

IPv6 Maintenance  
Internet-Draft  
Updates: 2460,7045 (if approved)  
Intended status: Standards Track  
Expires: April 8, 2016

F. Baker  
Cisco Systems  
October 6, 2015

IPv6 Hop-by-Hop Header Handling  
draft-baker-6man-hbh-header-handling-03

Abstract

This note updates the IPv6 Specification (RFC 2460), specifically commenting on the Hop-by-Hop Options Header (section 4.3) and option format and handling (section 4.2).

It also updates RFC 7045, which noted that RFC 2460 is widely violated in this respect, but merely legitimized this situation with a SHOULD. The present document tries to address the issue more fundamentally.

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 April 8, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

## Table of Contents

1. Introduction . . . . .	2
1.1. Requirements Language . . . . .	3
2. Handling of options in extension headers . . . . .	3
2.1. Hop-by_hop Options . . . . .	3
2.2. Changing options in transit . . . . .	4
2.3. Adding headers or options in transit . . . . .	5
2.4. Interactions with the Security Extension Header . . . . .	5
3. Interoperation with RFC 2460 . . . . .	5
4. IANA Considerations . . . . .	6
5. Security Considerations . . . . .	6
6. Privacy Considerations . . . . .	6
7. Acknowledgements . . . . .	7
8. References . . . . .	7
8.1. Normative References . . . . .	7
8.2. Informative References . . . . .	7
Appendix A. Change Log . . . . .	8
Author's Address . . . . .	9

## 1. Introduction

The IPv6 Specification [RFC2460] specifies a number of extension headers. These, and the ordering considerations given, were defined based on experience with IPv4 options. They were, however, prescient with respect to their actual use - the IETF community did not know how they would be used. In at least one case, the Hop-by-Hop option, most if not all implementations implement it by punting to a software path. In the words of [RFC7045],

The IPv6 Hop-by-Hop Options header SHOULD be processed by intermediate forwarding nodes as described in [RFC2460]. However, it is to be expected that high-performance routers will either ignore it or assign packets containing it to a slow processing path. Designers planning to use a Hop-by-Hop option need to be aware of this likely behaviour.

Fernando Gont, in his Observations on IPv6 EH Filtering in the Real World [I-D.ietf-v6ops-ipv6-ehs-in-real-world], and the operational community in IPv6 Operations, consider any punt to a software path to be an attack vector. Hence, IPv6 packets containing the Hop-by-Hop Extension Header (and in some cases, any extension header) get dropped in transit.

The subject of this document is implementation approaches to obviate or mitigate the attack vector, and updating the Hop-by-Hop option with respect to current issues.

### 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 [RFC2119].

## 2. Handling of options in extension headers

Packets containing the Hop-by-Hop Extension Header SHOULD be processed at substantially the same rate as packets that do not.

If a hop-by-hop header option is not implemented, or is not in use, in a given system (such as, for example, an interface that is not configured for RSVP receiving an RSVP Alert Option), the option MUST be skipped.

If a hop-by-hop header is present in a packet's extension header chain and it is not the first extension header, the packet MUST be discarded by the first system that observes the fact (Section 2.2 of [RFC7045]). This will normally be in the system using the IPv6 address in the Destination Address, as [RFC2460] precludes other routers from parsing the header chain. The only obvious exception to that is a router or firewall configured to parse the IPv6 header chain.

### 2.1. Hop-by\_hop Options

At this writing, there are several defined Hop-by-Hop options:

PAD Options: The PAD1 and PADn options [RFC2460] define empty space.

Router Alert Option: The IPv6 Router Alert Option [RFC2711] [RFC6398] is intended to force the punting of a datagram to software, in cases in which RSVP or other protocols need that to happen.

Jumbo Payload: Carries a length field for a packet whose length exceeds 0xFFFF octets. [RFC2675]

RPL Option: The RPL option carries routing information used in a RPL network[RFC6553]

Quickstart Option Identifies TCP quick-start configuration, and allows an intermediate router to reduce the configuration parameters as appropriate. [RFC4782]

Common Architecture Label IPv6 Security Option: Encodes security labels on packets [RFC5570]

SMF Option: Simplified Multicast Forwarding Option[RFC6621]

MPL Option: Supports multicast in an RPL network [I-D.ietf-roll-trickle-mcast]

DFF Option: Depth-First Forwarding [RFC6971]

There are also options that have been defined for the Destination Options header. These are not listed here.

While this is not true of older implementations, modern equipment is capable of parsing the Extension Header chain, and can be extended to perform at least a cursory examination of the Hop-by-Hop options. For example, such implementations SHOULD be able to identify and skip the PAD1 and PADn options, and perform more complicated processing only if configured by software to do so. More to the point: it isn't clear what the purpose of the JumboFrame option is if not to be understood by anyone that looks at it.

Question asked by a reviewer: "Is this configurable? How will router know that HbH needs to be skipped on one interface and not on others."

Answer: the system knows whether RSVP has been configured on an interface. When such configuration is present, it can configure the hardware with what it wants done with the Router Alert. In the absence of such configuration, hardware should be configured to skip the option if found.

## 2.2. Changing options in transit

Section 4.2 of [RFC2460] explicitly allows for options that may be updated in transit. It is likely that the original authors intended that to be very simple, such as having the originating end system provide the container, and having intermediate systems update it - perhaps performing some calculation, and in any event storing the resulting value. Examples of such a use might be in [XCP] or [RCP].

As a side comment, the Routing Header, which is an extension header rather than a list of options, is treated similarly; when a system is the destination of a packet and not the last one in the Routing

Header's list, it swaps the destination address with the indicated address in the list, and updates the hop count and the list depth accordingly.

Such options must be marked appropriately (their option type is of the form XX1XXXXX), and are excluded from checksum calculations in AH and ESP.

### 2.3. Adding headers or options in transit

Use cases under current consideration take this a step further: a router or middleware process MAY add an extension header, MAY add an option to the header, which may extend the length of the Hop-by-Hop Extension Header, or MAY process such an option in a manner that extends both the length of the option and the Extension Header containing it. The obvious implication is that other equipment in the network may not understand or implement the new option type. As such, the Option Type value of such an option MUST indicate that it is to be skipped by a system that does not understand it. Since, by definition, it is being updated in transit and not included in any AH or ESP integrity check if present, the Option Type MUST also indicate that it may be updated in transit, and so is excluded from AH and ESP processing. By implication, such an Option Type MUST be of the form 001XXXXX.

### 2.4. Interactions with the Security Extension Header

The interactions with the IP Authentication Header [RFC4302] and IP Encapsulating Security Payload (ESP) [RFC4303], as in the case of existing option uses, is minimally defined. AH and ESP call for the exclusion of mutable data in their calculations by zeroing it out prior to performing the integrity check calculation. However, in the case that network operation has changed the length of the option or the extension header, that may still cause the integrity check to fail. Specifications that define such options SHOULD consider the implications of this for AH and ESP. An option whose insertion would affect the integrity check MUST be removed prior to the integrity check, and as a result the packet restored to its state as originally sent.

