . . .                   | 22 |
| 6. Security Considerations . . . . .                                                 | 23 |
| 7. Privacy Considerations . . . . .                                                  | 24 |
| 8. IANA Considerations . . . . .                                                     | 25 |
| 8.1. New entry in the Well-Known URIs Registry . . . . .                             | 26 |
| 8.2. Additional Information PvD Keys Registry . . . . .                              | 26 |
| 8.3. PvD Option Flags Registry . . . . .                                             | 26 |

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

## 1. Introduction

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

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

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

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

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

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

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

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

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

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

### 1.1. Specification of Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 2. Terminology

This document uses the following terminology:

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

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

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

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

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

### 3. Provisioning Domain Identification using Router Advertisements

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

#### 3.1. PvD ID Option for Router Advertisements

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

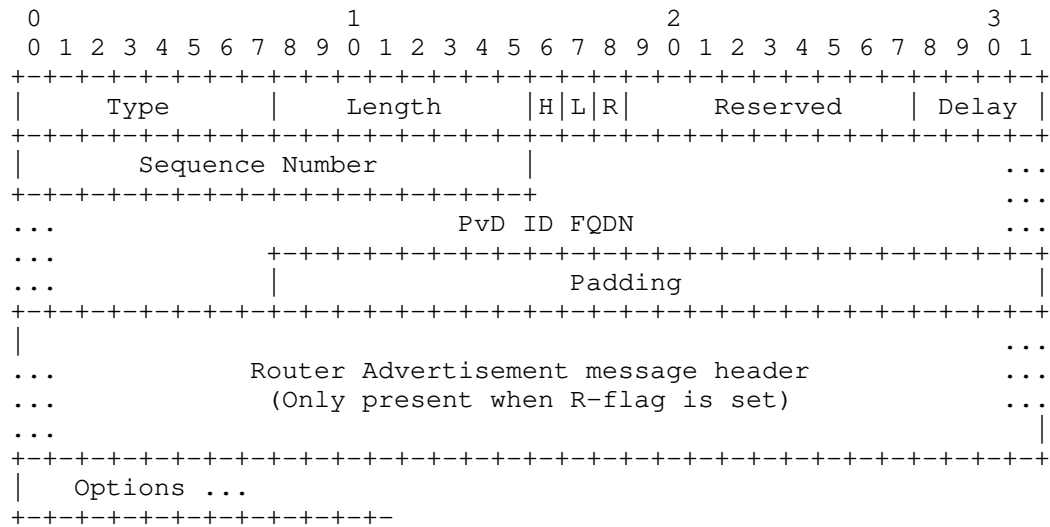


Figure 1: PvD ID Router Advertisements Option Format

Type: (8 bits) Set to 21.

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

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

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

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

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

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

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

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

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

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

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

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



| 0                                              |   |   |   |   |   |   |   |   |   | 1           |   |   |   |   |   |   |   |   |   | 2              |   |   |   |   |   |   |   |   |   | 3           |   |  |  |  |  |  |  |  |  |
|------------------------------------------------|---|---|---|---|---|---|---|---|---|-------------|---|---|---|---|---|---|---|---|---|----------------|---|---|---|---|---|---|---|---|---|-------------|---|--|--|--|--|--|--|--|--|
| 0                                              | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0           | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0              | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0           | 1 |  |  |  |  |  |  |  |  |
| Type: 21                                       |   |   |   |   |   |   |   |   |   | Length: 12  |   |   |   |   |   |   |   |   |   | 1 0 0 Reserved |   |   |   |   |   |   |   |   |   | Delay:1     |   |  |  |  |  |  |  |  |  |
| Seq number: 123                                |   |   |   |   |   |   |   |   |   | 7           |   |   |   |   |   |   |   |   |   | e              |   |   |   |   |   |   |   |   |   |             |   |  |  |  |  |  |  |  |  |
| x                                              |   |   |   |   |   |   |   |   |   | a           |   |   |   |   |   |   |   |   |   | m              |   |   |   |   |   |   |   |   |   | p           |   |  |  |  |  |  |  |  |  |
| l                                              |   |   |   |   |   |   |   |   |   | e           |   |   |   |   |   |   |   |   |   | 3              |   |   |   |   |   |   |   |   |   | o           |   |  |  |  |  |  |  |  |  |
| r                                              |   |   |   |   |   |   |   |   |   | g           |   |   |   |   |   |   |   |   |   | 0              |   |   |   |   |   |   |   |   |   | 0 (padding) |   |  |  |  |  |  |  |  |  |
| 0 (padding)                                    |   |   |   |   |   |   |   |   |   | 0 (padding) |   |   |   |   |   |   |   |   |   | 0 (padding)    |   |   |   |   |   |   |   |   |   | 0 (padding) |   |  |  |  |  |  |  |  |  |
| RDNSS option (RFC 8106) length: 5              |   |   |   |   |   |   |   |   |   |             |   |   |   |   |   |   |   |   |   |                |   |   |   |   |   |   |   |   |   | ...         |   |  |  |  |  |  |  |  |  |
| ...                                            |   |   |   |   |   |   |   |   |   |             |   |   |   |   |   |   |   |   |   |                |   |   |   |   |   |   |   |   |   | ...         |   |  |  |  |  |  |  |  |  |
| ...                                            |   |   |   |   |   |   |   |   |   |             |   |   |   |   |   |   |   |   |   |                |   |   |   |   |   |   |   |   |   |             |   |  |  |  |  |  |  |  |  |
| Prefix Information Option (RFC 4861) length: 4 |   |   |   |   |   |   |   |   |   |             |   |   |   |   |   |   |   |   |   |                |   |   |   |   |   |   |   |   |   | ...         |   |  |  |  |  |  |  |  |  |
| ...                                            |   |   |   |   |   |   |   |   |   |             |   |   |   |   |   |   |   |   |   |                |   |   |   |   |   |   |   |   |   |             |   |  |  |  |  |  |  |  |  |
| ...                                            |   |   |   |   |   |   |   |   |   |             |   |   |   |   |   |   |   |   |   |                |   |   |   |   |   |   |   |   |   |             |   |  |  |  |  |  |  |  |  |

Figure 2

### 3.2. Router Behavior

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

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

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

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

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

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

### 3.3. Non-PvD-aware Host Behavior

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

### 3.4. PvD-aware Host Behavior

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

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

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

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

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

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

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

#### 3.4.1. DHCPv6 configuration association

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

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

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

### 3.4.2. DHCPv4 configuration association

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

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

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

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

### 3.4.3. Connection Sharing by the Host

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

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

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

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

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

#### 3.4.4. Usage of DNS Servers

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

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

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

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

#### 4. Provisioning Domain Additional Information

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

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

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

##### 4.1. Retrieving the PvD Additional Information

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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



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

#### 4.2. Operational Consideration to Providing the PvD Additional Information

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

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

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

#### 4.3. PvD Additional Information Format

The PvD Additional Information is a JSON object.

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

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

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

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

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

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

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

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

#### 4.3.1. Example

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

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

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

#### 4.4. Detecting misconfiguration and misuse

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

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

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

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

## 5. Operational Considerations

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

### 5.1. Exposing Extra RA Options to PvD-Aware Hosts

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

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

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

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

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

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

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

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

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

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

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

### 5.3. Enabling Multi-homing for PvD-Aware Hosts

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

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

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

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

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

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

#### 5.4. Providing Additional Information to PvD-Aware Hosts

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

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

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

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

The HTTP server will respond with the JSON additional information:

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

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

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

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

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

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

## 6. Security Considerations

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

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

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

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

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

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



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

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

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

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

## 7. Privacy Considerations

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

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

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

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

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

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

## 8. IANA Considerations

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

### 8.1. New entry in the Well-Known URIs Registry

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

URI suffix: 'pvd'

Change controller: IETF

Specification document: this document

Status: permanent

Related information: N/A

### 8.2. Additional Information PvD Keys Registry

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

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

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

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

### 8.3. PvD Option Flags Registry

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

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

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

#### 8.4. PvD JSON Media Type Registration

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

Type Name: application

Subtype Name: pvd+json

Required parameters: N/A

Optional parameters: N/A

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

Security considerations: See Section 6.

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

Published specification: This document

Applications that use this media type: This media type is intended to be used by networks advertising additional Provisioning Domain information, and clients looking up such information.

Fragment identifier considerations: N/A

Additional information: N/A

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

Intended usage: COMMON

Restrictions on usage: N/A

Author: IETF

Change controller: IETF

## 9. Acknowledgments

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

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

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

## 10. References

### 10.1. Normative References

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <<https://www.rfc-editor.org/info/rfc2818>>.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<https://www.rfc-editor.org/info/rfc3339>>.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <<https://www.rfc-editor.org/info/rfc4191>>.
- [RFC4343] Eastlake 3rd, D., "Domain Name System (DNS) Case Insensitivity Clarification", RFC 4343, DOI 10.17487/RFC4343, January 2006, <<https://www.rfc-editor.org/info/rfc4343>>.

- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<https://www.rfc-editor.org/info/rfc4941>>.
- [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <<https://www.rfc-editor.org/info/rfc6724>>.
- [RFC6980] Gont, F., "Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery", RFC 6980, DOI 10.17487/RFC6980, August 2013, <<https://www.rfc-editor.org/info/rfc6980>>.
- [RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <<https://www.rfc-editor.org/info/rfc7493>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.
- [RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <<https://www.rfc-editor.org/info/rfc7556>>.
- [RFC8028] Baker, F. and B. Carpenter, "First-Hop Router Selection by Hosts in a Multi-Prefix Network", RFC 8028, DOI 10.17487/RFC8028, November 2016, <<https://www.rfc-editor.org/info/rfc8028>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <<https://www.rfc-editor.org/info/rfc8615>>.

## 10.2. Informative References

- [I-D.kline-mif-mpvd-api-reqs]  
Kline, E., "Multiple Provisioning Domains API Requirements", draft-kline-mif-mpvd-api-reqs-00 (work in progress), November 2015.
- [I-D.stenberg-mif-mpvd-dns]  
Stenberg, M. and S. Barth, "Multiple Provisioning Domains using Domain Name System", draft-stenberg-mif-mpvd-dns-00 (work in progress), October 2015.
- [IEEE8021X]  
IEEE, "IEEE Standards for Local and Metropolitan Area Networks, Port-based Network Access Control, IEEE Std".
- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <<https://www.rfc-editor.org/info/rfc2131>>.
- [RFC3646] Droms, R., Ed., "DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3646, DOI 10.17487/RFC3646, December 2003, <<https://www.rfc-editor.org/info/rfc3646>>.
- [RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "Secure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <<https://www.rfc-editor.org/info/rfc3971>>.
- [RFC4389] Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery Proxies (ND Proxy)", RFC 4389, DOI 10.17487/RFC4389, April 2006, <<https://www.rfc-editor.org/info/rfc4389>>.
- [RFC6105] Levy-Abegnoli, E., Van de Velde, G., Popoviciu, C., and J. Mohacsi, "IPv6 Router Advertisement Guard", RFC 6105, DOI 10.17487/RFC6105, February 2011, <<https://www.rfc-editor.org/info/rfc6105>>.

- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <<https://www.rfc-editor.org/info/rfc6146>>.
- [RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, DOI 10.17487/RFC6147, April 2011, <<https://www.rfc-editor.org/info/rfc6147>>.
- [RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", RFC 6296, DOI 10.17487/RFC6296, June 2011, <<https://www.rfc-editor.org/info/rfc6296>>.
- [RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <<https://www.rfc-editor.org/info/rfc6960>>.
- [RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <<https://www.rfc-editor.org/info/rfc7049>>.
- [RFC7278] Byrne, C., Drown, D., and A. Vizdal, "Extending an IPv6 /64 Prefix from a Third Generation Partnership Project (3GPP) Mobile Interface to a LAN Link", RFC 7278, DOI 10.17487/RFC7278, June 2014, <<https://www.rfc-editor.org/info/rfc7278>>.
- [RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <<https://www.rfc-editor.org/info/rfc7540>>.
- [RFC8106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 8106, DOI 10.17487/RFC8106, March 2017, <<https://www.rfc-editor.org/info/rfc8106>>.



- [RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <<https://www.rfc-editor.org/info/rfc8415>>.
- [URN] IANA, "Uniform Resource Names (URN) Namespaces", <<https://www.iana.org/assignments/urn-namespaces/urn-namespaces.xhtml>>.

## Authors' Addresses

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

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

Eric Vyncke  
Cisco  
De Kleetlaan, 6  
Diegem 1831  
Belgium

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

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

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

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

Email: [dschinazi.ietf@gmail.com](mailto:dschinazi.ietf@gmail.com)

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

Email: wenshao@cisco.com

ippm, 6man  
Internet-Draft  
Intended status: Standards Track  
Expires: September 30, 2020

S. Bhandari  
F. Brockners  
Cisco  
T. Mizrahi  
Huawei Network.IO Innovation Lab  
A. Kfir  
B. Gafni  
Mellanox Technologies, Inc.  
M. Spiegel  
Barefoot Networks, an Intel company  
S. Krishnan  
Kaloom  
M. Smith  
March 29, 2020

Deployment Considerations for In-situ OAM with IPv6 Options  
draft-ioametal-ippm-6man-ioam-ipv6-deployment-03

Abstract

In-situ Operations, Administration, and Maintenance (IOAM) records operational and telemetry information in the packet while the packet traverses a path between two points in the network. This document outlines how IOAM can be enabled in an IPv6 network.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 30, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                                                                     |   |
|---------------------------------------------------------------------|---|
| 1. Introduction . . . . .                                           | 2 |
| 2. Conventions . . . . .                                            | 3 |
| 2.1. Requirements Language . . . . .                                | 3 |
| 2.2. Abbreviations . . . . .                                        | 3 |
| 3. Considerations for IOAM deployment in IPv6 networks . . . . .    | 3 |
| 4. IOAM domains bounded by hosts . . . . .                          | 4 |
| 5. IOAM domains bounded by network devices . . . . .                | 4 |
| 5.1. Deployment options . . . . .                                   | 5 |
| 5.1.1. IPv6-in-IPv6 encapsulation . . . . .                         | 5 |
| 5.1.2. IP-in-IPv6 encapsulation with ULA . . . . .                  | 5 |
| 5.1.3. x-in-IPv6 Encapsulation that is used Independently . . . . . | 6 |
| 6. Security Considerations . . . . .                                | 6 |
| 7. IANA Considerations . . . . .                                    | 6 |
| 8. Acknowledgements . . . . .                                       | 7 |
| 9. References . . . . .                                             | 7 |
| 9.1. Normative References . . . . .                                 | 7 |
| 9.2. Informative References . . . . .                               | 7 |
| Authors' Addresses . . . . .                                        | 8 |

## 1. Introduction

In-situ Operations, Administration, and Maintenance (IOAM) records operational and telemetry information in the packet while the packet traverses a path between two points in the network.

[I-D.ioametal-ippm-6man-ioam-ipv6-options] defines how IOAM data fields are encapsulated in the IPv6 [RFC8200]. This document discusses deployment options for networks which leverage IOAM data fields encapsulated in the IPv6 protocol.

Deployment considerations differ, whether the IOAM domain starts and ends on hosts or whether the IOAM encapsulating and decapsulating nodes are network devices that forward traffic, such as routers.

## 2. Conventions

### 2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

### 2.2. Abbreviations

Abbreviations used in this document:

E2E:            Edge-to-Edge

IOAM:          In-situ Operations, Administration, and Maintenance

ION:           IOAM Overlay Network

OAM:           Operations, Administration, and Maintenance

POT:           Proof of Transit

## 3. Considerations for IOAM deployment in IPv6 networks

IOAM deployment in an IPv6 network should take the following considerations and requirements into account:

- C1 It is desirable that the addition of IOAM data fields neither changes the way routers forward the packets, nor the forwarding decision the routers takes. The packet with the added OAM information should follow the same path within the domain that the same packet without the OAM information would follow within the domain even in the presence of ECMP. Such a behavior is particularly interesting for deployments where IOAM data fields are only added "on-demand", e.g. to provide further insights in case of undesired network behavior for certain flows. Implementations of IOAM should ensure that ECMP behavior for packets with and without IOAM data fields is the same.
- C2 Given that IOAM data fields increase the total size of the packet, the size of the packet including the IOAM data could exceed the PMTU. In particular, the incremental trace IOAM HbH Option, which is proposed to support hardware implementations of IOAM, changes Option Data Length en-route. Operators of an IOAM domain are to ensure that the addition of OAM information does not lead to fragmentation of the packet, e.g. by configuring the MTU of

transit routers and switches to a sufficiently high value. Careful control of the MTU in a network is one of the reasons why IOAM is considered a domain specific feature, see also [I-D.ietf-ippm-ioam-data]. In addition, the PMTU tolerance range in the IOAM domain should be identified (e.g. through configuration) and IOAM encapsulation operations and/or IOAM data field insertion (in case of incremental tracing) should not be performed if it exceeds the packet size beyond PMTU.

- C3 Packets with IOAM data or associated ICMP errors, should not arrive at destinations which have no knowledge of IOAM. Consider using IOAM in transit devices; misleading ICMP errors due to addition and/or presence of OAM data in the packet can confuse a source of the packet that did not insert the OAM information.
- C4 OAM data leaks may affect the forwarding behavior and state of network elements outside an IOAM domain. IOAM domains SHOULD provide a mechanism to prevent data leaks or be able to assure that upon leak network elements outside the domain are not affected i.e they continue to process other valid packets.
- C5 The source of that inserted and leaked the IOAM data must be easy to identify for the purpose of troubleshooting, due to the high complexity of troubleshooting a source that inserted the IOAM data and did not remove it when the packet traversed across an AS. Such a troubleshooting process may require coordination between multiple operators, complicated configuration verification, packet capture analysis, etc.
- C6 Compliance with [RFC8200] would require OAM data to be encapsulated instead of header/option insertion directly into in-flight packets using the original IPv6 header.

#### 4. IOAM domains bounded by hosts

For deployments where the IOAM domain is bounded by hosts, hosts will perform the operation of IOAM data field encapsulation and decapsulation. IOAM data is carried in IPv6 packets as Hop-by-Hop or Destination options, see [I-D.ioametal-ippm-6man-ioam-ipv6-options].

#### 5. IOAM domains bounded by network devices

For deployments where the IOAM domain is bounded by network devices, network devices such as routers form the edge of an IOAM domain. Network devices will perform the operation of IOAM data field encapsulation and decapsulation.

## 5.1. Deployment options

This section lists out possible deployment options that can be employed to meet the requirements listed in Section 3.

### 5.1.1. IPv6-in-IPv6 encapsulation

Leverage an IPv6-in-IPv6 approach: Preserve the original IP packet and add an IPv6 header including IOAM data fields in an extension header in front of it, to forward traffic within and across the IOAM domain. The overlay network formed by the additional IPv6 header with the IOAM data fields included in an extension header is referred to as IOAM Overlay Network (ION) in this document.

1. Perform an IPv6-in-IPv6 approach. The source address of the outer IPv6 header is that of the IOAM encapsulating node. The destination address of the outer IPv6 header is the same as the inner IPv6 destination address, i.e. the destination address of the packet does not change.
2. To simplify debugging in case of leaked IOAM data fields in packets, consider a new IOAM E2E destination option to identify the Source IOAM domain (AS, v6 prefix). Insert this option into the IOAM destination options EH attached to the outer IPv6 header. This additional information would allow for easy identification of an AS operator that is the source of packets with leaked IOAM information. Note that leaked packets with IOAM data fields would only occur in case a router would be misconfigured. [I-D.ioametal-ippm-6man-ioam-ipv6-options] requires that by default, packets with extension headers which carry IOAM data fields are dropped unless the router's interfaces are explicitly configured for IOAM.
3. All the IOAM options are defined with type "00 - skip over this option and continue processing the header. So presence of the options must not cause packet drop in the network elements that do not understand the option. In addition [I-D.ietf-6man-hbh-header-handling] should be considered.

### 5.1.2. IP-in-IPv6 encapsulation with ULA

The "IP-in-IPv6 encapsulation with ULA" [RFC4193] approach can be used to apply IOAM to an IPv6 as well as an IPv4 network. In addition, it fulfills requirement C4 (avoid leaks) by using ULA for the ION. Similar to the IPv6-in-IPv6 encapsulation approach above, the original IP packet is preserved. An IPv6 header including IOAM data fields in an extension header is added in front of it, to forward traffic within and across the IOAM domain. IPv6 addresses

for the ION, i.e. the outer IPv6 addresses are assigned from the ULA space. Addressing and routing in the ION are to be configured so that the IP-in-IPv6 encapsulated packets follow the same path as the original, non-encapsulated packet would have taken. This would create an internal IPv6 forwarding topology using the IOAM domain's interior ULA address space which is parallel with the forwarding topology that exists with the non-IOAM address space (the topology and address space that would be followed by packets that do not have supplemental IOAM information). Establishment and maintenance of the parallel IOAM ULA forwarding topology could be automated, e.g. similar to how LDP [RFC5036] is used in MPLS to establish and maintain an LSP forwarding topology that is parallel to the network's IGP forwarding topology.

Transit across the ION could leverage the transit approach for traffic between BGP border routers, as described in [RFC1772], "A.2.3 Encapsulation". Assuming that the operational guidelines specified in Section 4 of [RFC4193] are properly followed, the probability of leaks in this approach will be almost close to zero. If the packets do leak through IOAM egress device misconfiguration or partial IOAM egress device failure, the packets' ULA destination address is invalid outside of the IOAM domain. There is no exterior destination to be reached, and the packets will be dropped when they encounter either a router external to the IOAM domain that has a packet filter that drops packets with ULA destinations, or a router that does not have a default route.

#### 5.1.3. x-in-IPv6 Encapsulation that is used Independently

In some cases it is desirable to monitor a domain that uses an overlay network that is deployed independently of the need for IOAM, e.g., an overlay network that runs Geneve-in-IPv6, or VXLAN-in-IPv6. In this case IOAM can be encapsulated in as an extension header in the tunnel (outer) IPv6 header. Thus, the tunnel encapsulating node is also the IOAM encapsulating node, and the tunnel end point is also the IOAM decapsulating node.

## 6. Security Considerations

This document discusses the deployment of IOAM with IPv6 options. Security considerations of the specific IOAM data fields are described in [I-D.ietf-ippm-ioam-data].

## 7. IANA Considerations

There are no IANA considerations that apply to this document.



## 8. Acknowledgements

The authors would like to thank Mark Smith, Tom Herbert, Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya Nadahalli, Stefano Previdi, Hemant Singh, Erik Nordmark, LJ Wobker, and Andrew Yourtchenko for the comments and advice. For the IPv6 encapsulation, this document leverages concepts described in [I-D.kitamura-ipv6-record-route]. The authors would like to acknowledge the work done by the author Hiroshi Kitamura and people involved in writing it.

## 9. References

### 9.1. Normative References

- [I-D.ietf-ippm-ioam-data]  
Brockners, F., Bhandari, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., Chang, R., daniel.bernier@bell.ca, d., and J. Lemon, "Data Fields for In-situ OAM", draft-ietf-ippm-ioam-data-04 (work in progress), October 2018.
- [I-D.ioametal-ippm-6man-ioam-ipv6-options]  
Bhandari, S., Brockners, F., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Kfir, A., Gafni, B., Lapukhov, P., Spiegel, M., and S. Krishnan, "In-situ OAM IPv6 Options", draft-ioametal-ippm-6man-ioam-ipv6-options-01 (work in progress), October 2018.
- [RFC1772] Rekhter, Y. and P. Gross, "Application of the Border Gateway Protocol in the Internet", RFC 1772, DOI 10.17487/RFC1772, March 1995, <<https://www.rfc-editor.org/info/rfc1772>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

### 9.2. Informative References

- [I-D.ietf-6man-hbh-header-handling]  
Baker, F. and R. Bonica, "IPv6 Hop-by-Hop Options Extension Header", March 2016.

- [I-D.kitamura-ipv6-record-route]  
Kitamura, H., "Record Route for IPv6 (PR6) Hop-by-Hop Option Extension", draft-kitamura-ipv6-record-route-00 (work in progress), November 2000.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<https://www.rfc-editor.org/info/rfc4193>>.
- [RFC5036] Andersson, L., Ed., Minei, I., Ed., and B. Thomas, Ed., "LDP Specification", RFC 5036, DOI 10.17487/RFC5036, October 2007, <<https://www.rfc-editor.org/info/rfc5036>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8250] Elkins, N., Hamilton, R., and M. Ackermann, "IPv6 Performance and Diagnostic Metrics (PDM) Destination Option", RFC 8250, DOI 10.17487/RFC8250, September 2017, <<https://www.rfc-editor.org/info/rfc8250>>.

#### Authors' Addresses

Shwetha Bhandari  
Cisco Systems, Inc.  
Cessna Business Park, Sarjapura Marathalli Outer Ring Road  
Bangalore, KARNATAKA 560 087  
India

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

Frank Brockners  
Cisco Systems, Inc.  
Kaiserswerther Str. 115,  
RATINGEN, NORDRHEIN-WESTFALEN 40880  
Germany

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

Tal Mizrahi  
Huawei Network.IO Innovation Lab  
Israel

Email: [tal.mizrahi.phd@gmail.com](mailto:tal.mizrahi.phd@gmail.com)

Aviv Kfir  
Mellanox Technologies, Inc.  
350 Oakmead Parkway, Suite 100  
Sunnyvale, CA 94085  
U.S.A.

Email: avivk@mellanox.com

Barak Gafni  
Mellanox Technologies, Inc.  
350 Oakmead Parkway, Suite 100  
Sunnyvale, CA 94085  
U.S.A.

Email: gbarak@mellanox.com

Mickey Spiegel  
Barefoot Networks, an Intel company  
4750 Patrick Henry Drive  
Santa Clara, CA 95054  
US

Email: mickey.spiegel@intel.com

Suresh Krishnan  
Kaloom

Email: suresh@kaloom.com

Mark Smith  
PO BOX 521  
HEIDELBERG, VIC 3084  
AU

Email: markzzzsmith+id@gmail.com

ippm,6man  
Internet-Draft  
Intended status: Standards Track  
Expires: September 29, 2019

S. Bhandari  
F. Brockners  
C. Pignataro  
Cisco  
H. Gredler  
RtBrick Inc.  
J. Leddy  
Comcast  
S. Youell  
JMPC  
T. Mizrahi  
Huawei Network.IO Innovation Lab  
A. Kfir  
B. Gafni  
Mellanox Technologies, Inc.  
P. Lapukhov  
Facebook  
M. Spiegel  
Barefoot Networks  
S. Krishnan  
Kaloom  
R. Asati  
Cisco  
March 28, 2019

In-situ OAM IPv6 Options  
draft-ioametal-ippm-6man-ioam-ipv6-options-02

Abstract

In-situ Operations, Administration, and Maintenance (IOAM) records operational and telemetry information in the packet while the packet traverses a path between two points in the network. This document outlines how IOAM data fields are encapsulated in IPv6.

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 September 29, 2019.

#### Copyright Notice

Copyright (c) 2019 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. Conventions . . . . .                            | 2 |
| 2.1. Requirements Language . . . . .                | 3 |
| 2.2. Abbreviations . . . . .                        | 3 |
| 3. In-situ OAM Metadata Transport in IPv6 . . . . . | 3 |
| 4. Security Considerations . . . . .                | 5 |
| 5. IANA Considerations . . . . .                    | 6 |
| 6. Acknowledgements . . . . .                       | 6 |
| 7. References . . . . .                             | 6 |
| 7.1. Normative References . . . . .                 | 6 |
| 7.2. Informative References . . . . .               | 7 |
| Authors' Addresses . . . . .                        | 7 |

#### 1. Introduction

In-situ Operations, Administration, and Maintenance (IOAM) records operational and telemetry information in the packet while the packet traverses a path between two points in the network. This document outlines how IOAM data fields are encapsulated in the IPv6 [RFC8200].

#### 2. Conventions

## 2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 2.2. Abbreviations

Abbreviations used in this document:

E2E:            Edge-to-Edge

IOAM:          In-situ Operations, Administration, and Maintenance

OAM:           Operations, Administration, and Maintenance

POT:           Proof of Transit

## 3. In-situ OAM Metadata Transport in IPv6

In-situ OAM in IPv6 is used to enhance diagnostics of IPv6 networks. It complements other mechanisms proposed to enhance diagnostics of IPv6 networks, such as the IPv6 Performance and Diagnostic Metrics Destination Option described in [RFC8250].

IOAM data fields are encapsulated in "option data" fields of two types of extension headers in IPv6 packets - either Hop-by-Hop Options header or Destination options header. The selection of a particular extension header type depends on IOAM usage, as described in section 4 of [I-D.ietf-ippm-ioam-data]. Multiple options with the same Option Type MAY appear in the same Hop-by-Hop Options or Destination Options header, with varying content.

In order for IOAM to work in IPv6 networks, IOAM MUST be explicitly enabled per interface on every node within the IOAM domain. Unless a particular interface is explicitly enabled (i.e. explicitly configured) for IOAM, a router MUST drop packets which contain extension headers carrying IOAM data-fields. This is the default behavior and is independent of whether the Hop-by-Hop options or Destination options are used to encode the IOAM data. This ensures that IOAM data does not unintentionally get forwarded outside the IOAM domain.

An IPv6 packet carrying IOAM data in an Extension header can have other extension headers, compliant with [RFC8200].



2. Incremental Tracing Option: The in-situ OAM Incremental Tracing option defined in [I-D.ietf-ippm-ioam-data] is represented as a IPv6 option in hop by hop extension header:

Option Type: 001xxxxx 8-bit identifier of the IOAM type of option. xxxxx=TBD.

IOAM Type: IOAM Incremental Trace Option Type.

3. Proof of Transit Option: The in-situ OAM POT option defined in [I-D.ietf-ippm-ioam-data] is represented as a IPv6 option in hop by hop extension header:

Option Type: 001xxxxx 8-bit identifier of the IOAM type of option. xxxxx=TBD.

IOAM Type: IOAM POT Option Type.

4. Edge to Edge Option: The in-situ OAM E2E option defined in [I-D.ietf-ippm-ioam-data] is represented as a IPv6 option in IPv6 option in destination options extension header:

Option Type: 000xxxxx 8-bit identifier of the IOAM type of option. xxxxx=TBD.

IOAM Type: IOAM E2E Option Type.

All the in-situ OAM IPv6 options defined here have alignment requirements. Specifically, they all require 4n alignment. This ensures that 4 octet fields specified in [I-D.ietf-ippm-ioam-data] such as transit delay are aligned at a multiple-of-4 offset from the start of the Hop-by-Hop Options header. In addition, to maintain IPv6 extension header 8-octet alignment and avoid the need to add or remove padding at every hop, the Trace-Type for Incremental Tracing Option in IPv6 MUST be selected such that the IOAM node data length is a multiple of 8-octets.

#### 4. Security Considerations

This document describes the encapsulation of IOAM data fields in IPv6. Security considerations of the specific IOAM data fields for each case (i.e., Trace, Proof of Transit, and E2E) are described in defined in [I-D.ietf-ippm-ioam-data].

As this document describes new options for IPv6, these are similar to the security considerations of [RFC8200] and the new weakness documented in [RFC8250].



## 5. IANA Considerations

This draft requests the following IPv6 Option Type assignments from the Destination Options and Hop-by-Hop Options sub-registry of Internet Protocol Version 6 (IPv6) Parameters.

<http://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#ipv6-parameters-2>

| Hex Value | Binary Value |     |       | Description | Reference    |
|-----------|--------------|-----|-------|-------------|--------------|
|           | act          | chg | rest  |             |              |
| TBD_1_0   | 00           | 0   | TBD_1 | IOAM        | [This draft] |
| TBD_1_1   | 00           | 1   | TBD_1 | IOAM        | [This draft] |

## 6. Acknowledgements

The authors would like to thank Tom Herbert, Eric Vyncke, Nalini Elkins, Srihari Raghavan, Ranganathan T S, Karthik Babu Harichandra Babu, Akshaya Nadahalli, Stefano Previdi, Hemant Singh, Erik Nordmark, LJ Wobker, Mark Smith, and Andrew Yourtchenko for the comments and advice. For the IPv6 encapsulation, this document leverages concepts described in [I-D.kitamura-ipv6-record-route]. The authors would like to acknowledge the work done by the author Hiroshi Kitamura and people involved in writing it.

## 7. References

### 7.1. Normative References

- [I-D.ietf-ippm-ioam-data] Brockners, F., Bhandari, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., Chang, R., and d. daniel.bernier@bell.ca, "Data Fields for In-situ OAM", draft-ietf-ippm-ioam-data-01 (work in progress), October 2017.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

## 7.2. Informative References

- [I-D.kitamura-ipv6-record-route]  
Kitamura, H., "Record Route for IPv6 (PR6) Hop-by-Hop Option Extension", draft-kitamura-ipv6-record-route-00 (work in progress), November 2000.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8250] Elkins, N., Hamilton, R., and M. Ackermann, "IPv6 Performance and Diagnostic Metrics (PDM) Destination Option", RFC 8250, DOI 10.17487/RFC8250, September 2017, <<https://www.rfc-editor.org/info/rfc8250>>.

## Authors' Addresses

Shwetha Bhandari  
Cisco Systems, Inc.  
Cessna Business Park, Sarjapura Marathalli Outer Ring Road  
Bangalore, KARNATAKA 560 087  
India

Email: shwethab@cisco.com

Frank Brockners  
Cisco Systems, Inc.  
Kaiserswerther Str. 115,  
RATINGEN, NORDRHEIN-WESTFALEN 40880  
Germany

Email: fbrockne@cisco.com

Carlos Pignataro  
Cisco Systems, Inc.  
7200-11 Kit Creek Road  
Research Triangle Park, NC 27709  
United States

Email: cpignata@cisco.com

Hannes Gredler  
RtBrick Inc.

Email: hannes@rtbrick.com

John Leddy  
Comcast

Email: John\_Leddy@cable.comcast.com

Stephen Youell  
JP Morgan Chase  
25 Bank Street  
London E14 5JP  
United Kingdom

Email: stephen.youell@jpmorgan.com

Tal Mizrahi  
Huawei Network.IO Innovation Lab  
Israel

Email: tal.mizrahi.phd@gmail.com

Aviv Kfir  
Mellanox Technologies, Inc.  
350 Oakmead Parkway, Suite 100  
Sunnyvale, CA 94085  
U.S.A.

Email: avivk@mellanox.com

Barak Gafni  
Mellanox Technologies, Inc.  
350 Oakmead Parkway, Suite 100  
Sunnyvale, CA 94085  
U.S.A.