## 3. Interoperation with RFC 2460

There are four possible modes of interaction with routers that don't implement the Hop-By-Hop Option in the fast path:

1. Presume that they cannot handle the Hop-By-Hop option at close to wire speed, and that's OK.

2. Presume that they will drop traffic containing Hop-By-Hop options.
3. Presume that they can handle the Hop-By-Hop option at or close to wire speed, and are configured to do so.
4. Presume that they don't exist, perhaps because older routers are configured to ignore all Hop-by-Hop options.

If the first model actually works in a given network, it may be acceptable in that domain. It is not a model that will work in the general Internet, however.

The second model (which is most probable at this writing) is a description of the general Internet in 2015.

The third and fourth models, if applicable in a given context, are what one might hope for. Vendors are in a position to either have an option to ignore the Hop-By-Hop header in older equipment, or add such an option in upgraded software (fourth model). New equipment is expected to follow the third model by implementing the recommendations in Section 2.

#### 4. IANA Considerations

This memo asks the IANA for no new parameters.

#### 5. Security Considerations

In general, modification of a datagram in transit is considered very closely from the viewpoint of the End-to-End Principle, which in this context may be summarized as "the network should do nothing that is of concern to the communicating applications or introduces operational issues." The concept of changing the length of an Extension Header or an option contained within it (Section 2.3) is of concern in that context. The obvious concern is around the interaction with AH or ESP, and a less obvious concern relates to Path MTU, which might change if the size of an underlying header changes. Section 2.4 is intended to mitigate that issue. However, some ramifications, such as with Path MTU, may not be completely solvable in the general Internet, but require use cases to be confined to a network or set of consenting networks.

#### 6. Privacy Considerations

Data formats in this memo reveal no personally identifying information.

## 7. Acknowledgements

This note grew out of a discussion among the author, Ole Troan, Mark Townsley, Frank Brockners, and Shwetha Bhandari, and benefited from comments by Dennis Ferguson, Brian Carpenter, Panos Kampanakis, JINMEI Tatuya, and Joe Touch.

## 8. References

### 8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<http://www.rfc-editor.org/info/rfc2460>>.

### 8.2. Informative References

- [I-D.ietf-roll-trickle-mcast] Hui, J. and R. Kelsey, "Multicast Protocol for Low power and Lossy Networks (MPL)", draft-ietf-roll-trickle-mcast-12 (work in progress), June 2015.
- [I-D.ietf-v6ops-ipv6-ehs-in-real-world] Gont, F., Linkova, J., Chown, T., and S. LIU, "Observations on IPv6 EH Filtering in the Real World", draft-ietf-v6ops-ipv6-ehs-in-real-world-00 (work in progress), April 2015.
- [RCP] Dukkupati, N., "Rate Control Protocol (RCP): Congestion control to make flows complete quickly", Stanford University, 2006.
- [RFC2675] Borman, D., Deering, S., and R. Hinden, "IPv6 Jumbograms", RFC 2675, DOI 10.17487/RFC2675, August 1999, <<http://www.rfc-editor.org/info/rfc2675>>.
- [RFC2711] Partridge, C. and A. Jackson, "IPv6 Router Alert Option", RFC 2711, DOI 10.17487/RFC2711, October 1999, <<http://www.rfc-editor.org/info/rfc2711>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<http://www.rfc-editor.org/info/rfc4302>>.

- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<http://www.rfc-editor.org/info/rfc4303>>.
- [RFC4782] Floyd, S., Allman, M., Jain, A., and P. Sarolahti, "Quick-Start for TCP and IP", RFC 4782, DOI 10.17487/RFC4782, January 2007, <<http://www.rfc-editor.org/info/rfc4782>>.
- [RFC5570] StJohns, M., Atkinson, R., and G. Thomas, "Common Architecture Label IPv6 Security Option (CALIPSO)", RFC 5570, DOI 10.17487/RFC5570, July 2009, <<http://www.rfc-editor.org/info/rfc5570>>.
- [RFC6398] Le Faucheur, F., Ed., "IP Router Alert Considerations and Usage", BCP 168, RFC 6398, DOI 10.17487/RFC6398, October 2011, <<http://www.rfc-editor.org/info/rfc6398>>.
- [RFC6553] Hui, J. and JP. Vasseur, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams", RFC 6553, DOI 10.17487/RFC6553, March 2012, <<http://www.rfc-editor.org/info/rfc6553>>.
- [RFC6621] Macker, J., Ed., "Simplified Multicast Forwarding", RFC 6621, DOI 10.17487/RFC6621, May 2012, <<http://www.rfc-editor.org/info/rfc6621>>.
- [RFC6971] Herberg, U., Ed., Cardenas, A., Iwao, T., Dow, M., and S. Cespedes, "Depth-First Forwarding (DFF) in Unreliable Networks", RFC 6971, DOI 10.17487/RFC6971, June 2013, <<http://www.rfc-editor.org/info/rfc6971>>.
- [RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, DOI 10.17487/RFC7045, December 2013, <<http://www.rfc-editor.org/info/rfc7045>>.
- [XCP] Katabi, D., Handley, M., and C. Rohrs, "Congestion control for high bandwidth-delay product networks", SIGCOMM Symposium proceedings on Communications architectures and protocols , 2002.

#### Appendix A. Change Log

Initial Version: June 2015

01 Version: June 2015, responding to list discussion



Internet-Draft

October 2015

02 Version: July 2015, discussed at IETF 93

03 Version: October 2015

Author's Address

Fred Baker  
Cisco Systems  
Santa Barbara, California 93117  
USA

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

6man  
Internet-Draft  
Updates: 2460 (if approved)  
Intended status: Best Current Practice  
Expires: February 22, 2016

F. Gont  
UTN-FRH / SI6 Networks  
W. Liu  
Huawei Technologies  
R. Bonica  
Juniper Networks  
August 21, 2015

Transmission and Processing of IPv6 Options  
draft-gont-6man-ipv6-opt-transmit-02.txt

Abstract

Various IPv6 options have been standardized since the core IPv6 standard was first published. This document updates RFC 2460 to clarify how nodes should deal with such IPv6 options and with any options that are defined in the future. It complements [RFC7045], which offers a similar clarification regarding IPv6 Extension Headers.

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 February 22, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

## Table of Contents

1. Introduction and Problem Statement . . . . .	2
2. Terminology and Conventions Used in This Document . . . . .	3
2.1. Terminology . . . . .	3
2.2. Conventions . . . . .	3
3. Considerations for All IPv6 Options . . . . .	4
4. Processing of currently-defined IPv6 Options . . . . .	5
4.1. Hop-by-Hop Options Header . . . . .	5
4.2. Destination Options Header . . . . .	7
5. IANA Considerations . . . . .	7
6. Security Considerations . . . . .	9
7. Acknowledgements . . . . .	10
8. References . . . . .	10
8.1. Normative References . . . . .	10
8.2. Informative References . . . . .	13
Authors' Addresses . . . . .	14

## 1. Introduction and Problem Statement

Various IPv6 options have been standardized since the core IPv6 standard [RFC2460] was first published. Except for the padding options (Pad1 and PadN), all the options that have so far been specified are meant to be employed with specific IPv6 Extension Header (EH) types. Additionally, some options have specific requirements such as, for example, only allowing a single instance of the option in the corresponding IPv6 extension header. This establishes some criteria for validating packets that employ IPv6 options.