Email: gbarak@mellanox.com

Petr Lapukhov  
Facebook  
1 Hacker Way  
Menlo Park, CA 94025  
US

Email: petr@fb.com

Mickey Spiegel  
Barefoot Networks  
4750 Patrick Henry Drive  
Santa Clara, CA 95054  
US

Email: mspiegel@barefootnetworks.com

Suresh Krishnan  
Kaloom

Email: suresh@kaloom.com

Rajiv Asati  
Cisco Systems, Inc.  
7200 Kit Creek Road  
Research Triangle Park , NC 27709  
US

Email: rajiva@cisco.com

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: March 13, 2020

Z. Li  
S. Peng  
Huawei Technologies  
K. LEE  
LG U+  
September 10, 2019

IPv6 Encapsulation for SFC and IFIT  
draft-li-6man-ipv6-sfc-ifit-02

Abstract

Service Function Chaining (SFC) and In-situ Flow Information Telemetry (IFIT) are important path services along with the packets. In order to support these services, several encapsulations have been defined. The document analyzes the problems of these encapsulations in the IPv6 scenario and proposes the possible optimized encapsulation for IPv6.

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].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 13, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                                               |    |
|-----------------------------------------------|----|
| 1. Introduction . . . . .                     | 2  |
| 2. Terminology . . . . .                      | 3  |
| 3. Problem Statement . . . . .                | 3  |
| 4. Design Consideration . . . . .             | 4  |
| 4.1. Service Options . . . . .                | 4  |
| 4.2. IPv6 Service Metadata Options . . . . .  | 7  |
| 4.2.1. SFC Service Metadata Option . . . . .  | 7  |
| 4.2.2. IOAM Service Metadata Option . . . . . | 8  |
| 4.2.3. IFA Service Metadata Option . . . . .  | 8  |
| 5. IANA Considerations . . . . .              | 9  |
| 6. Security Considerations . . . . .          | 9  |
| 7. References . . . . .                       | 9  |
| 7.1. Normative References . . . . .           | 9  |
| 7.2. Informative References . . . . .         | 11 |
| Authors' Addresses . . . . .                  | 11 |

## 1. Introduction

Service Function Chaining (SFC) [RFC7665] and In-situ Flow Information Telemetry (IFIT) [I-D.song-opsawg-ifit-framework] are important path services along with the packets. In order to support these services, several encapsulations have been defined. Network Service Header (NSH) is defined in [RFC8300] as the encapsulation for SFC. For IFIT encapsulations, In-situ OAM (iOAM) Header is defined in [I-D.ietf-ippm-ioam-data] and Postcard-Based Telemetry (PBT) Header is defined in [I-D.song-ippm-postcard-based-telemetry]. Inband Flow Analyzer (IFA) is also defined in [I-D.kumar-ippm-ifa] to record flow specific information from an end station and/or switches across a network. In the application scenario of IPv6, these encapsulations propose challenges for the data plane. The document analyzes the problems and proposes the possible optimized encapsulation for IPv6.

## 2. Terminology

SFC: Service Function Chaining

IFIT: In-situ Flow Information Telemetry

IOAM: In-situ OAM

PBT: Postcard-Based Telemetry

IFA: Inband Flow Analyzer

SRH: Segment Routing Header

## 3. Problem Statement

The problems posed by the current encapsulations for SFC and IFIT in the application scenarios of IPv6 and SRv6 include:

1. According to the encapsulation order recommended in [RFC8200], if the IOAM is encapsulated in the IPv6 Hop-by-Hop options header, in the incremental trace mode of IOAM as the number of nodes traversed by the IPv6 packets increases, the recorded IOAM information will increase accordingly. This will increase the length of the Hop-by-Hop options header and cause increasing difficulties in reading the subsequent Segment Routing Extension Header (SRH) [I-D.ietf-6man-segment-routing-header] and thereby reduce the forwarding performance of the data plane greatly.

2. With the introduction of SRv6 network programming [I-D.ietf-spring-srv6-network-programming], the path services along with the IPv6 packets can be processed at all the IPv6 network nodes or only at the SRv6 enabled network nodes along the path. It is necessary to distinguish the encapsulations for the specific path service which should be processed by the IPv6 path or the SRv6 path.

3. Both NSH and IOAM need the Metadata field to record metadata information. However currently these metadata has to be recorded separately which may generate redundant metadata information or increase the cost of process.

4. There is unnecessary inconsistency in the current encapsulations for IOAM, IFA and PBT in the IPv6 scenario. Especially it seems unnecessary to define a new specific IPv6 header for IFA, i.e. IFA header.

#### 4. Design Consideration

To solve the problems stated above, in the application scenarios of IPv6 and SRv6, the encapsulations of SFC and IFIT can be optimized with the following design considerations:

- o To separate the SFC/IFIT path service into two parts, i.e. instruction and recording parts. The instruction part (normally with fixed length) can be placed in the front IPv6 extension headers including Hop-by-Hop options header, Destination options header, Routing header, etc. while the recording part can be placed in the back IPv6 extension headers such as being placed after IPv6 Routing Header. In this way the path service instruction in the IPv6 extension headers can be fixed as much as possible to facilitate hardware process to keep forwarding performance while the SFC/IFIT metadata recording part is placed afterwards which enables to stop recording when too much recording information has to be carried to reach the limitation of hardware process.
- o To define SFC/IFIT path service instructions as IPv6 options uniformly which can be placed either in the Hop-by-hop options which indicates the path service processed by all IPv6 enabled nodes along the path or in the SRH option TLVs which indicates the path service processed only by the SRv6 nodes along the SRv6 path indicated by the Segment List in the SRH.
- o To define a unified IPv6 metadata header which can be used as a container to record the service metadata of SFC, IFIT and other possible path services.

According to the above design optimization consideration, in the application scenarios of IPv6 and SRv6 the encapsulations for SFC and IFIT can be defined as below.

##### 4.1. Service Options

###### 1. NSH Service Option



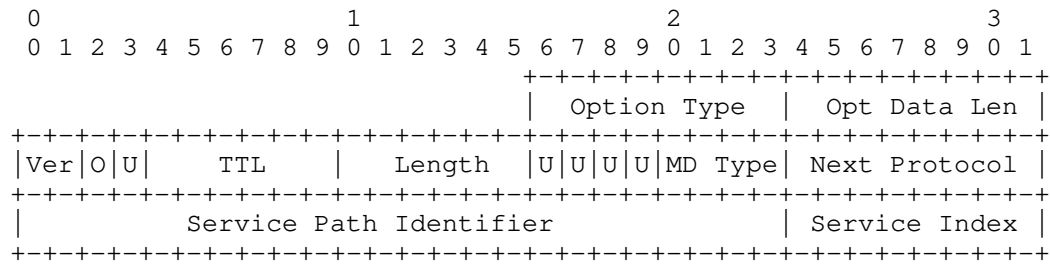


Figure 1. IPv6 Options with NSH instructions

Option Type: TBD\_0

Opt Data Len: 8 octets.

Other fields: refer to [RFC8300].

## 2. IOAM Service Option

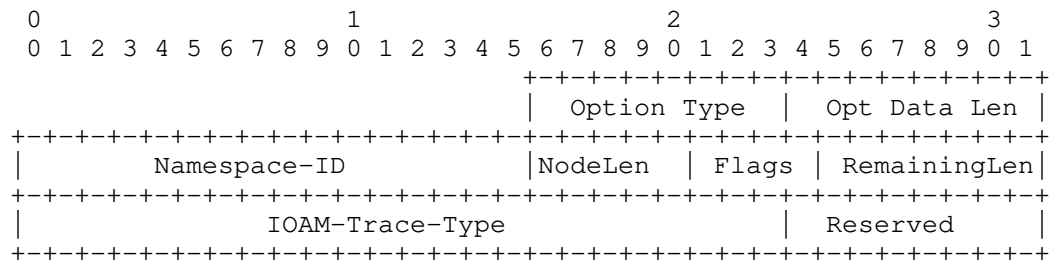


Figure 2. IPv6 Options with IOAM instructions

Option Type: TBD\_1

Opt Data Len: 8 octets.

Other fields: refer to [I-D.ietf-ippm-ioam-data].

## 3. PBT Service Option

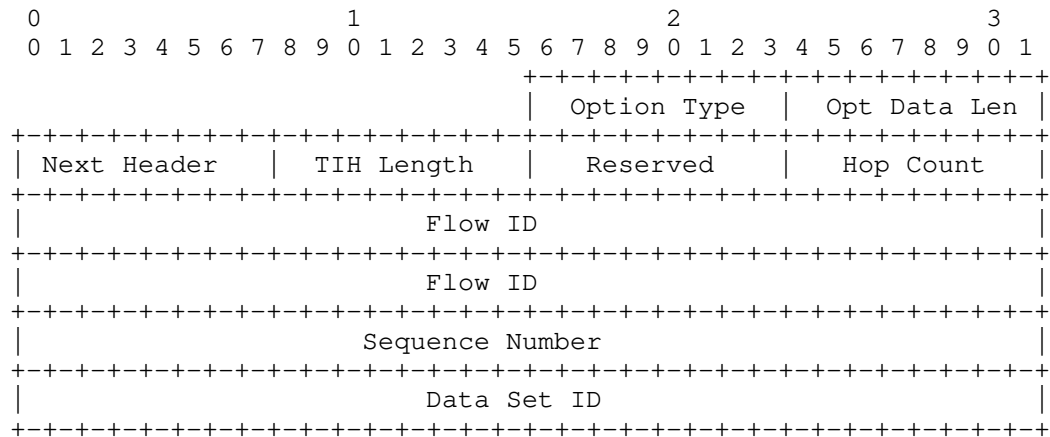


Figure 3. IPv6 Options with PBT instructions

Option Type: TBD\_2

Opt Data Len: 20 octets.

Other fields: refer to [I-D.song-ippm-postcard-based-telemetry].

#### 4. IFA Service Option

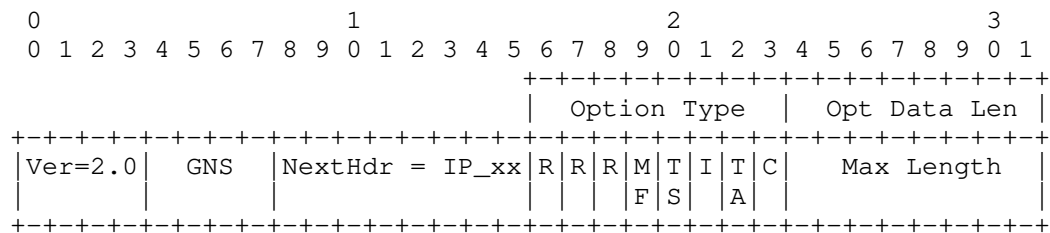


Figure 4. IPv6 Options with IFA instructions

Option Type: TBD\_3

Opt Data Len: 4 octets.

Other fields: refer to [I-D.kumar-ippm-ifa].

These options can be put in the IPv6 Hop-by-Hop Options Header or SRH TLV.

## 4.2. IPv6 Service Metadata Options

As introduced in [I-D.li-6man-enhanced-extension-header], IPv6 Metadata Header is defined as a new type of IPv6 extension header. The metadata is the information recorded by each hop for specific path services, and carried in corresponding service metadata options. The length of the metadata is variable.

### 4.2.1. SFC Service Metadata Option

For the SFC service, the corresponding SFC service metadata option is defined as shown in Figure 5.

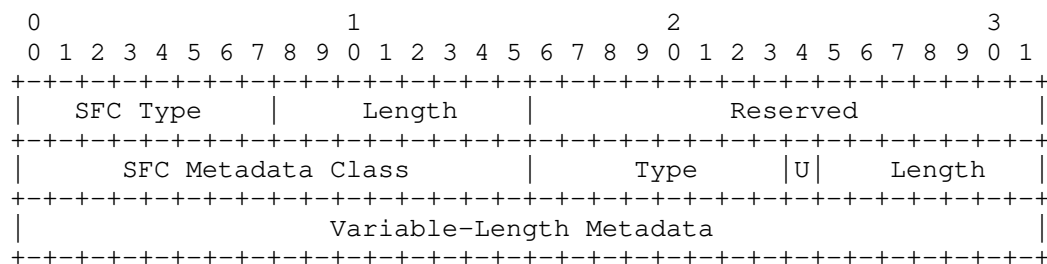


Figure 5. SFC Service Metadata

|                |                                                                                                                                                               |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SFC Type       | 8-bit identifier of the service type, i.e. SFC. The value is TBD-4.                                                                                           |
| Length         | 8-bit unsigned integer. Length of the Service Metadata field, in octets.                                                                                      |
| Metadata Class | Defines the scope of the Type field to provide a hierarchical namespace. IANA has set up the "NSH MD Class" registry, which contains 16-bit values [RFC8300]. |
| Type           | Indicates the explicit type of metadata being carried. The definition of the Type is the responsibility of the MD Class owner.                                |
| Unassigned bit | One unassigned bit is available for future use. This bit MUST NOT be set, and it MUST be ignored on receipt.                                                  |
| Length         | Indicates the length of the variable-length metadata, in bytes. Detailed specification in [RFC8300].                                                          |

#### 4.2.2. IOAM Service Metadata Option

For the IOAM service, the corresponding IOAM service metadata option is defined as shown in Figure 6.

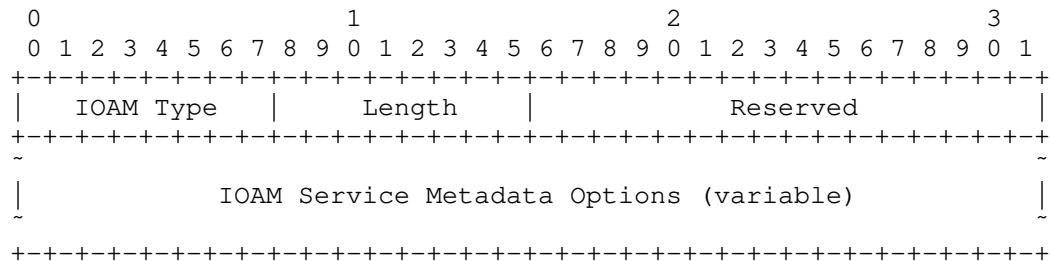


Figure 6. IOAM Service Metadata

|                               |                                                                                                                            |
|-------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| IOAM Type                     | 8-bit identifier of the IOAM Service Metadata type. The value is TBD-5.                                                    |
| Length                        | 8-bit unsigned integer. Length of the IOAM Service Metadata field, in octets.                                              |
| RESERVED                      | 8-bit reserved field MUST be set to zero upon transmission and ignored upon receipt.                                       |
| IOAM Service Metadata Options | IOAM option data is present as specified by the IOAM Type field, and is defined in Section 4 of [I-D.ietf-ippm-ioam-data]. |

All the IOAM IPv6 options require 4n alignment. This ensures that 4 octet fields specified in [I-D.ietf-ippm-ioam-data] such as transit delay are aligned at a multiple-of-4 offset from the start of the IPv6 Metadata header.

In addition, to maintain IPv6 extension header 8-octet alignment and avoid the need to add or remove padding at every hop, the Trace-Type for Incremental Tracing Option in IPv6 MUST be selected such that the IOAM node data length is a multiple of 8-octets.

#### 4.2.3. IFA Service Metadata Option

For the IOAM service, the corresponding IOAM service metadata option is defined as shown in Figure 6.

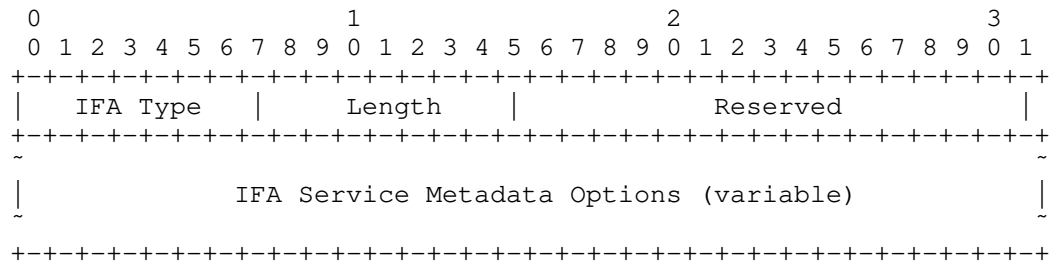


Figure 6. IFA Service Metadata

|                              |                                                                                      |
|------------------------------|--------------------------------------------------------------------------------------|
| IFA Type                     | 8-bit identifier of the IFA Service Metadata type. The value is TBD-6.               |
| Length                       | 8-bit unsigned integer. Length of the IOAM Service Metadata field, in octets.        |
| RESERVED                     | 8-bit reserved field MUST be set to zero upon transmission and ignored upon receipt. |
| IFA Service Metadata Options | IFA option data is present as specified by the IFA Type field.                       |

## 5. IANA Considerations

| Value | Description                | Reference    |
|-------|----------------------------|--------------|
| TBD_0 | NSH Service Option         | [This draft] |
| TBD_1 | IOAM Service Option        | [This draft] |
| TBD_2 | PBT Service Option         | [This draft] |
| TBD_3 | IFA Service Option         | [This draft] |
| TBD_4 | SFC Service Metadata Type  | [This draft] |
| TBD_5 | IOAM Service Metadata Type | [This draft] |
| TBD_6 | IFA Service Metadata Type  | [This draft] |

## 6. Security Considerations

TBD.