[RFC2460] specifies that IPv6 extension headers (with the exception of the Hop-by-Hop Options extension header) are not 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. However, in practice this is not really the case: some routers, and a variety of middleboxes such as firewalls, load balancers, or packet classifiers, might inspect other parts of each packet [RFC7045]. Hence both end-nodes and intermediate nodes may end up inspecting the contents of extension headers and discard packets based on the presence of specific IPv6 options.

This document clarifies the default processing of IPv6 options. In those cases in which the specifications add additional constraints/requirements regarding IPv6 options, such additional constraints/requirements are also taken into account.

## 2. Terminology and Conventions Used in This Document

### 2.1. Terminology

In the remainder of this document, the term "forwarding node" refers to any router, firewall, load balancer, prefix translator, or any other device or middlebox that forwards IPv6 packets with or without examining the packet in any way.

In this document, "standard" IPv6 options are those specified in detail by IETF Standards Actions [RFC5226]. "Experimental" options include those defined by any Experimental RFC and the option types 0x1E, 0x3E, 0x5E, 0x7E, 0x9E, 0xBE, 0xDE, and 0xFE, defined by [RFC3692] and [RFC4727] when used as experimental options. "Defined" options are the "standard" options plus the "experimental" ones.

The terms "permit" (allow the traffic), "drop" (drop with no notification to sender), and "reject" (drop with appropriate notification to sender) are employed as defined in [RFC3871]. Throughout this document we also employ the term "discard" as a generic term to indicate the act of discarding a packet, irrespective of whether the sender is notified of such packet drops.

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

### 2.2. Conventions

This document clarifies some basic validation of IPv6 options, and specifies the default processing of them. We recommend that a configuration option is made available to govern the processing of each IPv6 option type, on a per-EH-type granularity. Such configuration options may include the following possible settings:

- o Permit this IPv6 Option type
- o Drop (and log) packets containing this IPv6 option type
- o Reject (and log) packets containing this IPv6 option type (where the packet drop is signaled with an ICMPv6 error message)

- o Rate-limit the processing of packets containing this IPv6 option type
- o Ignore this IPv6 option type (forwarding packets that contain them)

We note that special care needs to be taken when devices log packet drops/rejects. Devices should count the number of packets dropped/rejected, but the logging of drop/reject events should be limited so as to not overburden device resources.

Finally, we note that when discarding packets, it is generally desirable that the sender be signaled of the packet drop, since this is of use for trouble-shooting purposes. However, throughout this document (when recommending that packets be discarded) we generically refer to the action as "discard" without specifying whether the sender is signaled of the packet drop.

### 3. Considerations for All IPv6 Options

Forwarding nodes that discard packets (by default) based on the presence of IPv6 options are known to cause connectivity failures and deployment problems. Any forwarding node along an IPv6 packet's path, which forwards the packet for any reason, SHOULD do so regardless of any IPv6 Destination Options that are present, as required by [RFC2460]. Exceptionally, if a forwarding node is designed to examine IPv6 Destination Options for any reason, such as firewalling, it MUST recognise and deal appropriately with all standard IPv6 options types and SHOULD recognise and deal appropriately with all experimental IPv6 options. The list of standard and experimental option types is maintained by IANA (see [IANA-IPV6-PARAM]), and implementors are advised to check this list regularly for updates.

In the case of some options meant to be included in IPv6 extension headers other than Hop-by-Hop Options, [RFC2460] requires destination hosts to discard the corresponding packet if the option is unrecognised. However, intermediate forwarding nodes SHOULD NOT do this, since doing so might cause them to inadvertently discard traffic using a recently standardised IPv6 option not yet recognised by the intermediate node. The exceptions to this rule are discussed next.

If a forwarding node discards a packet containing a standard IPv6 option, it MUST be the result of a configurable policy and not just the result of a failure to recognise such an option. This means that the discard policy for each standard type of IPv6 option MUST be

individually configurable. The default configuration SHOULD allow all standard IPv6 options.

Experimental IPv6 options SHOULD be treated in the same way as standard IPv6 options, including an individually configurable discard policy.

A node that processes the contents of an extension header MUST discard the corresponding packet if it contains any defined options that are not meant for the extension header being processed. This document requests IANA to add a new column to [IANA-IPV6-PARAM] to clearly mark the IPv6 Extension Header type(s) for which each option (defined by IETF Standards Action or IESG Approval) is valid.

A node that processes the contents of an IPv6 extension header MAY discard the corresponding packet if it contains any options that have become deprecated. Whether or not such packets are dropped SHOULD be configurable, and the default setting MUST be to not drop such packets.

A node that processes the contents of an extension header and encounters an undefined (unrecognised) IPv6 option MUST react to such option according to the highest-order two bits of the option type, as specified by Section 4.2 of [RFC2460].

A node that processes an IPv6 extension header MAY discard a packet containing any experimental IPv6 options.

#### 4. Processing of currently-defined IPv6 Options

The following subsections provide advice on how to process the IPv6 options that have been defined at the time of this writing, according to the rules specified in the previous sections.

##### 4.1. Hop-by-Hop Options Header

A node that processes the Hop-by-Hop Options extension header MUST discard the corresponding packet if it contains any options that are not valid for the Hop-by-Hop Options extension header [IANA-IPV6-PARAM].

A node that processes the Hop-by-Hop Options extension header MUST discard a packet containing multiple instances (i.e., more than one) of this option in the Hop-by-Hop Options extension header:

- o Type 0x05: Router Alert [RFC2711]

NOTE: The rationale for discarding the packet is that [RFC2711] forbids multiple instances of this option.

A node that processes the Hop-by-Hop Options extension header MUST discard a packet that carries a Fragment Header and also contains this option in the Hop-by-Hop Options extension header:

- o Type 0xC2: Jumbo Payload [RFC2675]

NOTE: The rationale for discarding the packet is that [RFC2675] forbids the use of the Jumbo Payload Option in packets that carry a Fragment Header.

A node that processes the Hop-by-Hop Options extension header MAY discard a packet containing any of the following options in that header:

- o Type=0x4D: Deprecated

NOTE: The rationale for discarding the packet is that the aforementioned option has been deprecated.

A node that processes the Hop-by-Hop Options extension header MAY discard a packet containing any of the following options in that header:

- o Type 0x1E: RFC3692-style Experiment [RFC4727]
- o Type 0x3E: RFC3692-style Experiment [RFC4727]
- o Type 0x5E: RFC3692-style Experiment [RFC4727]
- o Type 0x7E: RFC3692-style Experiment [RFC4727]
- o Type 0x9E: RFC3692-style Experiment [RFC4727]
- o Type 0xBE: RFC3692-style Experiment [RFC4727]
- o Type 0xDE: RFC3692-style Experiment [RFC4727]
- o Type 0xFE: RFC3692-style Experiment [RFC4727]

NOTE: This is in line with the corresponding specification in [RFC7045] for experimental extension headers.

#### 4.2. Destination Options Header

A node that processes the Destination Options header MUST discard a packet containing any options that are not valid for the Destination Options header [IANA-IPV6-PARAM].

A node that processes the Destination Options extension header MAY discard a packet containing any of the following options in that header:

- o Type 0x8A: Endpoint Identification [nimrod-eid] [NIMROD-DOC]
- o Type 0x4D: Deprecated

NOTE: The rationale for discarding the packet is that the aforementioned options have been deprecated.

A node that processes the Destination Options extension header MAY discard a packet containing any of the following options in that header:

- o Type 0x1E: RFC3692-style Experiment [RFC4727]
- o Type 0x3E: RFC3692-style Experiment [RFC4727]
- o Type 0x5E: RFC3692-style Experiment [RFC4727]
- o Type 0x7E: RFC3692-style Experiment [RFC4727]
- o Type 0x9E: RFC3692-style Experiment [RFC4727]
- o Type 0xBE: RFC3692-style Experiment [RFC4727]
- o Type 0xDE: RFC3692-style Experiment [RFC4727]
- o Type 0xFE: RFC3692-style Experiment [RFC4727]

NOTE: This is in line with the corresponding specification in [RFC7045] for experimental extension headers.

#### 5. IANA Considerations

IANA is requested to add an extra column entitled "Extension Header Types" to the "Destination Options and Hop-by-Hop Options" registry [IANA-IPV6-PARAM], to clearly mark the IPv6 Extension Header types for which each option (defined by IETF Standards Action or IESG Approval) is valid (see the list below). This also applies to Destination Options and Hop-by-Hop Options defined in the future.



What follows is the initial list of IPv6 options and the corresponding marks that indicate which Extension Header type(s) these IPv6 options are valid for:

Hex Value	Description	Reference	EH Types
0x00	Pad1	[RFC2460]	DH
0x01	PadN	[RFC2460]	DH
0xC2	Jumbo Payload	[RFC2675]	H
0x63	RPL Option	[RFC6553]	H
0x04	Tunnel Encapsulation Limit	[RFC2473]	D
0x05	Router Alert	[RFC2711]	H
0x26	Quick-Start	[RFC4782]	H
0x07	CALIPSO	[RFC5570]	H
0x08	SMF_DPD	[RFC6621]	H
0xC9	Home Address	[RFC6275]	D
0x8A	Endpoint Identification	[nimrod-eid][NIMROD-DOC]	D
0x8B	ILNP Nonce	[RFC6744]	D
0x8C	Line-Identification Option	[RFC6788]	D
0x4D	Deprecated		U
0x6D	MPL Option	[I-D.ietf-roll-trickle-mcast]	H
0xEE	IPv6 DFF Header	[RFC6971]	H
0x1E	RFC3692-style Experiment	[RFC4727]	DH

0x3E	RFC3692-style Experiment	[RFC4727]	DH
0x5E	RFC3692-style Experiment	[RFC4727]	DH
0x7E	RFC3692-style Experiment	[RFC4727]	DH
0x9E	RFC3692-style Experiment	[RFC4727]	DH
0xBE	RFC3692-style Experiment	[RFC4727]	DH
0xDE	RFC3692-style Experiment	[RFC4727]	DH
0xFE	RFC3692-style Experiment	[RFC4727]	DH

Additionally, the following legend should be added to the registry:

D: Destination Options Header

H: Hop-by-Hop Options Header

U: Unknown

## 6. Security Considerations

Forwarding nodes that operate as firewalls MUST conform to the requirements in this document. In particular, packets containing standard IPv6 options are only to be discarded as a result of an intentionally configured policy.

These requirements do not affect a firewall's ability to filter out traffic containing unwanted or suspect IPv6 options, if configured to do so. However, the changes do require firewalls to be capable of permitting any or all IPv6 options, if configured to do so. The default configurations are intended to allow normal use of any standard IPv6 option, avoiding the interoperability issues described in Section 1 and Section 3.

As noted above, the default configuration might discard packets containing experimental IPv6 options.

## 7. Acknowledgements

This document is heavily based on [RFC7045], authored by Brian Carpenter and Sheng Jiang.

The authors of this document would like to thank (in alphabetical order) Brian Carpenter, Mike Heard, and Jen Linkova, for providing valuable comments on earlier versions of this document.

## 8. References

### 8.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<http://www.rfc-editor.org/info/rfc1034>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2205] Braden, R., Ed., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, DOI 10.17487/RFC2205, September 1997, <<http://www.rfc-editor.org/info/rfc2205>>.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<http://www.rfc-editor.org/info/rfc2460>>.
- [RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <<http://www.rfc-editor.org/info/rfc2473>>.
- [RFC2675] Borman, D., Deering, S., and R. Hinden, "IPv6 Jumbograms", RFC 2675, DOI 10.17487/RFC2675, August 1999, <<http://www.rfc-editor.org/info/rfc2675>>.
- [RFC2710] Deering, S., Fenner, W., and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6", RFC 2710, DOI 10.17487/RFC2710, October 1999, <<http://www.rfc-editor.org/info/rfc2710>>.
- [RFC2711] Partridge, C. and A. Jackson, "IPv6 Router Alert Option", RFC 2711, DOI 10.17487/RFC2711, October 1999, <<http://www.rfc-editor.org/info/rfc2711>>.

- [RFC3692] Narten, T., "Assigning Experimental and Testing Numbers Considered Useful", BCP 82, RFC 3692, DOI 10.17487/RFC3692, January 2004, <<http://www.rfc-editor.org/info/rfc3692>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<http://www.rfc-editor.org/info/rfc4302>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<http://www.rfc-editor.org/info/rfc4303>>.
- [RFC4304] Kent, S., "Extended Sequence Number (ESN) Addendum to IPsec Domain of Interpretation (DOI) for Internet Security Association and Key Management Protocol (ISAKMP)", RFC 4304, DOI 10.17487/RFC4304, December 2005, <<http://www.rfc-editor.org/info/rfc4304>>.
- [RFC4727] Fenner, B., "Experimental Values In IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers", RFC 4727, DOI 10.17487/RFC4727, November 2006, <<http://www.rfc-editor.org/info/rfc4727>>.
- [RFC4782] Floyd, S., Allman, M., Jain, A., and P. Sarolahti, "Quick-Start for TCP and IP", RFC 4782, DOI 10.17487/RFC4782, January 2007, <<http://www.rfc-editor.org/info/rfc4782>>.
- [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, <<http://www.rfc-editor.org/info/rfc5095>>.
- [RFC5201] Moskowitz, R., Nikander, P., Jokela, P., Ed., and T. Henderson, "Host Identity Protocol", RFC 5201, DOI 10.17487/RFC5201, April 2008, <<http://www.rfc-editor.org/info/rfc5201>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <<http://www.rfc-editor.org/info/rfc5226>>.
- [RFC5533] Nordmark, E. and M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6", RFC 5533, DOI 10.17487/RFC5533, June 2009, <<http://www.rfc-editor.org/info/rfc5533>>.