## 7. References

### 7.1. Normative References

- [I-D.guichard-spring-nsh-sr]  
Guichard, J., Song, H., Tantsura, J., Halpern, J., Henderickx, W., Boucadair, M., and S. Hassan, "NSH and Segment Routing Integration for Service Function Chaining (SFC)", draft-guichard-spring-nsh-sr-01 (work in progress), March 2019.
- [I-D.ietf-6man-segment-routing-header]  
Filsfils, C., Dukes, D., Previdi, S., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-22 (work in progress), August 2019.
- [I-D.ietf-ippm-ioam-data]  
Brockners, F., Bhandari, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., Chang, R., daniel.bernier@bell.ca, d., and J. Lemon, "Data Fields for In-situ OAM", draft-ietf-ippm-ioam-data-06 (work in progress), July 2019.
- [I-D.ietf-spring-srv6-network-programming]  
Filsfils, C., Camarillo, P., Leddy, J., daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming-01 (work in progress), July 2019.
- [I-D.kumar-ippm-ifa]  
Kumar, J., Anubolu, S., Lemon, J., Manur, R., Holbrook, H., Ghanwani, A., Cai, D., Ou, H., and L. Yizhou, "Inband Flow Analyzer", draft-kumar-ippm-ifa-01 (work in progress), February 2019.
- [I-D.song-ippm-postcard-based-telemetry]  
Song, H., Zhou, T., Li, Z., Shin, J., and K. Lee, "Postcard-based On-Path Flow Data Telemetry", draft-song-ippm-postcard-based-telemetry-04 (work in progress), June 2019.
- [I-D.song-opsawg-ifit-framework]  
Song, H., Li, Z., Zhou, T., Qin, F., Shin, J., and J. Jin, "In-situ Flow Information Telemetry Framework", draft-song-opsawg-ifit-framework-04 (work in progress), September 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <<https://www.rfc-editor.org/info/rfc8300>>.

## 7.2. Informative References

- [RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <<https://www.rfc-editor.org/info/rfc7665>>.

## Authors' Addresses

Zhenbin Li  
Huawei Technologies  
Huawei Bld., No.156 Beiqing Rd.  
Beijing 100095  
China

Email: [lizhenbin@huawei.com](mailto:lizhenbin@huawei.com)

Shuping Peng  
Huawei Technologies  
Huawei Bld., No.156 Beiqing Rd.  
Beijing 100095  
China

Email: [pengshuping@huawei.com](mailto:pengshuping@huawei.com)

Kihoon LEE  
LG U+  
71, Magokjungang 8-ro, Gangseo-gu  
Seoul  
Republic of Korea  
  
Email: soho8416@lguplus.co.kr



Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 12, 2019

Z. Li  
S. Peng  
Huawei Technologies  
March 11, 2019

Service-aware IPv6 Network  
draft-li-6man-service-aware-ipv6-network-00

## Abstract

A multitude of applications are carried over the network, which have varying needs for network bandwidth, latency, jitter, and packet loss, etc. Some applications such as online gaming and live video streaming have very demanding network requirements thereof require special treatments in the network. However, since the current network is lack of enough information of service requirements of such applications it is difficult to guarantee the SLA or it may take long time to provide such guarantee. This document proposes the solution to make use of IPv6 extensions header to convey the service requirement information along with the packet to the network to facilitate the service deployment and network resource adjustment to guarantee SLA for applications. Then it defines the service-aware options which can be used in the different IPv6 extension headers for the purpose.

## 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].

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 12, 2019.

## Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                                        |    |
|----------------------------------------|----|
| 1. Introduction . . . . .              | 2  |
| 2. Use Cases . . . . .                 | 3  |
| 2.1. Online Gaming . . . . .           | 3  |
| 2.2. Video streaming . . . . .         | 3  |
| 3. Problem Statement . . . . .         | 3  |
| 4. Framework . . . . .                 | 4  |
| 5. Service-aware Options . . . . .     | 5  |
| 5.1. Service-aware ID Option . . . . . | 5  |
| 5.2. Service-Para Option . . . . .     | 7  |
| 6. IANA Considerations . . . . .       | 10 |
| 7. Security Considerations . . . . .   | 11 |
| 8. References . . . . .                | 11 |
| 8.1. Normative References . . . . .    | 11 |
| 8.2. Informative References . . . . .  | 11 |
| Authors' Addresses . . . . .           | 11 |

## 1. Introduction

A multitude of applications are carried over the network, which have varying needs for network bandwidth, latency, jitter, and packet loss, etc. Some applications such as online gaming and live video streaming have very demanding network requirements thereof require special treatments in the network. However, since the current network is lack of enough information of service requirements of such applications it is difficult to guarantee the SLA or it may take long time to provide such guarantee. This document proposes the solution to take use of IPv6 extensions header to convey the service requirement information along with the packet to the network to facilitate the service deployment and network resource adjustment to guarantee SLA for applications. Then it defines the service-aware

options which can be used in the different IPv6 extension headers for the purpose.

## 2. Use Cases

This section shows the various demanding requirements of some applications in the following use cases. The traffic of these applications needs to be differentiated from other traffic and applied with special treatments in the network.

### 2.1. Online Gaming

Good network performance is normally a prerequisite for satisfactory game play, especially for the online gaming. The maximum allowable ping rate (network latency) and the required minimum download/upload speed (network bandwidth) are the key factors to make the online gaming playable. Shooting or racing online gaming is normally based on quick action and needs to update the game status in real time by continuously sending and receiving updates to/from the game server and/or other players. The network paths with low latency and low packet loss need to be explicitly selected from the game players to the game server.

### 2.2. Video streaming

The network latency, jitter, bandwidth, and packet loss are the key factors for the video streaming. Live video streaming has even more strict requirements. High quality video source (e.g. from Netflix) require more bandwidth in order to stream properly. Real time streaming services also requires real time content delivery from the web server to the end user ideally via carefully planned explicit TE paths. The online gaming often involves live video streaming.

## 3. Problem Statement

[RFC3272] reviews a number of IETF activities which are primarily intended to evolve the IP architecture to support new service definitions which allow preferential or differentiated treatment to be accorded to certain types of traffic. The challenge when using traditional ways to guarantee SLA is that the packets are not able to carry enough information of service requirements of applications. The network devices mainly relies on the 5-tuple of the packets which cannot provide fine-grained service process. If more information is needed, it has to refer to DPI which will introduce more cost in the network and impose security challenges.

In the era of SDN the orchestrator is introduced for the orchestration of applications and the network. The SDN controller

can be aware of the service requirements of the applications on the network through the interface interworking with the orchestrator. The service requirements is used by the controller for traffic management. The method raises the following problems: 1) The whole loop is long and time-consuming which is not suitable for the real-time adjustment for applications; 2) Too many interfaces are involved in the loop which proposes more challenges of standardization and inter-operability, and it is difficult to be standardized for easy interworking.

#### 4. Framework

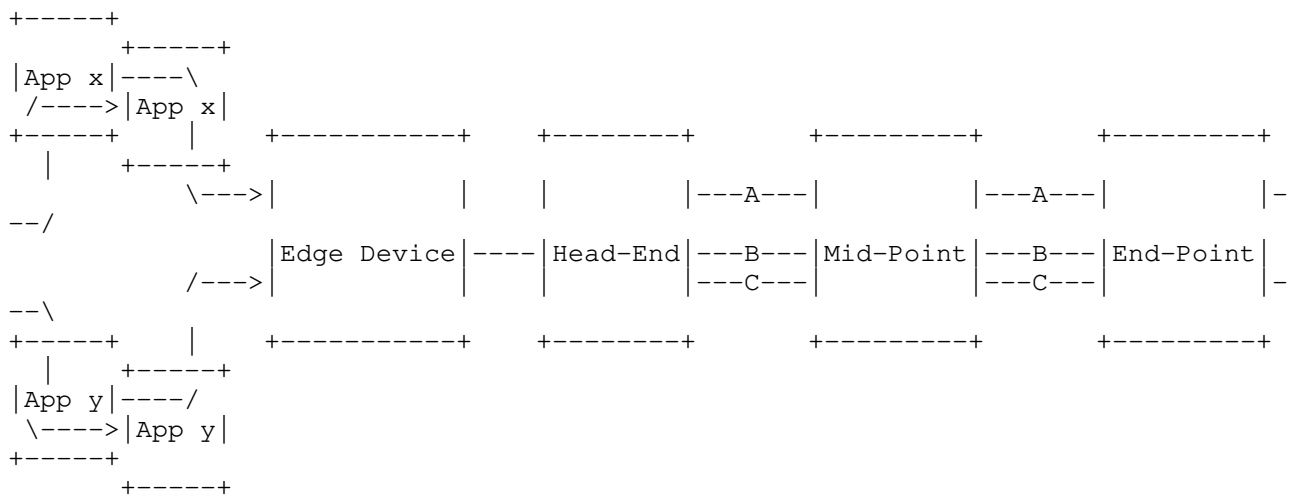


Figure 1 Service-aware IPv6 Network

In the service-aware IPv6 network shown in Figure 1, there are following components:

1. Service-aware Apps: The IPv6 enabled applications runs in the host which can add the service requirements of the applications on network through the IPv6 extension header ([RFC8200]) or remove it from the IPv6 extension header. The service requirement information includes the IPv6 service-aware ID which identifies the IPv6 packets of the traffic belongs to the specific SLA level/Applications/User and the parameters for the specific service such as bandwidth, delay, delay variation, packet loss ratio, etc. The service requirements will be processed by the IPv6 enabled nodes along the path or the SRv6 ([I-D.filsfils-spring-srv6-network-programming]) enabled node along the SRv6 path which be programmed in the host. The Apps can also need not to add any service requirement information in the IPv6 extension header.

2. Service-aware Edge Device: The Edge Device can add the service requirements of the applications on network through the IPv6 extension header on behalf of the IPv6 enabled applications or change the service requirements conveyed by the packets of the service-aware applications according to local policies which is out of the scope of this document. The service requirements will be processed by the

IPv6 enabled nodes along the path or the SRv6 enabled node along the SRv6 path which be programmed by the Edge Device.

3. Service-process Head-End: The service requirements may be processed as a service path such as SRv6 TE path of SFC at the Service-process Head-End. The service requirements conveyed in the IPv6 packets can be mapped to a service path which satisfies the specific requirement, trigger to set up the new service path by the Head-End, or trigger the global traffic adjustment by the controller according to the information provided by the network devices. The process depends on the local policy which is out of the scope this document.

4. Service-process Mid-Point: The Mid-Point provides the path service according to the service path set up by the Head-End which satisfies the service requirement conveyed by the IPv6 packets. The Mid-Point may also adjust the resource locally to guarantee the service requirements depending on specific policies which is out of the scope of this document.

5. Service-process End-Point: The process of the specific service path will end at the End-Point. The service requirements information can be removed at the End-Point or go on to be conveyed with the IPv6 packets.

In this way the network is able to be aware of the service requirements of the applications explicitly. According to these service requirement information carried in the IPv6 packets the network is able to adjust its resource fast to satisfy the service requirement of applications. The flow-driven method also reduces the challenges of inter-operability and loop control loop.

## 5. Service-aware Options

Two service-aware options are defined, i.e. Service-aware ID option and Service-Para Option to support the Service-aware IPv6 network.

### 5.1. Service-aware ID Option

The Service-aware ID option indicates the information of the applications, users, and service requirements, which is defined in the following figure:

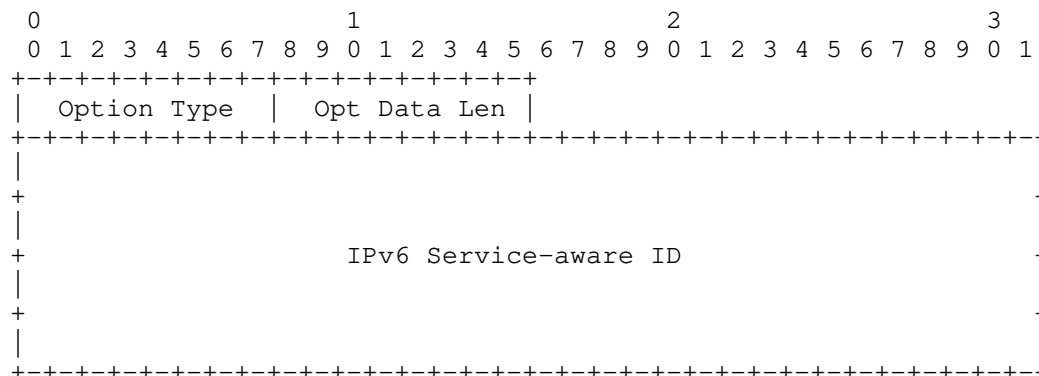


Figure 2. IPv6 Service-aware ID Option

Option Type: TBD

Opt Data Len: 16 octets.

The IPv6 Service-aware ID is 128bits long which can have the following structures:

-- Structure I: Any combination of SLA level (e.g. Gold, Silver, Bronze), APP ID, and/or user ID. The length of each field is variable, which is shown in the following diagram:

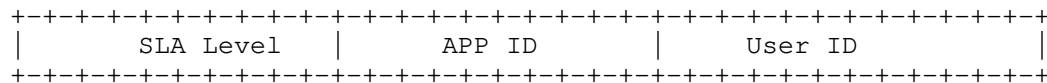


Figure 3. IPv6 Service-aware ID Structure I

```
-- Structure II: Any combination of SLA level (e.g. Gold, Silver,
Bronze), APP ID, and/or user ID plus the arguments which indicates
the service requirements of the identified application, which is
shown in the following diagram:
```

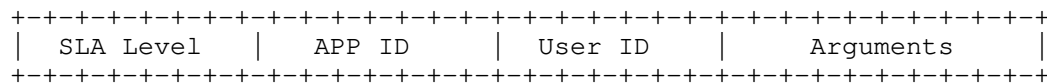


Figure 4. IPv6 Service-aware ID Structure II

-- Structure III: An SRv6 SID, with its arguments as the information specified in Structure 2, which is shown in the following diagram:

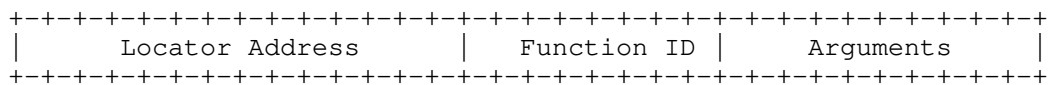


Figure 5. IPv6 Service-aware ID Structure III

This Option can be put into the IPv6 Hop-by-Hop Options, Destination Options, and SRH TLV ([I-D.ietf-6man-segment-routing-header]).

## 5.2. Service-Para Option

The Service-Para Option is a variable-length option carrying multiple service requirement parameters. Each service requirement parameter is put into the corresponding Service Para Sub-TLV, as shown in Figure 6. This Option can be put into the IPv6 Hop-by-Hop Options, Destination Options, and SRH TLV.

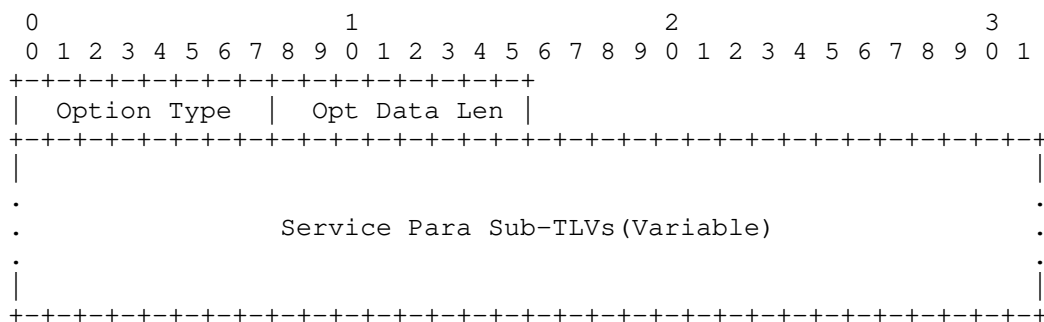


Figure 6. IPv6 Service-Para Option

|                       |                                                              |
|-----------------------|--------------------------------------------------------------|
| Option Type           | TBD                                                          |
| Opt Data Len          | 8-bit unsigned integer. Length of the Service Para Sub-TLVs. |
| Service Para Sub-TLVs | Variable-length field with Service Para Sub-TLVs.            |

The corresponding Service Para Sub-TLVs are shown in the following figures respectively.

## 1. BW Sub-TLV

This BW sub-TLV indicates the bandwidth requirement of applications. The format of this sub-TLV is shown in the following diagram:

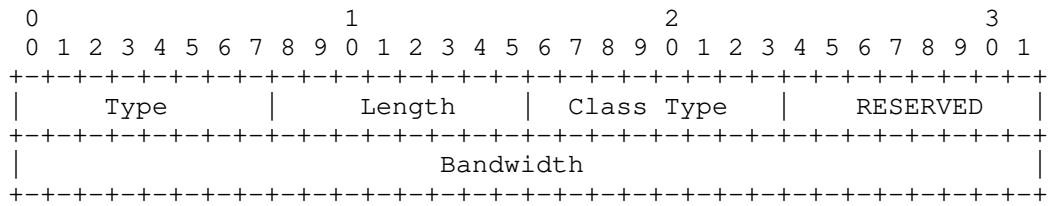


Figure 7. BW Sub-TLV

where:

Type: TBD

Length: 4

Class Type: The Bandwidth Type.

RESERVED: This field is reserved for future use. It MUST be set to 0 when sent and MUST be ignored when received.

Bandwidth: This field carries the bandwidth requirement along the path.

## 2. Delay Sub-TLV

This Delay Sub-TLV indicates the delay requirement of applications. The format of this sub-TLV is shown in the following diagram:

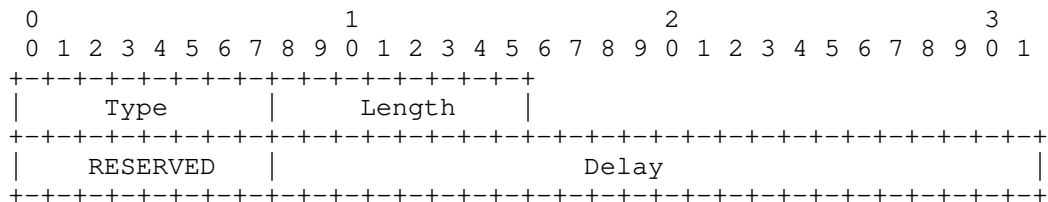


Figure 8. Delay Sub-TLV

where:

Type: TBD

Length: 4

RESERVED: This field is reserved for future use. It MUST be set to 0 when sent and MUST be ignored when received.



**Delay:** This 24-bit field carries the delay requirements in microseconds, encoded as an integer value. When set to the maximum value 16,777,215 (16.777215 sec), then the delay is at least that value and may be larger. This value is the highest delay that can be tolerated.

### 3. Delay Variation Sub-TLV

This Delay Variation Sub-TLV indicates the delay variation requirement of applications. The format of this sub-TLV is shown in the following diagram:

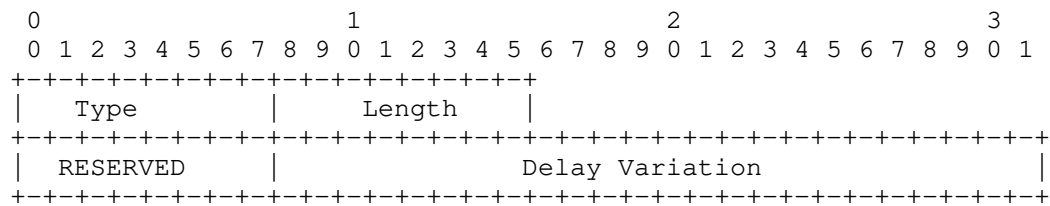


Figure 9. Delay Variation Sub-TLV

where:

Type: TBD

Length: 4

**RESERVED:** This field is reserved for future use. It MUST be set to 0 when sent and MUST be ignored when received.

**Delay Variation:** This 24-bit field carries the delay variation requirements in microseconds, encoded as an integer value.

### 4. Packet Loss Ratio Sub-TLV

This Packet Loss Ratio Sub-TLV indicates the packet loss ratio requirement of applications. The format of this sub-TLV is shown in the following diagram:

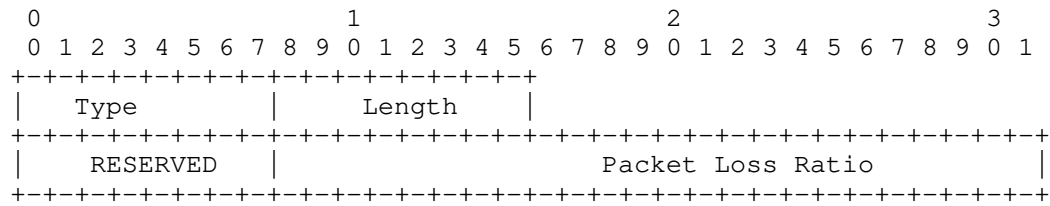


Figure 10. Packet Loss Ratio Sub-TLV

where:

Type: TBD

Length: 4

RESERVED: This field is reserved for future use. It MUST be set to 0 when sent and MUST be ignored when received.

Link Loss: This 24-bit field carries link packet loss ratio requirement. This value is the highest packet-loss ratio that can be tolerated.

## 6. IANA Considerations

IANA maintains the registry for the Options and Sub-TLVs.

Service-Para Option will require one new type code per sub-TLV defined in this document:

Type Value

-----

TBD Service-aware ID Option

TBD Service-Para Option

TBD BW Sub-TLV

TBD Delay Sub-TLV

TBD Delay Variation Sub-TLV

TBD Packet Loss Sub-TLV

## 7. Security Considerations

TBD

## 8. References

### 8.1. Normative References

- [I-D.filsfils-spring-srv6-network-programming]  
Filsfils, C., Camarillo, P., Leddy, J.,  
daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6  
Network Programming", draft-filsfils-spring-srv6-network-  
programming-07 (work in progress), February 2019.
- [I-D.ietf-6man-segment-routing-header]  
Filsfils, C., Previdi, S., Leddy, J., Matsushima, S., and  
d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header  
(SRH)", draft-ietf-6man-segment-routing-header-16 (work in  
progress), February 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate  
Requirement Levels", BCP 14, RFC 2119,  
DOI 10.17487/RFC2119, March 1997,  
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6  
(IPv6) Specification", STD 86, RFC 8200,  
DOI 10.17487/RFC8200, July 2017,  
<<https://www.rfc-editor.org/info/rfc8200>>.

### 8.2. Informative References

- [RFC3272] Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., and X.  
Xiao, "Overview and Principles of Internet Traffic  
Engineering", RFC 3272, DOI 10.17487/RFC3272, May 2002,  
<<https://www.rfc-editor.org/info/rfc3272>>.

### Authors' Addresses

Zhenbin Li  
Huawei Technologies  
Huawei Bld., No.156 Beiqing Rd.  
Beijing 100095  
China

Email: [lizhenbin@huawei.com](mailto:lizhenbin@huawei.com)

Shuping Peng  
Huawei Technologies  
Huawei Bld., No.156 Beiqing Rd.  
Beijing 100095  
China

Email: pengshuping@huawei.com

6MAN Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: August 19, 2019

A. Petrescu  
CEA, LIST  
February 15, 2019

The length of the prefix of an IPv6 link-local address ranges from 10 to  
127  
draft-petrescu-6man-ll-prefix-len-07

#### Abstract

A rejected Errata to RFC4291 "IPv6 Addr Archi" on the topic of link-local addresses 'needs' a draft. This is an answer to that need.

The length of the prefix of an IPv6 link-local address is variable. The minimal value is 10 decimal. The maximum value is 127 decimal.

#### Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 19, 2019.

#### Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                       |                                       |   |
|-----------------------|---------------------------------------|---|
| 1.                    | Definitions and Statements            | 2 |
| 2.                    | Terminology                           | 3 |
| 3.                    | Context                               | 3 |
| 4.                    | Example of use of LL Prefix Length 32 | 4 |
| 5.                    | Security Considerations               | 5 |
| 6.                    | IANA Considerations                   | 5 |
| 7.                    | Contributors                          | 5 |
| 8.                    | Acknowledgements                      | 5 |
| 9.                    | Normative References                  | 6 |
| Appendix A. ChangeLog |                                       | 6 |
| Author's Address      |                                       | 6 |

## 1. Definitions and Statements

The prefix of an IP address is formed by the  $n$  leftmost bits of the address. (in a left-to-right writing system).

The prefix of an IP address is used for goals such as: identify the type of an IPv6 address (link-local, global, others), identify the belonging of an IP address to a particular subnetwork, assist the forwarding (or not forwarding) decisions, and others.

The minimal length of the prefix of an IPv6 link-local address (the value of n) is equal to 10 decimal. The maximum is 127.

The prefix of an IPv6 link-local address is represented textually as "fe80::n", where n MAY be any value between 10 and 127.

Regardless of the prefix length, the leftmost 10 bits of an IPv6 link-local address MUST be set to binary 1111111010 (hexadecimal fe80).

The illustration of an IPv6 link-local address is:

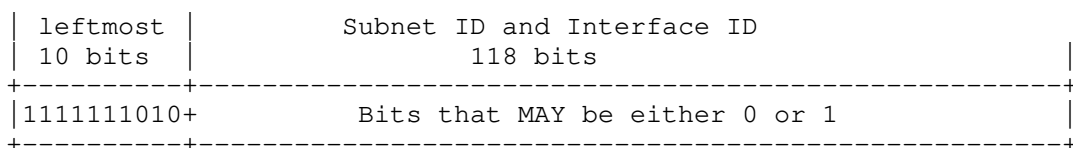


Figure 1: The IPv6 link-local address

Examples: fe80::1/10, fe80:1::1/32 and fe80::1:1/64 are all IPv6 link-local addresses; their prefix lengths are 10, 32 and 64 respectively. Each such IPv6 address has the leftmost 10 bits equal to binary 1111111010.

The Difficulty: the number binary 1111111010 can not be written in hexadecimal without specifying the number of significant bits (fe80::/10); yet that does not make it a 'prefix'. Converting 1111111010 to hexadecimal leads to 3FA (because in a left-to-right writing system the leading 0s before comma are irrelevant); yet '3FA' is not commonly known to be the leading bits of an IPv6 link-local address, fe80::/10 is.

## 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 RFC 2119 [RFC2119].

**prefix:** a contiguous string of bits valid for forwarding operations and for subnet formation. A prefix MUST have an integer length value from 1 to 127 (except when the prefix length is for default route, in which case the value is 0) and a prefix length must be indicated in its textual representation (e.g. 2001:db8::/32 is the prefix and 32 is the prefix length).

**textual representation of a prefix:** e.g. fe80::/64.

**n leading bits:** the first n bits in a string of bits read from left to right in a writing system that is read left-to-right. E.g. the 10 leading bits of the fe80::/64 textual representation of the IPv6 link-local prefix are 1111111010.

## 3. Context

The RFC "IPv6 Address Archi" illustrates the format of the link-local addresses. From the illustration it MAY be understood that the length of the link-local prefix is 10 bits of value 1111111010 and 54 0 bits.

IANA lists the "IPv6 prefix", and "Address Block", to be "fe80::/10" on its website. It is possible that in the future the IETF could decide to use the bits 11-53.

The RFC 2464 "IPv6-over-Ethernet" states that the prefix for link-local addresses is "fe80::/64".

RFC 6874, "Representing IPv6 Zone Identifiers in Address Literals and Uniform Resource Identifiers" specifies the link-local addresses to be under prefix "fe80::/10".

RFC 8415 "DHCPv6" considers link-local addresses are indicated by the prefix fe80::/10.

Several knowledgeable interpretations state that, generally speaking, the prefix length of link-local addresses is 10, but it is 64 in the particular case of Stateless Address-Autoconfiguration (SLAAC). In this latter case, the prefix is named a "subnet prefix", or "prefix on a link", and it is "fe80::/64".

Implementations of an IPv6 stack in a particular operating system allow for the manual configuration of both prefix lengths 64 and 10 for link-local addresses. In another operating system the prefix length for link-local addresses can not be explicitly specified by the end user, but may be indirectly derived from two distinct textual formats by using an unspecified rule. In yet another operating system an end user can not use a link-local address whose value is fe80:1::1; because in that OS the hosts drop incoming packets whose or destination address matches fe80::/10 and contains a non-0 value in bits 15-31 (like fe80:1::1 does).

Misconfigurations and lack of interoperability MAY arise between computers that use mixed prefix lengths for link-local addresses.

A memo describes the use of IPv6 link-local addresses in applications. The filename of the Internet Draft is draft-smith-ipv6-link-locals-apps-00.

Historical note: earlier, the link-local prefix fe80::/10 and site-local prefix fec0::/10 were grouped into a common fe80::/9. If bits 10-64 were 0 then the prefix was a link-local, otherwise a site-local. The site-local addresses were later deprecated by RFC 3879.

#### 4. Example of use of LL Prefix Length 32

This figure shows two routers each with two interfaces; one such interface is connected to the other router; there are two interfaces that point elsewhere.



```

i1 ----- i2 i3-----i4
--|Router1|-----|Router2|---

```

i2 address is fe80:12::1:1/32 ('12' means subnet between R1 and R2, '1' is R1, 2nd '1' is 'front' interface)  
i3 address is fe80:12::2:2/32

Figure 2: Figure

One router's interface (connected to the other router) uses address fe80:12::1:1/32 and the other router's corresponding interface uses address fe80:12::2:2/32.

## 5. Security Considerations

The clarification of the definition of the prefix length of the IPv6 link-local prefix at IANA is: call it 'leading bits' and not 'prefix', or state that the IPv6 prefix length of link-local addresses is 10 decimal. This clarification has beneficial impact in the algorithm implementation for calculation of the opaque and stable Interface Identifiers for IPv6 link-local addresses. It also positively impacts some implementations of IPv6 forwarding.

## 6. IANA Considerations

IANA is requested to change the name of the column head in the table that depicts the "Internet Protocol Version 6 Address Space". The name should be "The n leading bits of an address" instead of "IPv6 Prefix".

The desired effect of this change is that the IPv6 link-local prefix be "fe80::/n" and that the 10 leading bits of this prefix be 1111111010. A second effect is that the textual representation "fe80::/10" as an IPv6 link-local prefix should disappear from that IANA page, because it is wrong.

## 7. Contributors

Listed from 6man WG discussion.

## 8. Acknowledgements

The following persons are acknowledged for the discussion that is reflected in this draft. Not all points are reflected. Some points are copied almost entirely.

Ole Troan, Scott Timothy Morizot, Brian Carpenter, Fred Baker, Mark Smith, Peter Occil, Philip Homburg, Albert Manfredi, &#144;\_&#150;3/4 &#146;B&#141;AE (TATUYA Jinmei, key explainer of a particular OS behaviour), Fernando Gont, Christian Huitema, Simon Hobson, Matthew Petach, Yucel Guven, Sander Steffann, Dennis Ferguson, Musa Stephen Honlue, Fred Templin (for inspiration in an unrelated DHCP discussion).

Peter Paluch submitted the Errata suggestion to RFC 4291 about link-local addresses, and Brian Haberman rejected it, by requiring a draft. Igor Lubashev pointed to that Errata.

## 9. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

## Appendix A. ChangeLog

The changes are listed in reverse chronological order, most recent changes appearing at the top of the list.

-07: added the fact that DHCPv6 spec considers the link-local addresses to be fe80::/10; added a valuable explanation of ll behaviour of a particularly important OS.

-04: added an example advantage of using prefix length 32.

-03:

-02: corrected a typo in "fe80::/1" and added a 7-bit encoding for one persons name (in addition to the japanese-shift-jis encoding which is not understood by xml2rfc.)

## Author's Address

Alexandre Petrescu  
CEA, LIST  
CEA Saclay  
Gif-sur-Yvette , Ile-de-France 91190  
France

Phone: +33169089223  
Email: [Alexandre.Petrescu@cea.fr](mailto:Alexandre.Petrescu@cea.fr)

6MAN Working Group  
Internet-Draft  
Updates: RFC4291, RFC4007 (if approved)  
Intended status: Standards Track  
Expires: December 8, 2019

A. Petrescu  
CEA, LIST  
L. Velvindron  
Cyberstorm.mu  
N. Kottapalli  
Benu Networks  
G. Mishra  
Verizon Communications Inc. (VZ)  
D. Mudric  
Avaya  
June 6, 2019

The length of the prefix of an IPv6 link-local address ranges from 10 to  
127  
draft-petrescu-6man-ll-prefix-len-21

## Abstract

A rejected Erratum to RFC4291 "IPv6 Addr Archi" on the topic of link-local addresses 'would need' a draft. This draft is an answer to that need.

The length of the prefix of an IPv6 link-local address is variable. The minimal value is 10 decimal. The maximum value is 127 decimal.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 8, 2019.

## Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                                                    |    |
|----------------------------------------------------|----|
| 1. Definitions and Statements . . . . .            | 2  |
| 2. Terminology . . . . .                           | 4  |
| 3. Justification . . . . .                         | 4  |
| 4. Problem Statement . . . . .                     | 4  |
| 5. Kinds of Solutions . . . . .                    | 5  |
| 6. Context of Documents . . . . .                  | 6  |
| 7. Context of OS Behaviour . . . . .               | 8  |
| 8. Historical Note . . . . .                       | 9  |
| 9. Example of use of LL Prefix Length 32 . . . . . | 9  |
| 10. Use-Cases . . . . .                            | 10 |
| 10.1. Use-Case Convoy . . . . .                    | 10 |
| 10.2. Intuitive Next-Hop . . . . .                 | 12 |
| 11. Security Considerations . . . . .              | 12 |
| 12. IANA Considerations . . . . .                  | 13 |
| 13. Contributors . . . . .                         | 13 |
| 14. Acknowledgements . . . . .                     | 13 |
| 15. Normative References . . . . .                 | 13 |
| Appendix A. ChangeLog . . . . .                    | 13 |
| Authors' Addresses . . . . .                       | 15 |

## 1. Definitions and Statements

The prefix of an IP address is formed by the  $n$  leftmost bits of the address. (in a left-to-right writing system).

The prefix of an IP address is used for goals such as: identify the type of an IPv6 address (link-local, global, others), identify the belonging of an IP address to a particular subnetwork, assist the forwarding (or not forwarding) decisions, and others.

The minimal length of the prefix of an IPv6 link-local address (the value of  $n$ ) is equal to 10 decimal. The maximum is 127.

The prefix of an IPv6 link-local address is represented textually as "fe80::/n", where  $n$  MAY be any value between 10 and 127.

Regardless of the prefix length, the leftmost 10 bits of an IPv6 link-local address MUST be set to binary 1111111010 (hexadecimal fe80).

The RFC 4291 illustration of an IPv6 link-local address is:

| 10 bits    | 54 bits | 64 bits      |
|------------|---------|--------------|
| 1111111010 | 0       | interface ID |

Figure 1: The IPv6 link-local address

There is an error in this RFC 4291 illustration. The error is in requiring the 54 bits to be 0. The bits at position 11 to 16 are not 0 (the first 6 bits of the 54 bits). If they were 0 then 0xFEAF::1/10 were an invalid link-local address, whereas it is.