- [RFC5570] StJohns, M., Atkinson, R., and G. Thomas, "Common Architecture Label IPv6 Security Option (CALIPSO)", RFC 5570, DOI 10.17487/RFC5570, July 2009, <<http://www.rfc-editor.org/info/rfc5570>>.
- [RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, DOI 10.17487/RFC6275, July 2011, <<http://www.rfc-editor.org/info/rfc6275>>.
- [RFC6398] Le Faucheur, F., Ed., "IP Router Alert Considerations and Usage", BCP 168, RFC 6398, DOI 10.17487/RFC6398, October 2011, <<http://www.rfc-editor.org/info/rfc6398>>.
- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<http://www.rfc-editor.org/info/rfc6550>>.
- [RFC6553] Hui, J. and JP. Vasseur, "The Routing Protocol for Low-Power and Lossy Networks (RPL) Option for Carrying RPL Information in Data-Plane Datagrams", RFC 6553, DOI 10.17487/RFC6553, March 2012, <<http://www.rfc-editor.org/info/rfc6553>>.
- [RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6554, DOI 10.17487/RFC6554, March 2012, <<http://www.rfc-editor.org/info/rfc6554>>.
- [RFC6621] Macker, J., Ed., "Simplified Multicast Forwarding", RFC 6621, DOI 10.17487/RFC6621, May 2012, <<http://www.rfc-editor.org/info/rfc6621>>.
- [RFC6740] Atkinson, RJ. and SN. Bhatti, "Identifier-Locator Network Protocol (ILNP) Architectural Description", RFC 6740, DOI 10.17487/RFC6740, November 2012, <<http://www.rfc-editor.org/info/rfc6740>>.
- [RFC6744] Atkinson, RJ. and SN. Bhatti, "IPv6 Nonce Destination Option for the Identifier-Locator Network Protocol for IPv6 (ILNPv6)", RFC 6744, DOI 10.17487/RFC6744, November 2012, <<http://www.rfc-editor.org/info/rfc6744>>.

- [RFC6788] Krishnan, S., Kavanagh, A., Varga, B., Ooghe, S., and E. Nordmark, "The Line-Identification Option", RFC 6788, DOI 10.17487/RFC6788, November 2012, <<http://www.rfc-editor.org/info/rfc6788>>.
- [RFC6971] Herberg, U., Ed., Cardenas, A., Iwao, T., Dow, M., and S. Cespedes, "Depth-First Forwarding (DFF) in Unreliable Networks", RFC 6971, DOI 10.17487/RFC6971, June 2013, <<http://www.rfc-editor.org/info/rfc6971>>.
- [RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, DOI 10.17487/RFC7045, December 2013, <<http://www.rfc-editor.org/info/rfc7045>>.
- [RFC7112] Gont, F., Manral, V., and R. Bonica, "Implications of Oversized IPv6 Header Chains", RFC 7112, DOI 10.17487/RFC7112, January 2014, <<http://www.rfc-editor.org/info/rfc7112>>.

## 8.2. Informative References

- [Biondi2007]  
Biondi, P. and A. Ebalard, "IPv6 Routing Header Security", CanSecWest 2007 Security Conference, 2007, <[http://www.secdev.org/conf/IPv6\\_RH\\_security-csw07.pdf](http://www.secdev.org/conf/IPv6_RH_security-csw07.pdf)>.
- [I-D.ietf-roll-trickle-mcast]  
Hui, J. and R. Kelsey, "Multicast Protocol for Low power and Lossy Networks (MPL)", draft-ietf-roll-trickle-mcast-12 (work in progress), June 2015.
- [I-D.ietf-v6ops-ipv6-ehs-in-real-world]  
Gont, F., Linkova, J., Chown, T., and S. LIU, "Observations on IPv6 EH Filtering in the Real World", draft-ietf-v6ops-ipv6-ehs-in-real-world-00 (work in progress), April 2015.
- [IANA-IPV6-PARAM]  
Internet Assigned Numbers Authority, "Internet Protocol Version 6 (IPv6) Parameters", December 2013, <<http://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml>>.
- [NIMROD-DOC]  
Nimrod Documentation Page, , <<http://ana-3.lcs.mit.edu/~jnc/nimrod/>>.

## [nimrod-eid]

Lynn, C., "Endpoint Identifier Destination Option", IETF Internet Draft, draft-ietf-nimrod-eid-00.txt, November 1995.

[RFC3871] Jones, G., Ed., "Operational Security Requirements for Large Internet Service Provider (ISP) IP Network Infrastructure", RFC 3871, DOI 10.17487/RFC3871, September 2004, <<http://www.rfc-editor.org/info/rfc3871>>.

[RFC7126] Gont, F., Atkinson, R., and C. Pignataro, "Recommendations on Filtering of IPv4 Packets Containing IPv4 Options", BCP 186, RFC 7126, DOI 10.17487/RFC7126, February 2014, <<http://www.rfc-editor.org/info/rfc7126>>.

## Authors' Addresses

Fernando Gont  
UTN-FRH / 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: <http://www.si6networks.com>

Will(Shucheng) Liu  
Huawei Technologies  
Bantian, Longgang District  
Shenzhen 518129  
P.R. China

Email: [liushucheng@huawei.com](mailto:liushucheng@huawei.com)

Ronald P. Bonica  
Juniper Networks  
2251 Corporate Park Drive  
Herndon, VA 20171  
US

Phone: 571 250 5819  
Email: [rbonica@juniper.net](mailto:rbonica@juniper.net)

IPv6 maintenance Working Group (6man)  
Internet-Draft  
Obsoletes: 6564 (if approved)  
Intended status: Standards Track  
Expires: March 19, 2016

F. Gont  
SI6 Networks / UTN-FRH  
W. Liu  
Huawei Technologies  
S. Krishnan  
Ericsson  
H. Pfeifer  
Rohde & Schwarz  
September 16, 2015

IPv6 Universal Extension Header  
draft-gont-6man-rfc6564bis-01

Abstract

In IPv6, optional internet-layer information is encoded in separate headers that may be placed between the IPv6 header and the transport-layer header. There are a small number of such extension headers currently defined. This document describes the issues that can arise when defining new extension headers and specifies a new IPv6 Extension Header - the Universal Extension Header - that overcomes the aforementioned problem, while enabling the extensibility of IPv6. Finally, this document formally obsoletes RFC 6564.

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 March 19, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.