The better illustration of an IPv6 link-local address is:

| leftmost 10 bits | Subnet ID and Interface ID 118 bits |
|------------------|-------------------------------------|
| 1111111010+      | Bits that MAY be either 0 or 1      |

Figure 2: The IPv6 link-local address, better illustration

Examples: fe80::1/10, fe80:1::1/32, fe80::1:1/64 are all IPv6 link-local addresses; their prefix lengths are 10, 32 and 64 respectively. Also, 0xfeaf::1/10, 0xfebf:1::1/82 are also valid IPv6 link-local addresses (remark no 'fe80'). Each such IPv6 address has the leftmost 10 bits equal to binary 1111111010.

A notation difficulty: the number binary 1111111010 can not be written in hexadecimal without specifying the number of significant bits (fe80::/10); yet that does not make it a 'prefix'. Converting 1111111010 to hexadecimal leads to 3FA (because in a left-to-right writing system the leading 0s before comma are irrelevant); yet '3FA' is not commonly known to be the leading bits of an IPv6 link-local address, fe80::/10 is.

## 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 RFC 2119 [RFC2119].

**prefix:** a contiguous string of bits valid for forwarding operations and for subnet formation. A prefix, in general, MUST have an integer length value from 1 to 127 (except when the prefix length is for default route, in which case the value is 0) and a prefix length must be indicated in its textual representation (e.g. 2001:db8::/32 is the prefix and 32 is the prefix length).

**textual representation of a prefix:** e.g. fe80::/64.

**n leading bits:** the first n bits in a string of bits read from left to right in a writing system that is read left-to-right. E.g. the 10 leading bits of the fe80::/64 textual representation of the IPv6 link-local prefix are 1111111010.

## 3. Justification

One justification is the following: in a managed network the administrator configures link-local addresses with the 54 bits set. The network runs dynamic routing protocols OSPF, ISIS and EIGRP. The adjacency must work in a mixed vendor environment.

## 4. Problem Statement

IPv6 link-local addresses are typically self-configured according to 4 RFCs and relying on the fe80::/10 IANA allocation, RFC4291 54 0 bits, and RFC2464 MAC-based 64bit Interface IDs.

In some cases, it is advantageous to manually configure link-local addresses. Manual configuration is useful for easy remembering by humans, and for parameter resilience during network interface replacement (set addresses in computer startup configuration files). Further, the manual configuration of addresses can be scripted by automated software for rapid prototyping; still, this automated formation of addresses is not the 'self-configuration' described in the 4 RFCs mentioned previously.

A self-configured link-local address according to the 4 RFCs is of the form fe80::64bitIID; an example of such address is fe80::dabc:fe13:5246:7109. This address is difficult to remember for humans because each of the 16 hex characters appears, and the appearance is disordered. Not only it is difficult to remember but it takes long to type. This is a problem on small screens and mouse-

less keyboards. An easy to remember and type link-local address is, for example, fe80:1::1.

Manual configuration of an LL address may use short IID and Subnet ID. The Subnet ID presence in the link-local address is useful in some wireless settings where the subnet structure parameters depend on the link locality. Other settings may also benefit.

When manually setting the link-local address it is necessary to know the length of the prefix of the subnet on which this link-local address is present. This length is necessary for on-link determination.

The problem is: manually setting a prefix length other than 64 to link-local addresses may provoke glitches.

## 5. Kinds of Solutions

Some solutions to the problem are: use an address of the form fe80:1::2/32, or use an address of the form fe80::1:2/64, where 1 is the Subnet ID and 2 is the Interface ID. Other solutions involve a hidden 'scope\_id' and the use of special syntax ('%') to denote an interface. Each of these solutions have other problems of their own: set some of the 54 mandatorily reset bits of RFC4291, not implementable on some OS; invade the IID with a Subnet ID, and potentially others.

Invading the IID with a Subnet ID happens in the following situation: if fe80::1:2 assumes fe80::/64 as prefix length, then it is impossible for '1' to be a Subnet ID. A Subnet ID must be covered by a prefix length, otherwise routing and on-link determination dont work. One cant have fe80::/64 as prefix, and '1' as prefix, and a 64bit ::1:2 Interface ID.

The 'scope\_id' is 'hidden' in some operating systems; this hide is known by noticing that the use of 'scope\_id' is not mandatory for LL addresses; instead of using 'scope\_id' it is possible to rely on the interface name. Some ifconfig commands on some OSs rely on the interface name and dont require the use of a 'scope\_id' (%) parameter. It is the case for linux and Windows.

In practice, the use of fe80::1:2 was tried. It used the 64bit prefix length. But it does not perform on-link determination meaningfully (the '1' is part of IID, not of Subnet ID).

Another solution is: use Unique Local Addresses RFC 4193. For example: Carl-front interface uses fc00:1::1, Carl-rear fc00:1::2, Car2-front fc00:2::1. A pseudo random number would rather be

generated for Global ID, when in production. A kind of solution involving ULAs has interesting properties, yet the ULA-addressed packets may be forwarded across subnets. This forwarding may be an inconvenient in some setting. The use of other than LL addresses, i.e. GUAs or ULAs; this has some advantages and some inconvenients (cant put LL in src of RA).

Other solutions involve the use of an 'fe80' prefix in the RA such that to configure link-local prefixes by a similar means than GUAs/ULAs. This also has advantages and drawbacks.

Another solution is: use DNS to hold long interface IDs and Subnet IDs. Such solutions recommend the use of name-to-address mapping, instead of easy to remember LLs; DNS is such tool; can be used in order to facilitate the remembering by humans. However, this has some advantages and some inconvenients (e.g. needs DNS-SD, mDNS and IP multicast routing for multi-subnets; chicken-and-egg between formation of LLs needing these DNS tools to work in the first place). A particular inconvenient is the movement of the problem instead of solving it: upon interface change (replace faulty interface with a new one) one has configure the DNS configuration files with a new pair name-address, instead of needing to configure startup scripts.

## 6. Context of Documents

draft-bourbaki-6man-classless-ipv6-05 describes the motivation of considering IPv6 to be classless. It gives a little bit of history of why it is how it is. It proposes the rigid 64 IID length to be probably the last remnant of the boundary.

draft-farmer-6man-routing-64-02 describes the relationship between routing and the 64-bit boundary; mainly GUA, no LL; t is ambiguous in its recommendation.

draft-farmer-6man-exceptions-64-09 describes the exceptions to the standard subnet boundary in IPv6 addressing; mainly about GUA, not about LL; the exceptions are: GUA with the first 3 bits 0, manually config'ed addresses, DHCPv6 assigned addresses, ND on-link determination, IPv6-over-foo.

A memo describes the use of IPv6 link-local addresses in applications. The filename of the Internet Draft is draft-smith-ipv6-link-locals-apps-00.

RFC7404 describes "Using Only Link-Local Addressing inside an IPv6 Network".



The RFC "IPv6 Address Archi" illustrates the format of the link-local addresses. From the illustration it MAY be understood that the length of the link-local prefix is 10 bits of value 1111111010 and 54 0 bits.

IANA lists the "IPv6 prefix", and "Address Block", to be "fe80::/10" on its website. It is possible that in the future the IETF could decide to use the bits 11-53.

The RFC 2464 "IPv6-over-Ethernet" states that the prefix for link-local addresses is "fe80::/64".

RFC 6874, "Representing IPv6 Zone Identifiers in Address Literals and Uniform Resource Identifiers" specifies the link-local addresses to be under prefix "fe80::/10".

RFC 8415 "DHCPv6" considers that link-local addresses are designated by the prefix fe80::/10.

RFC 4007 "IPv6 Scoped Address Architecture" discusses Zone ID. A Zone ID may be used - internally - in the 54bits of a link-local address, even though these 54bits appear to be reset. The document mentions at a point that fe80::1 could be used in two separate physical links (not virtual, like the loopback).

RFC4291 requires that an IPv6 link-local address be assigned on each interface. Yet, it does not require the link-local prefix to be associated to an interface.

RFC4861 requires that the link local prefix be present in the Prefix List associated with an interface, although it does not specify the length of the link local prefix.

RFC4862 "SLAAC" defines how GUAs and LLs self-configure. Whereas the GUA gets its prefix length from the RA (not from an RFC), the LL gets it from RFC4291 (not from RA). They are independent choices based on distinct sources.

Several knowledgeable interpretations state that, generally speaking, the prefix length of link-local addresses is 10, but it is 64 in the particular case of Stateless Address-Autoconfiguration (SLAAC). In this latter case, the prefix is named a "subnet prefix", or "prefix on a link", and it is "fe80::/64".

The term "link-local prefix" is sometimes used to mean the prefix for on-link determination, and is sometimes used to mean the reserved address space for link-local addresses (including all current and future use). The latter is fe80::/10. Of which the address

architecture spec only gives the addresses that match fe80::/64 the standard format (by specifying intermediate 54 bits are all 0). As a result the former is (currently) only fe80::/64.

For people in the RIR world it's a common thing: you get a prefix from the RIR and then make assignments from it for specific purposes. You can route the aggregate allocation, but you're not allowed to use the unassigned parts (until you make an assignment). In this case fe80::/10 is the allocation and fe80::/64 is the assignment.

## 7. Context of OS Behaviour

Interpretations of the situation of linux working ok with fe80:1::1 call it a violation of standards.

Independent testing shows that 'ifconfig add fe80:1::1' works on linux but fails on openbsd.

A command to assign fe80:1::1 on a Cisco router works ok.

On Cisco platforms IOS, XE ver 16.x, XR and NXOS it is possible to populate the entire set of 54 bits with one's (instead of the 0s requested by RFC 4291). A test was performed between such Cisco routers running OSPF. The OSPF neighbors were still coming up. The error that was expected (and that did not happen) was that some glitches may appear due to the lack of textual compression of 54bits into '::'. However, there is a caveat: it is unknown what might happen with OSPF in a mixed environment, where not only Cisco is used.

The address fe80::1/128 is present on the loopback interface of BSD.

Implementations of an IPv6 stack in a particular operating system (linux) allow for the manual configuration of both prefix lengths 64 and 10 for link-local addresses.

In another operating system the prefix length for link-local addresses can not be explicitly specified by the end user, but may be indirectly derived from two distinct textual formats by using an unspecified rule.

In yet another operating system (FreeBSD) an end user can not use a link-local address whose value is fe80:1::1; because in that OS the hosts drop incoming packets whose source or destination address matches fe80::/10 and contains a non-0 value in bits 15-31 (like fe80:1::1 does). The URL of the C code in OpenBSD that leads to make that packet drop is [https://github.com/openbsd/src/blob/master/sys/netinet6/ip6\\_input.c](https://github.com/openbsd/src/blob/master/sys/netinet6/ip6_input.c)

In a particular operating system (openbsd), it is possible to run SLAAC with Interface IDentifiers of length different than 64, e.g. 100; this implements RFC7217. In that same operating system it is not possible to use an Interface Identifier of length 100. At the same time, in another operating system (linux) it is possible to use Interface IDentifiers of length 100, yet SLAAC does not work with IID that is not 64. In an ideal linux-bsd operating system any length of IID would be possible.

On Windows 10 Operating System it is accepted to set fe80:1::1/32 address on a physical network interface, by using the Graphical User Interface.

On MAC OS Operating System it is not possible to set fe80:1::1/32 in the command line; the 'ifconfig en1 fe80:1::1/32' command reports 'bad value'. It also reports 'bad value' with just 'fe80:1::1' (remark - no prefix length specified; note that on linux OS, when the user does not specify the prefix length to an ifconfig command, the OS will make a prefix length of value 128, and the ifconfig command will succeed.)

The loopback interface is required to have a link-local address too (RFC4291), although some OSs dont (linux). The RFC4007 clarifies that, somehow.

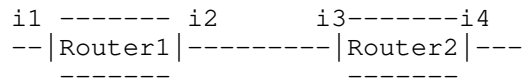
Misconfigurations and lack of interoperability MAY arise between computers that use mixed prefix lengths for link-local addresses.

## 8. Historical Note

Historical note: earlier, the link-local prefix fe80::/10 and site-local prefix fec0::/10 were grouped into a common fe80::/9. If bits 10-64 were 0 then the prefix was a link-local, otherwise a site-local. The site-local addresses were later deprecated by RFC 3879.

## 9. Example of use of LL Prefix Length 32

This figure shows two routers each with two interfaces; one such interface is connected to the other router; there are two interfaces that point elsewhere.



i2 address is fe80:12::1:1/32 ('12' means subnet between R1 and R2, '1' is R1, 2nd '1' is 'front' interface)  
 i3 address is fe80:12::2:2/32

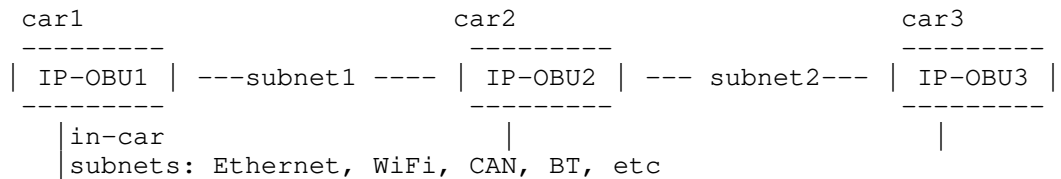
Figure 3: Figure

One router's interface (connected to the other router) uses address fe80:12::1:1/32 and the other router's corresponding interface uses address fe80:12::2:2/32.

## 10. Use-Cases

### 10.1. Use-Case Convoy

The topology in a linear convoy of cars, in V2V manner is like this:



(subnet1 is on 5880 MHz, subnet2 is on 5890 MHz)

(in the triangular convoy the figure is different)

Figure 4: Figure

Details about the restrictions with the current LL definition in the above topology:

In the above topology the restrictions with RFC4291 definitions, and the FreeBSD implementations are the following:

- RFC4291 needs 64bit MAC-based IIDs on the LLs on subnet1 and subnet2. The inconvenients of these restrictions are the following:
  - 64bit IIDs are too long to remember and type; easy to remember addresses are good for network debugging.
  - MAC-based IIDs may have some privacy risk; attackers on the road may listen to these IIDs (they are sent outside the car) and make associations that may help tracking users, like web cookies do.
  - RFC 4291 54 0 bits make it impossible to assign subnet-specific link-local addresses to subnet1 and subnet2.
- A RFC4291-compliant link-local address, like fe80::IID/64, assigned to an interface on subnet2, and replying from a ping from subnet1, does not give ensurance that subnets (on 5880 MHz or on 5890 MHz) are set up wrongly. It may be that the channels are set wrong (subnets are set up wrongly) as much as it may be that that fe80::IID/64 is in the same subnet as the pinger and the channels are right.

On another hand, if the LL addresses were like fe80:1::X on subnet1 and fe80:2::Y on subnet 2, then a ping issued from subnet1 to fe80:2::X and replying OK means clearly that the channels are set wrongly.

RFC4291 54 0 bits prevent this use of subnet-specific LL addresses.

- FreeBSD OS:
- forbids the manual assignment of LL addresses on interfaces (it is impossible to ifconfig add fe80::2 on an interface).
- FreeBSD OS does not implement OCB mode. OCB mode is an essential kind of link in vehicular communications.

Figure 5: Restrictions

Expected improvements:

- human users type short LL addresses, like fe80:1::1 instead of long to type addresses like fe80::IID64bit
- use fe80:1::1 and fe80:2::1 in two distinct subnets; if a ping from fe80:1::1 to fe80:1::2 that does not reply means the channels are wrong; otherwise (with fe80::IID) it is impossible to say whether the channels are wrong or that wrong address was used to ping (all fe80::IID64bit) look the same to a human - they are 'random').
- BSD allowing manual configuration of LL addresses may have other benefits outside the OCB context

Figure 6: Improvements

## 10.2. Intuitive Next-Hop

In some IP networks only link-local addresses are used as next hops, as described in RFC 7404. The next hop is part of an entry in the routing table. Make the next hop intuitive. The next hop is routinely used by sysadmin to ping and check whether it is reachable. Making the next hop intuitive can be achieved by mapping the global unicast address into both the subnet and interface id fields; in the past, experience shown that substituting just 'fe80' for the first 16 bytes of a GUA (such that to transform a GUA into an LL, to be used as next hop) ended up bleeding into the 54 0 bits required by RFC 4291.

## 11. Security Considerations

The clarification of the definition of the prefix length of the IPv6 link-local prefix at IANA is: call it 'leading bits' and not 'prefix', or state that the IPv6 prefix length of link-local addresses is 10 decimal. This clarification has beneficial impact in the algorithm implementation for calculation of the opaque and stable Interface Identifiers for IPv6 link-local addresses. It also positively impacts some implementations of IPv6 forwarding.

A prefix length of value 65 would lead to an Interface ID length of value 63; a 63 bit IID would provide less privacy protection than a 64 bit IID, but more than a 62 bit IID. A prefix length of value 127 would lead to a 1 bit IID which would provide almost no privacy protection.

## 12. IANA Considerations

IANA is requested to change the name of the column head in the table that depicts the "Internet Protocol Version 6 Address Space". The name should be "The n leading bits of an address" instead of "IPv6 Prefix".

The desired effect of this change is that the IPv6 link-local prefix be "fe80::/n" and that the 10 leading bits of this prefix be 1111111010. A second effect would be that the textual representation "fe80::/10" as an IPv6 link-local prefix would disappear from that IANA page.

## 13. Contributors

Listed from 6man WG discussion.

## 14. Acknowledgements

The following persons are acknowledged for the discussion that is reflected in this draft. Not all points are reflected. Some points are copied almost entirely.