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

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. A Problem with RFC 6564 . . . . .	3
4. Implications . . . . .	3
5. UEH Specification . . . . .	4
6. Forbidding New IPv6 Extension Headers . . . . .	5
7. Operation of the UEH . . . . .	5
8. IANA Considerations . . . . .	5
9. Security Considerations . . . . .	6
10. Acknowledgements . . . . .	6
11. Contributors . . . . .	6
12. References . . . . .	6
12.1. Normative References . . . . .	6
12.2. Informative References . . . . .	6
Authors' Addresses . . . . .	7

## 1. Introduction

There has recently been a lot of work in the area of IPv6 Extension Headers. Firstly, there has been research about the extent to which IPv6 packets employing Extension Headers are dropped in the public Internet [GONT-IEPG-Nov13] [GONT-IEPG-Mar14], and debate about the motivation behind such policy [I-D.gont-v6ops-ipv6-ehs-packet-drops]. Secondly, there has been a fair share of work to improve some technicalities of IPv6 Extension Headers (see e.g. [RFC7112] [RFC7045]) in the hopes that they can be reliably used in the public Internet.

A key challenge for IPv6 Extension Headers to be "deployable" in the public Internet is that they should not impair any nodes's ability to process the entire IPv6 header chain. One of the steps meant in that direction has been the specification of a Uniform Format for IPv6 Extension Headers [RFC6564], which was meant to be employed by any IPv6 Extension Headers that might be defined in the future, such that middle-boxes can still process the entire IPv6 header chain if new extension headers were specified. However, a problem in the

aforementioned specification prevents such uniform format from being of use.

Section 3 discusses the aforementioned flaw in the Uniform Format for Extension Headers specified in [RFC6564]. Section 4 explicitly describes the implications of the aforementioned flaw. Section 5 specifies the new Universal Extension Header (UEH). Section 7 explains how new IPv6 extensions would be specified with the UEH. Section 6 formally forbids the specification of new IPv6 Extension Headers (with new Next Header values), and mandates that any new IPv6 extensions be conveyed/encoded in the UEH specified in this document.

## 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].

## 3. A Problem with RFC 6564

A key problem with the Uniform Format for IPv6 Extension Headers [RFC6564] lies in that both IPv6 Extension Headers and Transport Protocols share the same "Next Header" registry/namespace. Thus, given an "unknown Next Header value", it is impossible to tell whether the aforementioned value refers to an IPv6 Extension Header that employs the aforementioned uniform format, or an "unknown" upper-layer protocol (e.g. an "unknown" transport protocol). That is, while [RFC6564] specifies the syntax for a Uniform Format for IPv6 Extension Headers, it does not provide a mechanism for a node to identify whether the aforementioned format is being employed in the first place.

## 4. Implications

The current impossibility to parse an IPv6 header chain that includes unknown Next Header values results in concrete implications for the extensibility of the IPv6 protocol, and the deployability of new transport protocols. Namely,

- o New IPv6 extension headers cannot be incrementally deployed.
- o New transport protocols cannot be incrementally deployed.

Since there is no way for a node to process IPv6 extension headers that employ unknown next header values, an IPv6 host that receives a packet that employs a new IPv6 extension header will not be able to parse the IPv6 header chain past that unknown extension header, and hence it will drop the aforementioned packet

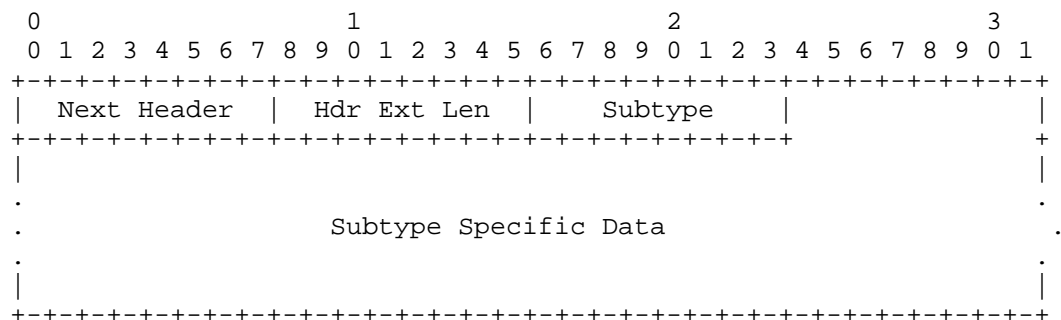
[I-D.gont-v6ops-ipv6-ehs-packet-drops]. In a similar way, a middlebox that needs to process the transport-protocol header will be faced with the dilemma of what to do with packets that employ unknown Next Header values. Since they will not be able to parse the IPv6 header chain past the unknown Next Header, it is very likely that they will drop such packets.

Unfortunately, since transport protocols share the same namespace as IPv6 Extension Headers, new transport protocols will pose the same challenge to middle-boxes, and hence they will be likely dropped in the network.

We believe that the current situation has implications that are generally overlooked, and that, whatever the outcome, it should be the result of an explicit decision by our community, rather than simply "omission".

## 5. UEH Specification

This document specifies a new IPv6 Extension Header: Universal Extension Header. This Extension Header is identified by the value [TBD] of [IANA-IP-PROTO]. The syntax of the Universal Extension Header is:



where:

Next Header

8-bit selector. Identifies the type of header immediately following the extension header. Uses the same values as the IPv4 Protocol field [IANA-IP-PROTO].

Hdr Ext Len

8-bit unsigned integer. Length of the extension header in 8-octet units, not including the first 8 octets.

#### Subtype

8-bit unsigned integer. Specifies the subtype for this extension header. It uses a new namespace managed by IANA [IANA-UEH].

#### Subtype Specific Data

Variable length. Fields specific to this extension header/Subtype.

The Universal Extension Header specified in this document MAY appear multiple times in the same IPv6 packet.

### 6. Forbidding New IPv6 Extension Headers

Since the specification of any new IPv6 Extension Headers (i.e., with new Next Header values) would hamper (among other things) the incremental deployment of extensions and new transport protocols, and basic operational practices such as the enforcement of simple ACLs, new IPv6 Extension Headers MUST NOT be specified in any future specifications. Any IPv6 extensions that would require a new IPv6 Extension Header MUST be implemented with the Universal Extension Header specified in this document. This minimizes breakage in intermediate nodes that need to parse the entire IPv6 header chain.

### 7. Operation of the UEH

This section describes the operation of the Universal Extension Header.

The goal of the UEH is to provide a common syntax for all future IPv6 extensions. Any future extension headers will be encoded in a UEH, and will be identified by a specific UEH Subtype assigned by IANA at the time the corresponding specification is published. The UEH thus provides the "common syntax" required to process "unrecognized extensions", and the Subtype field identifies the specific extension being encoded in the UEH. Any "future extension headers" would actually be new Subtypes (assigned by IANA) of the UEH.

As a result, unrecognized Next Header values should be interpreted to identify an upper-layer protocol, rather than an IPv6 extension header.

### 8. IANA Considerations

IANA is requested to create a new registry to maintain the Universal Extension Header Subtypes [IANA-UEH].

## 9. Security Considerations

Enabling nodes to parse an entire IPv6 header chain even in the presence of unrecognized extensions allows for security mechanisms to be implemented and deployed.

## 10. Acknowledgements

The authors would like to thank [TBD] for providing valuable input on earlier versions of this document.

## 11. Contributors

C.M. Heard identified the problems related with the Uniform Format for IPv6 Extension Headers specified in [RFC6564], and participated in the brainstorming that led to this document.

## 12. References

### 12.1. Normative References

- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<http://www.rfc-editor.org/info/rfc2460>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC6564] Krishnan, S., Woodyatt, J., Kline, E., Hoagland, J., and M. Bhatia, "A Uniform Format for IPv6 Extension Headers", RFC 6564, DOI 10.17487/RFC6564, April 2012, <<http://www.rfc-editor.org/info/rfc6564>>.
- [RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, DOI 10.17487/RFC7045, December 2013, <<http://www.rfc-editor.org/info/rfc7045>>.

### 12.2. Informative References

- [RFC7112] Gont, F., Manral, V., and R. Bonica, "Implications of Oversized IPv6 Header Chains", RFC 7112, DOI 10.17487/RFC7112, January 2014, <<http://www.rfc-editor.org/info/rfc7112>>.

[I-D.gont-v6ops-ipv6-ehs-packet-drops]

Gont, F., Hilliard, N., Doering, G., LIU, S., and W. Kumari, "Operational Implications of IPv6 Packets with Extension Headers", draft-gont-v6ops-ipv6-ehs-packet-drops-00 (work in progress), July 2015.

[GONT-IEPG-Nov13]

Gont, F., "Fragmentation and Extension Header Support in the IPv6 Internet", IEPG 88, November 3, 2013. Vancouver, BC, Canada, 2013, <<http://www.iepg.org/2013-11-ietf88/fgont-iepg-ietf88-ipv6-frag-and-eh.pdf>>.

[GONT-IEPG-Mar14]

Gont, F. and T. Chown, "More results from measurements of IPv6 Extension Header probing", IEPG 89, March 2, 2014. London, U.K., 2014, <<http://www.iepg.org/2014-03-02-ietf89/fgont-iepg-ietf89-eh-update.pdf>>.

[IANA-IP-PROTO]

Internet Assigned Numbers Authority, "Assigned Internet Protocol Numbers", April 2011, <<http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>>.

[IANA-UEH]

Internet Assigned Numbers Authority, "Universal Extension Header Subtypes", 2014.

Authors' Addresses

Fernando Gont  
SI6 Networks / UTN-FRH  
Evaristo Carriego 2644  
Haedo, Provincia de Buenos Aires 1706  
Argentina

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

Will (Shucheng) Liu  
Huawei Technologies  
Bantian, Longgang District  
Shenzhen 518129  
P.R. China

Email: [liushucheng@huawei.com](mailto:liushucheng@huawei.com)

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

Phone: +1 514 345 7900 x42871  
Email: suresh.krishnan@ericsson.com

Hagen Paul Pfeifer  
Rohde & Schwarz  
Muehldorfstrasse 15  
Munich 81671  
Germany

Phone: +49 89 4129 15515  
Email: hagen.pfeifer@rohde-schwarz.com  
URI: <http://www.rohde-schwarz.com/>

Network Working Group  
Internet-Draft  
Obsoletes: 4291 (if approved)  
Intended status: Standards Track  
Expires: April 21, 2016

R. Hinden  
Check Point Software  
S. Deering  
Retired  
October 19, 2015

IP Version 6 Addressing Architecture  
draft-hinden-6man-rfc4291bis-06

Abstract

This specification defines the addressing architecture of the IP Version 6 (IPv6) protocol. The document includes the IPv6 addressing model, text representations of IPv6 addresses, definition of IPv6 unicast addresses, anycast addresses, and multicast addresses, and an IPv6 node's required addresses.

This document obsoletes RFC 4291, "IP Version 6 Addressing Architecture".

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 April 21, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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



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

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

## Table of Contents

1. Introduction . . . . .	3
2. IPv6 Addressing . . . . .	3
2.1. Addressing Model . . . . .	4
2.2. Text Representation of IPv6 Addresses . . . . .	4
2.2.1. Text Representation of Addresses . . . . .	4
2.2.2. Text Representation of Address Prefixes . . . . .	5
2.2.3. Recommendation for outputting IPv6 addresses . . . . .	7
2.3. Address Type Identification . . . . .	9
2.4. Unicast Addresses . . . . .	10
2.4.1. Interface Identifiers . . . . .	11
2.4.2. The Unspecified Address . . . . .	12
2.4.3. The Loopback Address . . . . .	13
2.4.4. Global Unicast Addresses . . . . .	13
2.4.5. IPv6 Addresses with Embedded IPv4 Addresses . . . . .	13
2.4.5.1. IPv4-Compatible IPv6 Address . . . . .	14
2.4.5.2. IPv4-Mapped IPv6 Address . . . . .	14
2.4.6. Link-Local IPv6 Unicast Addresses . . . . .	14
2.4.7. Site-Local IPv6 Unicast Addresses . . . . .	15
2.5. Anycast Addresses . . . . .	15
2.5.1. Required Anycast Address . . . . .	16
2.6. Multicast Addresses . . . . .	16
2.6.1. Pre-Defined Multicast Addresses . . . . .	19
2.7. A Node's Required Addresses . . . . .	21
3. IANA Considerations . . . . .	22
4. Security Considerations . . . . .	22
5. Acknowledgments . . . . .	22
6. References . . . . .	23
6.1. Normative References . . . . .	23
6.2. Informative References . . . . .	23

Appendix A. Creating Modified EUI-64 Format Interface Identifiers . . . . .	24
Appendix B. CHANGES SINCE RFC 4291 . . . . .	27
Authors' Addresses . . . . .	29

## 1. Introduction

This specification defines the addressing architecture of the IP Version 6 protocol. It includes the basic formats for the various types of IPv6 addresses (unicast, anycast, and multicast).

## 2. IPv6 Addressing

IPv6 addresses are 128-bit identifiers for interfaces and sets of interfaces (where "interface" is as defined in Section 2 of [I-D.hinden-6man-rfc2460bis]). There are three types of addresses:

- Unicast: An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address.
- Anycast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to an anycast address is delivered to one of the interfaces identified by that address (the "nearest" one, according to the routing protocols' measure of distance).
- Multicast: An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address.

There are no broadcast addresses in IPv6, their function being superseded by multicast addresses.

In this document, fields in addresses are given a specific name, for example, "subnet". When this name is used with the term "ID" for identifier after the name (e.g., "subnet ID"), it refers to the contents of the named field. When it is used with the term "prefix" (e.g., "subnet prefix"), it refers to all of the address from the left up to and including this field.