Ole Troan, Scott Timothy Morizot, Brian Carpenter, Fred Baker, Mark Smith, Peter Occil, Philip Homburg, Albert Manfredi, &#144;\_&#150;3/4 &#146;B&#141;AE (TATUYA Jinmei), Fernando Gont, Christian Huitema, Simon Hobson, Matthew Petach, Yucel Guven, Sander Steffann, Dennis Ferguson, Musa Stephen Honlue, Fred Templin, Gyan Mishra, Yu Tianpeng, Darren Dukes, Dusan Mudric.

Peter Paluch submitted the Erratum suggestion to RFC 4291 about link-local addresses, and Brian Haberman rejected it, by noting 'would need' a draft. Igor Lubashev pointed to that Erratum.

## 15. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

## Appendix A. ChangeLog

The changes are listed in reverse chronological order, most recent changes appearing at the top of the list.

-21: clarified that the prefix length ranges from 1 to 127 - in general (in other cases than link local prefix).

-20: added brief explanation of IID len and privacy; added more examples of valid IPv6 link-local addresses; added an explanation of why the RFC 4291 figure of a link local address has errors.

-19: updated authorship.

-18: updated an author address; added a justification about the support of 54 set bits in LL addresses in a mixed vendor environment; added an illustration of the RFC4291 link-local address.

-17: added a new use-case for sysadmins in need of an intuitive LL address (to check with ping) used for next-hop of routing protocols.

-16: added a description of the behaviour of ifconfig fe80::1::1/32 on MAC and Windows 10 Operating Systems; added a suggestion about the use of ULA prefixes instead of LL prefixes; added a reference to an RFC 7404 about the use of only LL addresses in an IPv6 network; explained the result from practice of the use of 'fe80::1:2/64'; explained why the text says 'hidden' for '%' on some OSs; mentioned the DNS kind of solutions; added explanation of manual configuration and automation; added explanation of an example of complex to remember and type link-local addresses; added explanation of why DNS solution is a problem movement, not problem resolution.

-15: added references to draft-farmer-6man-exceptions-64-09 and draft-farmer-6man-routing-64-02, and interpreted them; added explanations of the solutions mentioned in WG discussion; added a use-case of car convoy with details about current restrictions of LL addressing and how a variable len plen for LL can improve the situation.

-14: updated authorship.

-13: added a Problem Statement section; added the name of the Organisation of one co-author; distinguished between 'need' and 'would need' a draft.

-12: the '64' in GUA vs '64' in LL issued by distinct sources: RA vs RFC4291 respectively; the address fe80::1/128 is present on the loopback interface of BSD; detailed, again, the distinction for 'on-link' determination; detailed, again, the distinction between 'assignment' and 'allocation'; added the fact that Cisco supports manual assignment of fe80::1.

-11: trying the attribute updates=RFC4291,RFC4007 in the rfc tag.

-10: syntax error corrected; more explanation about how FreeBSD C code blocks fe80::1; clarification in IANA section, but doubtful.



-09: added a reference to RFC 4007 about Zone ID in LL; added a reference to draft-bourbaki about IPv6 being classless; added the result of independent testing showing ifconfig add fe80:1::1 works on linux but fails on BSD; added URL to C code in BSD flavor that may be in charge of dropping packets whose src/dst is an LL like fe80:1::1; added two co-authors.

-08: added explanation of which RFC requires the LL address to be present, and which requires the LL prefix to be present; named the OSs, instead of staying generic; explained that the lack of requirement of ll address on lo in RFC4291 is covered by another RFC4007; explained that openbsd allows variable len IID for GUAs but not for LLs, yet linux allows the reverse, and concluded on an obvious ideal.

-07: added the fact that DHCPv6 spec considers the link-local addresses to be fe80::/10; added a valuable explanation of ll behaviour of a particularly important OS.

-04: added an example advantage of using prefix length 32.

-03:

-02: corrected a typo in "fe80::/1" and added a 7-bit encoding for one persons name (in addition to the japanese-shift-jis encoding which is not understood by xml2rfc.)

#### Authors' Addresses

Alexandre Petrescu  
CEA, LIST  
CEA Saclay  
Gif-sur-Yvette , Ile-de-France 91190  
France

Phone: +33169089223  
Email: Alexandre.Petrescu@cea.fr

Loganaden Velvindron  
Cyberstorm.mu  
street  
city , region code  
Mauritius

Phone: +phonenummer  
Email: loganaden@gmail.com

Naveen Kottapalli  
Benu Networks  
street  
City , Region code  
United States

Phone: +phonenumber  
Email: naveen.sarma@gmail.com

Gyan S. Mishra  
Verizon Communications Inc. (VZ)  
13101 Columbia Pike FDC1 Rm 304-D  
Silver Spring MD 20904  
United States

Phone: 301 502-1347  
Email: gyan.s.mishra@verizon.com

Dusan Mudric  
Avaya

Email: dmudric@avaya.com

IPv6 Maintenance  
Internet-Draft  
Intended status: Standards Track  
Expires: April 4, 2019

L. Colitti  
E. Kline  
J. Linkova  
Google  
October 1, 2018

Discovering PREF64 in Router Advertisements  
draft-pref64folks-6man-ra-pref64-02

Abstract

This document specifies a Router Advertisement option to communicate NAT64 prefixes to clients.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 4, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                                                                    |   |
|--------------------------------------------------------------------|---|
| 1. Introduction . . . . .                                          | 2 |
| 1.1. Requirements Language . . . . .                               | 2 |
| 1.2. Terminology . . . . .                                         | 2 |
| 2. Why include the NAT64 prefix in Router Advertisements . . . . . | 3 |
| 3. Semantics . . . . .                                             | 3 |
| 4. Option format . . . . .                                         | 4 |
| 5. Handling Multiple NAT64 Prefixes . . . . .                      | 4 |
| 6. Multihoming . . . . .                                           | 5 |
| 7. IANA Considerations . . . . .                                   | 6 |
| 8. Security Considerations . . . . .                               | 6 |
| 9. Acknowledgements . . . . .                                      | 6 |
| 10. References . . . . .                                           | 6 |
| 10.1. Normative References . . . . .                               | 6 |
| 10.2. Informative References . . . . .                             | 7 |
| 10.3. URIs . . . . .                                               | 8 |
| Authors' Addresses . . . . .                                       | 8 |

## 1. Introduction

NAT64 [RFC6146] with DNS64 [RFC6147] is a widely-deployed mechanism to provide IPv4 access on IPv6-only networks. In order to support functions such as local validation of DNSSEC [RFC4033] responses, 464xlat [RFC6877], and local IPv4 address synthesis [RFC8305], the host must be aware of the NAT64 prefix in use by the network. This document specifies a Router Advertisement [RFC4861] option to communicate the NAT64 prefix to hosts.

### 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

Pref64: an IPv6 prefix used for IPv6 address synthesis [RFC6146];

RA Router Advertisement, a message used by IPv6 routers to advertise their presence together with various link and Internet parameters ([RFC4861]);

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

## 2. Why include the NAT64 prefix in Router Advertisements

Fate sharing: NAT64 requires a routing to be configured. IPv6 routing configuration requires receiving an IPv6 Router Advertisement [RFC4861]. Compared to currently-deployed NAT64 prefix discovery methods such as [RFC7050], including the NAT64 prefix in the Router Advertisement minimizes the number of packets required to configure a host. This speeds up the process of connecting to a network that supports NAT64/DNS64, and simplifies host implementation by removing the possibility that the host can have an incomplete layer 3 configuration (e.g., IPv6 addresses and prefixes, but no NAT64 prefix).

Deployability: all IPv6 hosts and networks are required to support [RFC4861]. Other options such as [RFC7225] require implementing other protocols.

## 3. Semantics

For simplicity, this option only supports a NAT64 prefix length of 96 bits, as this is by the most common configuration used by hosts. Networks using one of the other prefix lengths supported in ([RFC6052]) can use other mechanisms such as [RFC7050] or [RFC7225]. If different prefix lengths become common, another RA option can be created to configure them.

This option may appear more than once in a Router Advertisement. Host behaviour with regards to synthesizing IPv6 addresses from IPv4 addresses SHOULD follow the recommendations given in Section 3 of [RFC7050], limited to the NAT64 prefixes that have non-zero lifetime.

This option specifies exactly one NAT64 prefix for all IPv4 destinations. If the network operator desires to route different parts of the IPv4 address space to different NAT64 devices, this can be accomplished by routing more specifics of the NAT64 prefix to those devices. For example, if the operator would like to route 10.0.0.0/8 through NAT64 device A and the rest of the IPv4 space through NAT64 device B, and the operator's NAT64 prefix is 2001:db8:a:b::/96, then the operator can route 2001:db8:a:b::a00:0/104 to NAT64 A and 2001:db8:a:b::/64 to NAT64 B.

In a network that provides both IPv4 and NAT64, it may be desirable for certain IPv4 addresses not to be translated. An example might be private address ranges that are local to the network and should not be reached through the NAT64. This type of configuration cannot be conveyed to hosts using this option, or through other NAT64 prefix provisioning mechanisms such as [RFC7050] or [RFC7225]. This problem does not apply in IPv6-only networks, because in such networks, the

host does not have an IPv4 address and cannot reach any IPv4 destinations without the NAT64.

#### 4. Option format

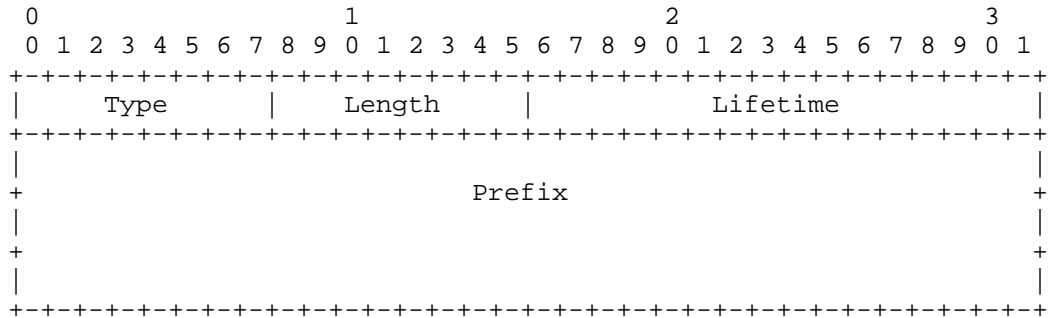


Figure 1: NAT64 Prefix Option Format

#### Fields:

- Type 8-bit identifier of the RDNSS option type as assigned by IANA: TBD
- Length 8-bit unsigned integer. The length of the option (including the Type and Length fields) is in units of 8 octets. The sender MUST set the Length to 2. A host MUST ignore the NAT64 prefix option if the length field value is 1. If the Length field value exceeds 2, the host MUST utilize the first 16 octets and ignore the rest of the option.
- Lifetime 16-bit unsigned integer. The maximum time in seconds over which this NAT64 prefix MAY be used. The value of Lifetime SHOULD by default be set to lesser of 3 x MaxRtrAdvInterval or 65535 seconds. A value of zero means that the prefix MUST no longer be used.
- Prefix The 96-bit NAT64 prefix.

#### 5. Handling Multiple NAT64 Prefixes

In some cases a host may receive multiple NAT64 prefixes from different sources. Possible scenarios include (but are not limited to):

- o the host is using multiple mechanisms to discover Pref64 prefixes (e.g. by using PCP ([RFC7225]) and/or by resolving IPv4-only fully

qualified domain name ([RFC7050]) in addition to receiving the Pref64 RA option);

- o The pref64 option presents in a single RA more than once;
- o the host receives multiple RAs with different Pref64 prefixes on one or multiple interfaces.

When multiple Pref64 were discovered via RA Pref64 Option (the Option presents more than once in a single RA or multiple RAs were received), host behaviour with regards to synthesizing IPv6 addresses from IPv4 addresses SHOULD follow the recommendations given in Section 3 of [RFC7050], limited to the NAT64 prefixes that have non-zero lifetime..

When different Pref64 are discovered by using multiple mechanisms, hosts SHOULD select one source of information only. The RECOMMENDED order is:

- o PCP-discovered prefixes ([RFC7225]), if supported;
- o Pref64 discovered via RA Option;
- o Pref64 resolving IPv4-only fully qualified domain name ([RFC7050])

Note that if the network provides Pref64 both via this RA option and [RFC7225], hosts that receive the Pref64 via RA option may choose to use it immediately before waiting for PCP to complete, and therefore some traffic may not reflect any more detailed configuration provided by PCP.

## 6. Multihoming

Like most IPv6 configuration information, the Pref64 option is specific to the network on which it is received. For example, a Pref64 option received on a particular wireless network may not be usable unless the traffic is also sourced on that network. Similarly, a host connected to a cellular network that provides NAT64 generally cannot use that NAT64 for destinations reached through a VPN tunnel that terminates outside that network.

Thus, correct use of this option on a multihomed host generally requires the host to be PVD-aware.

This issue is not specific to the Pref64 RA option and, for example, is quite typical for DNS resolving on multihomed hosts (e.g. a host might resolve a destination name by using the corporate DNS server

via the VPN tunnel but then send the traffic via its Internet-facing interface).

## 7. IANA Considerations

The IANA is requested to assign a new IPv6 Neighbor Discovery Option type for the PREF64 option defined in this document.

| Option Name   | Type  |
|---------------|-------|
| PREF64 option | (TBD) |

Table 1

The IANA registry for these options is:

<https://www.iana.org/assignments/icmpv6-parameters> [1]

## 8. Security Considerations

Because Router Advertisements are required in all IPv6 configuration scenarios, on IPv6-only networks, Router Advertisements must already be secured, e.g., by deploying RA guard [RFC6105]. Providing all configuration in Router Advertisements increases security by ensuring that no other protocols can be abused by malicious attackers to provide hosts with invalid configuration.

The security measures that must already be in place to ensure that Router Advertisements are only received from legitimate sources eliminate the problem of NAT64 prefix validation described in section 3.1 of [RFC7050].

## 9. Acknowledgements

Thanks to the following people (in alphabetical order) for their review and feedback: Brian E Carpenter, Nick Heatley, David Schinazi.

## 10. References

### 10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.



- [RFC6052] Bao, C., Huitema, C., Bagnulo, M., Boucadair, M., and X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, DOI 10.17487/RFC6052, October 2010, <<https://www.rfc-editor.org/info/rfc6052>>.

## 10.2. Informative References

- [I-D.ietf-intarea-provisioning-domains] Pfister, P., Vyncke, E., Pauly, T., Schinazi, D., and W. Shao, "Discovering Provisioning Domain Names and Data", draft-ietf-intarea-provisioning-domains-02 (work in progress), June 2018.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <<https://www.rfc-editor.org/info/rfc4033>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [RFC6105] Levy-Abegnoli, E., Van de Velde, G., Popoviciu, C., and J. Mohacsi, "IPv6 Router Advertisement Guard", RFC 6105, DOI 10.17487/RFC6105, February 2011, <<https://www.rfc-editor.org/info/rfc6105>>.
- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <<https://www.rfc-editor.org/info/rfc6146>>.
- [RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, DOI 10.17487/RFC6147, April 2011, <<https://www.rfc-editor.org/info/rfc6147>>.
- [RFC6877] Mawatari, M., Kawashima, M., and C. Byrne, "464XLAT: Combination of Stateful and Stateless Translation", RFC 6877, DOI 10.17487/RFC6877, April 2013, <<https://www.rfc-editor.org/info/rfc6877>>.
- [RFC7050] Savolainen, T., Korhonen, J., and D. Wing, "Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis", RFC 7050, DOI 10.17487/RFC7050, November 2013, <<https://www.rfc-editor.org/info/rfc7050>>.

- [RFC7225] Boucadair, M., "Discovering NAT64 IPv6 Prefixes Using the Port Control Protocol (PCP)", RFC 7225, DOI 10.17487/RFC7225, May 2014, <<https://www.rfc-editor.org/info/rfc7225>>.
- [RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <<https://www.rfc-editor.org/info/rfc7556>>.
- [RFC8305] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency", RFC 8305, DOI 10.17487/RFC8305, December 2017, <<https://www.rfc-editor.org/info/rfc8305>>.

### 10.3. URIs

- [1] <https://www.iana.org/assignments/icmpv6-parameters>

#### Authors' Addresses

Lorenzo Colitti  
Google  
Roppongi 6-10-1  
Minato, Tokyo 106-6126  
JP

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

Erik Kline  
Google  
Roppongi 6-10-1  
Minato, Tokyo 106-6126  
JP

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

Jen Linkova  
Google  
1 Darling Island Rd  
Pyrmont, NSW 2009  
AU

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

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: August 24, 2019

C. Qin, Ed.  
Y. Xi  
W. Xu  
Alibaba-Inc  
February 20, 2019

IPv6 Router Advertisement Option for Network Boot  
draft-qin-6man-nb-option-03

Abstract

This document specifies an IPv6 Router Advertisement (RA) option (called "Boot File URL option") to allow IPv6 routers to advertise configuration information for booting a node from the network.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 24, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                                                                |   |
|----------------------------------------------------------------|---|
| 1. Introduction . . . . .                                      | 2 |
| 1.1. Requirements Language . . . . .                           | 2 |
| 2. Option . . . . .                                            | 3 |
| 2.1. Boot File Uniform Resource Locator (URL) Option . . . . . | 3 |
| 3. Implementation Considerations . . . . .                     | 4 |
| 4. IANA Considerations . . . . .                               | 4 |
| 5. Security Considerations . . . . .                           | 5 |
| 6. Acknowledgements . . . . .                                  | 5 |
| 7. References . . . . .                                        | 5 |
| 7.1. Normative References . . . . .                            | 5 |
| 7.2. Informative References . . . . .                          | 6 |
| Authors' Addresses . . . . .                                   | 6 |

## 1. Introduction

This document describes an IPv6 Neighbor Discovery (ND) option (called BFURL option) that can be used to provide configuration information for nodes to be booted from network instead of local media.

IPv6 Stateless Address Autoconfiguration (SLAAC) [RFC4862] and IPv6 Neighbor Discovery (ND) [RFC4861] define ways to configure IPv6 addresses, on-link prefix list, default routers and other parameters.

The existing ND message (RA) is used to carry this network boot information. Nodes can get the boot file url and parameters through RA messages via BFRUL option. A boot file can be a boot-loader program or a minimal OS kernel. The node firmware needs to download the boot file and execute it.

This approach is useful in networks with no DHCPv6 infrastructure. The intention is to simplify the implementation of first-stage communicating functionalities of the nodes (i.e. PXE firmware), and network. The distribution of additional information and subsequent communications between the node and network side (e.g., an install server) should be handled by applications built in the boot file.

The configuration of a Boot-File-URL would be required onto routers sending RA messages. The configuration mechanism (manual or automatic) is out of scope of this document.

## 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. Option

The option formats comply with ND options per [RFC4861].

### 2.1. Boot File Uniform Resource Locator (URL) Option

Routers send this option to nodes with a URL to a boot file.

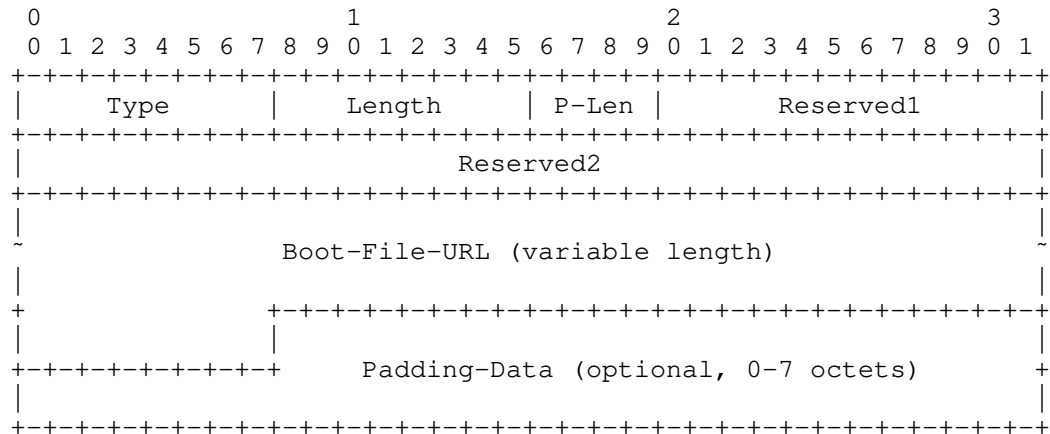


Figure 1

Fields description:

|               |                                                                                                                |
|---------------|----------------------------------------------------------------------------------------------------------------|
| Type          | TBD.                                                                                                           |
| Length        | 8-bit unsigned integer. The length of this option (including the Type and Length fields) in units of 8 octets. |
| P-Len         | 4-bit unsigned integer. The length of Padding-Data field in units of octets.                                   |
| Reserved1     | This field is unused. It MUST be initialized to zero and MUST be ignored by the receiver.                      |
| Reserved2     | This field is unused. It MUST be initialized to zero and MUST be ignored by the receiver.                      |
| Boot-File-URL | The URL for a boot file in string. Its format MUST comply with [RFC3986].                                      |
| Padding-Data  | 0-7 octets of zeros for padding the encoding of Boot-File-URL if required. Since the                           |

length of option MUST be a multiple of 8 octets, for the minimum multiple, the remaining octets following the encoding part of Boot-File-URL MUST be padded with zeros.

### 3. Implementation Considerations

In the current format of BFURL option (Section 2.1), there is no field defined to identify the architectures of nodes to be booted from network. Implementers should be aware of the details of their deployment environment and tailor the boot file to accommodate the network booting nodes of different types.

A new IPv6 Router Solicitation (RS) option can be defined in the future for nodes to send the information of architecture types they support to the network side for the selection of appropriate boot file.

Also, there is no field defined in the BFURL option nor any individual option specified in this document for the network booting parameters. We recommend that, the basic parameters required should be embedded in the boot file itself. That can ease the configuration of network booting functionality on the network devices. After the boot file or the built-in application is successfully executed, they should take the responsibility of guiding the subsequent phases of installation.

This document puts no constraints on the protocols used to download the boot file. While it is possible that the downloading protocol is specified in the URL by syntax.

Domain names may be used in the boot file URL rather than an IPv6 address. That requires the network booting nodes to support DNS implementation. The DNS server information can be also distributed by RA options per [RFC8106].

### 4. IANA Considerations

This document requires a new ND option type to be allocated.

| Option Name   | Type |
|---------------|------|
| Boot-File-URL | TBD. |

## 5. Security Considerations

Rogue RA messages with wrong URL information may be received in the untrusted environment and direct the network booting nodes to download boot file from an attacker's file server. The Secure Neighbor Discovery (SEND) protocol [RFC3971] is designed to allow all ND options including the BFURL option specified in this document to be sent with digital signature, which prevent this kind of attack.

To protect the boot file downloading process, using protocols like HTTPS is recommended. Further, security mechanisms can be implemented within the built-in application or the boot file to be executed, to secure the communications in the later stage.

## 6. Acknowledgements

The authors would like to thank Jinhui, He Qiang for their comments and inputs.

## 7. References

### 7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "Secure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <<https://www.rfc-editor.org/info/rfc3971>>.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.

- [RFC8106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 8106, DOI 10.17487/RFC8106, March 2017, <<https://www.rfc-editor.org/info/rfc8106>>.
- [UEFI23] UEFI Forum, "Unified Extensible Firmware Interface (UEFI) Specification, Version 2.7 Errata A", August 2017, <<http://www.uefi.org/>>.

## 7.2. Informative References

- [RFC3315] 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, <<https://www.rfc-editor.org/info/rfc3315>>.
- [RFC4578] Johnston, M. and S. Venaas, Ed., "Dynamic Host Configuration Protocol (DHCP) Options for the Intel Preboot eXecution Environment (PXE)", RFC 4578, DOI 10.17487/RFC4578, November 2006, <<https://www.rfc-editor.org/info/rfc4578>>.
- [RFC5970] Huth, T., Freimann, J., Zimmer, V., and D. Thaler, "DHCPv6 Options for Network Boot", RFC 5970, DOI 10.17487/RFC5970, September 2010, <<https://www.rfc-editor.org/info/rfc5970>>.

## Authors' Addresses

Chao Qin (editor)  
Alibaba-Inc  
P.R.China

Email: [jacni@jacni.com](mailto:jacni@jacni.com)

Yongqing Xi  
Alibaba-Inc  
P.R.China

Email: [yongqing.xyq@alibaba-inc.com](mailto:yongqing.xyq@alibaba-inc.com)

Wei Xu  
Alibaba-Inc  
P.R.China

Email: [arthur.xw@alibaba-inc.com](mailto:arthur.xw@alibaba-inc.com)



Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 25 April 2022

T. Winters  
QA Cafe  
O. Troan  
cisco  
22 October 2021

The Universal IPv6 Configuration Option  
draft-troan-6man-universal-ra-option-06

Abstract

One of the original intentions for the IPv6 host configuration, was to configure the network-layer parameters only with IPv6 ND, and use service discovery for other configuration information. Unfortunately that hasn't panned out quite as planned, and we are in a situation where all kinds of configuration options are added to RAs. This document proposes a new universal option for RA in a self-describing data format, with the list of elements maintained in an IANA registry, with greatly relaxed rules for registration.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 25 April 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components

extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

|                                                       |   |
|-------------------------------------------------------|---|
| 1. Introduction . . . . .                             | 2 |
| 2. Conventions . . . . .                              | 3 |
| 3. Introduction . . . . .                             | 3 |
| 4. The Universal IPv6 Configuration option . . . . .  | 3 |
| 5. CBOR encoding . . . . .                            | 5 |
| 6. Implementation Guidance . . . . .                  | 5 |
| 7. Implementation Status . . . . .                    | 5 |
| 8. Security Considerations . . . . .                  | 5 |
| 9. IANA Considerations . . . . .                      | 6 |
| 9.1. Universal configuration option . . . . .         | 6 |
| 9.2. Initial objects in the registry . . . . .        | 6 |
| 9.3. Initial objects in the registry . . . . .        | 6 |
| 9.3.1. CDDL/JSON Mapping Parameters to CBOR . . . . . | 6 |
| 9.3.2. Key Registry . . . . .                         | 7 |
| 10. Normative References . . . . .                    | 8 |
| 11. Informative References . . . . .                  | 9 |
| Appendix A. Acknowledgements . . . . .                | 9 |
| Authors' Addresses . . . . .                          | 9 |

## 1. Introduction

This document proposes a new universal option for the Router Advertisement IPv6 ND message [RFC4861]. Its purpose is to use the RA messages as opaque carriers for configuration information between an agent on a router and a host.

DHCP is suited to give per-client configuration information, while the RA mechanism advertises configuration information to all hosts on the link. There is a long running history of "conflict" between the two. The arguments go; there is less fate-sharing in DHCP, DHCP doesn't deal with multiple sources of information, or make it more difficult to change information independent of the lifetimes, RA cannot be used to configure different information to different clients and so on. And of course some options are only available in RAs and some options are only available in DHCP.

While this proposal does not resolve the DHCP vs RA debate, it proposes a solution to the problem of a very slow process of standardizing new Router Advertisement options, and the IETF spending an inordinate amount of time arguing over new configuration options in Router Advertisements. It is possible in the future to use the new universal option in DHCP, since this would lead to additional conflict resolution an additional document will need to be considered for that.

## 2. Conventions

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].

Additionally, the key words "\*MIGHT\*", "\*COULD\*", "\*MAY WISH TO\*", "\*WOULD PROBABLY\*", "\*SHOULD CONSIDER\*", and "\*MUST (BUT WE KNOW YOU WON'T)\*" in this document are to be interpreted as described in RFC 6919 [RFC6919].

## 3. Introduction

This document specifies a new "self-describing" universal configuration option. Currently new configuration option requires "standards action". The proposal is that no future IETF document will be required. The configuration option is described directly in the universal configuration IANA registry.

## 4. The Universal IPv6 Configuration option

The option data is described using the schema language CDDL [RFC8610], encoded in CBOR [RFC7049].

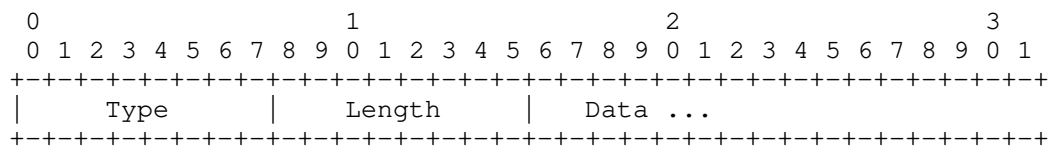


Figure 1: IPv6 Configuration Option Format

Fields:

Type: 42 for Universal IPv6 Configuration Option

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

Data: CBOR encoded data.

The Option is zero-padded to nearest 8-octet boundary.

Example of an JSON instance of the option:

```
{
 "ietf": {
 "dns": {
 "dnssl": [
 "example.com"
],
 "rdnss": [
 "2001:db8::1",
 "2001:db8::2"
]
 },
 "nat64": {
 "prefix": "64:ff9b::/96"
 },
 "rio": [
 {
 "prefix": "::/0",
 "next-hop": "fe80::1"
 },
 {
 "prefix": "2001:db8::/32",
 "next-hop": "fe80::2"
 }
]
 }
}
```

The universal IPv6 Configuration option MUST be small enough to fit within a single IPv6 ND packet. It then follows that a single element in the dictionary cannot be larger than what fits within a single option. Different elements can be split across multiple universal configuration options (in separate packets). All IANA registered elements are under the "ietf" key in the dictionary. Private configuration information can be included in the option using different keys.

If information learnt via this option conflicts with other configuration information learnt via Router Advertisement messages, that is considered a configuration error. How those conflicts should be resolved is left up to the implementation.

## 5. CBOR encoding

It is recommended that the user can configure the option using JSON. Likewise an application registering interest in an option SHOULD be able to use string keys. The CBOR encoding to save space, uses integers for map keys. The mapping table between integer and string map keys are part of the IANA registry for the option.

Values -23-23 encodes to a single byte in CBOR, and these values are reserved for IETF used map keys.

## 6. Implementation Guidance

The purpose of this option is to allow users to use the RA as an opaque carrier for configuration information without requiring code changes in the option carrying infrastructure.

On the router there should be an API allowing a user to add an element, e.g. a JSON object [RFC8259] or a pre-encoded CBOR string to RAs sent on a given interface.

On the host side, an API SHOULD be available allowing applications to subscribe to received configuration elements. It SHOULD be possible to subscribe to configuration object by dictionary key.

The contents of any elements that are not recognized, either in whole or in part, by the receiving host MUST be ignored and the remainder of option's contents MUST be processed as normal.

An implementation SHOULD provide a "JSON interface" for configuring the option.

## 7. Implementation Status

The Universal IPv6 configuration option sending side is implemented in VPP (<https://wiki.fd.io/view/VPP> (<https://wiki.fd.io/view/VPP>)).

The implementation is a prototype released under Apache license and available at: <https://github.com/vpp-dev/vpp/commit/156db316565e77de30890f6e9b2630bd97b0d61d> (<https://github.com/vpp-dev/vpp/commit/156db316565e77de30890f6e9b2630bd97b0d61d>).

## 8. Security Considerations

Unless there is a security relationship between the host and the router (e.g. SEND), and even then, the consumer of configuration information can put no trust in the information received.

## 9. IANA Considerations

IANA is requested to add a new registry for the Universal IPv6 Configuration option. The registry should be named "IPv6 Universal Configuration Information Option".

The schema field follows the CDDL schema definition in [RFC8610].

Changes and additions to the registry follow the policies below [RFC8126]:

| Range                      | Registration Procedure |
|----------------------------|------------------------|
| -23-23                     | Standards Action       |
| 24-32767                   | Specification Required |
| 32768-18446744073709551615 | Expert Review          |

Table 1

A new registration requires a new CBOR key to parameter name assignment and a CDDL definition.

### 9.1. Universal configuration option

The IANA is requested to add the universal option to the "IPv6 Neighbor Discovery Option Formats" registry with the value of 42.

### 9.2. Initial objects in the registry

The PVD [RFC8801] elements and DNS [RFC8106]) are included to provide an alternative representation for the proposed new options in that draft.

### 9.3. Initial objects in the registry

#### 9.3.1. CDDL/JSON Mapping Parameters to CBOR

| Parameter Name / JSON key | CBOR Key |
|---------------------------|----------|
| ietf                      | -23      |
| pio                       | -22      |

|                    |     |
|--------------------|-----|
| mtu                | -21 |
| rio                | -20 |
| dns                | -19 |
| nat64              | -18 |
| ipv6-only          | -17 |
| pvd                | -16 |
| prefix             | -15 |
| preferred-lifetime | -14 |
| valid-lifetime     | -13 |
| lifetime           | -12 |
| a-flag             | -11 |
| l-flag             | -10 |
| preference         | -9  |
| nexthop            | -8  |
| nssl               | -7  |
| dnss               | -6  |
| fqdn               | -5  |
| uri                | -4  |

Table 2

## 9.3.2. Key Registry

| CDDL                                                                                                                                                                                                                                                                                                                                                                         | Reference                                     |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------|
| <pre> ietf = {   ? pio : [+ pio]   ? rio : [+ rio]   ? dns : dns   ? nat64: nat64   ? ipv6-only: bool   ? pvd : pvd }  dns = {   nssl : [* tstr]   dnss : [+ ipv6-address]   lifetime : uint .size 4 }  nat64 = {   prefix : ipv6-prefix }  ipv6-only : bool  pvd = {   fqdn : tstr   uri : tstr   ? dns : dns   ? nat64: nat64   ? pio : [+ pio]   ? rio : [+ rio] } </pre> | <p>RFC8106</p> <p>RFC7050</p> <p>[v6only]</p> |

## 10. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.



- [RFC6919] Barnes, R., Kent, S., and E. Rescorla, "Further Key Words for Use in RFCs to Indicate Requirement Levels", RFC 6919, DOI 10.17487/RFC6919, April 2013, <<https://www.rfc-editor.org/info/rfc6919>>.
- [RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <<https://www.rfc-editor.org/info/rfc7049>>.
- [RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/info/rfc8610>>.

## 11. Informative References

- [RFC8106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 8106, DOI 10.17487/RFC8106, March 2017, <<https://www.rfc-editor.org/info/rfc8106>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [RFC8801] Pfister, P., Vyncke, É., Pauly, T., Schinazi, D., and W. Shao, "Discovering Provisioning Domain Names and Data", RFC 8801, DOI 10.17487/RFC8801, July 2020, <<https://www.rfc-editor.org/info/rfc8801>>.

## Appendix A. Acknowledgements

Many thanks to Dave Thaler for feedback and suggestions of a more effective CBOR encoding. Thank you very much to Carsten Bormann for CBOR and CDDL help.

## Authors' Addresses

T. Winters  
QA Cafe

Email: [tim@qacafe.com](mailto:tim@qacafe.com)

O. Troan  
cisco

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