In IPv6, all zeros and all ones are legal values for any field, unless specifically excluded. Specifically, prefixes may contain, or end with, zero-valued fields.

## 2.1. Addressing Model

IPv6 addresses of all types are assigned to interfaces, not nodes. An IPv6 unicast address refers to a single interface. Since each interface belongs to a single node, any of that node's interfaces' unicast addresses may be used as an identifier for the node.

All interfaces are required to have at least one Link-Local unicast address (see Section 2.8 for additional required addresses). A single interface may also have multiple IPv6 addresses of any type (unicast, anycast, and multicast) or scope. Unicast addresses with a scope greater than link-scope are not needed for interfaces that are not used as the origin or destination of any IPv6 packets to or from non-neighbors. This is sometimes convenient for point-to-point interfaces. There is one exception to this addressing model:

A unicast address or a set of unicast addresses may be assigned to multiple physical interfaces if the implementation treats the multiple physical interfaces as one interface when presenting it to the internet layer. This is useful for load-sharing over multiple physical interfaces.

Currently, IPv6 continues the IPv4 model in that a subnet prefix is associated with one link. Multiple subnet prefixes may be assigned to the same link.

## 2.2. Text Representation of IPv6 Addresses

### 2.2.1. Text Representation of Addresses

There are three conventional forms for representing IPv6 addresses as text strings:

1. The preferred form is `x:x:x:x:x:x:x:x`, where the 'x's are one to four hexadecimal digits of the eight 16-bit pieces of the address. Examples:

```
abcd:ef01:2345:6789:abcd:ef01:2345:6789
2001:db8:0:0:8:800:200c:417a
```

Note that it is not necessary to write the leading zeros in an individual field, but there must be at least one numeral in every field (except for the case described in 2.).

2. Due to some methods of allocating certain styles of IPv6 addresses, it will be common for addresses to contain long strings of zero bits. In order to make writing addresses containing zero

bits easier, a special syntax is available to compress the zeros. The use of "::" indicates one or more groups of 16 bits of zeros. The "::" can only appear once in an address. The "::" can also be used to compress leading or trailing zeros in an address.

For example, the following addresses

2001:db8:0:0:8:800:200c:417a	a unicast address
ff01:0:0:0:0:0:0:101	a multicast address
0:0:0:0:0:0:0:1	the loopback address
0:0:0:0:0:0:0:0	the unspecified address

may be represented as

2001:db8::8:800:200c:417a	a unicast address
ff01::101	a multicast address
::1	the loopback address
::	the unspecified address

3. An alternative form that is sometimes more convenient when dealing with a mixed environment of IPv4 and IPv6 nodes is x:x:x:x:x:x:d.d.d.d, where the 'x's are the hexadecimal values of the six high-order 16-bit pieces of the address, and the 'd's are the decimal values of the four low-order 8-bit pieces of the address (standard IPv4 representation). Examples:

```
0:0:0:0:0:0:13.1.68.3
0:0:0:0:0:ffff:129.144.52.38
```

or in compressed form:

```
::13.1.68.3
::ffff:129.144.52.38
```

### 2.2.2. Text Representation of Address Prefixes

The text representation of IPv6 address prefixes is similar to the way IPv4 address prefixes are written in Classless Inter-Domain

Routing (CIDR) notation [RFC4632]. An IPv6 address prefix is represented by the notation:

`ipv6-address/prefix-length`

where

`ipv6-address` is an IPv6 address in any of the notations listed in Section 2.2.

`prefix-length` is a decimal value specifying how many of the leftmost contiguous bits of the address comprise the prefix.

For example, the following are legal representations of the 60-bit prefix 20010db80000cd3 (hexadecimal):

`2001:0db8:0000:cd30:0000:0000:0000:0000/60`

`2001:0db8::cd30:0:0:0:0/60`

`2001:0db8:0:cd30::/60`

The following are NOT legal representations of the above prefix:

`2001:0db8:0:cd3/60` may drop leading zeros, but not trailing zeros, within any 16-bit chunk of the address

`2001:0db8::cd30/60` address to left of "/" expands to  
`2001:0db8:0000:0000:0000:0000:0000:cd30`

`2001:0db8::cd3/60` address to left of "/" expands to  
`2001:0db8:0000:0000:0000:0000:0000:0cd3`

When writing both a node address and a prefix of that node address (e.g., the node's subnet prefix), the two can be combined as follows:

the node address            `2001:0db8:0:cd30:123:4567:89ab:cdef`  
and its subnet number      `2001:0db8:0:cd30::/60`

can be abbreviated as      `2001:0db8:0:cd30:123:4567:89ab:cdef/60`

### 2.2.3. Recommendation for outputting IPv6 addresses

This section provides a recommendation for systems generating and outputting IPv6 addresses as text. Note, all implementations must accept and process all addresses in the formats defined in the previous two sections of this document. The recommendations are as follows:

1. The hexadecimal digits "a", "b", "c", "d", "e", and "f" in an IPv6 address must be represented in lowercase.
2. Leading zeros in a 16-Bit Field must be suppressed. For example,

2001:0db8::0001

is not correct and must be represented as

2001:db8::1

3. A single 16-bit 0000 field must be represented as 0.

The use of the symbol ":::" must be used to its maximum capability. For example:

2001:db8:0:0:0:0:2:1

must be shortened to

2001:db8::2:1

Likewise,

2001:db8::0:1

is not correct, because the symbol ":::" could have been used to produce a shorter representation

2001:db8::1.

4. When there is an alternative choice in the placement of a "::", the longest run of consecutive 16-bit 0 fields must be shortened, that is, in

2001:0:0:1:0:0:0:1

the sequence with three consecutive zero fields is shortened to

2001:0:0:1::1

5. When the length of the consecutive 16-bit 0 fields are equal, for example

2001:db8:0:0:1:0:0:1

the first sequence of zero bits must be shortened. For example

2001:db8::1:0:0:1

is the correct representation.

6. The symbol "::" must not be used to shorten just one 16-bit 0 field. For example, the representation

2001:db8:0:1:1:1:1:1

is correct, but

2001:db8::1:1:1:1:1

is not correct.

7. The text representation method describe in this section should also be use for text Representation of IPv6 Address Prefixes. For example

0:0:0:0:0:ffff:192.0.2.1

should be shown as

::ffff:192.0.2.1

8. The text representation method describe in this section should be applied for IPv6 addresses with embedded IPv4 address. For example

2001:0db8:0000:cd30:0000:0000:0000:0000/60

should be shown as

2001:0db8:0:cd30::/60

### 2.3. Address Type Identification

The type of an IPv6 address is identified by the high-order bits of the address, as follows:

Address type	Binary prefix	IPv6 notation	Section
-----	-----	-----	-----
Unspecified	00...0 (128 bits)	::/128	2.5.2
Loopback	00...1 (128 bits)	::1/128	2.5.3
Multicast	11111111	ff00::/8	2.7
Link-Local unicast	1111111010	fe80::/10	2.5.6
Global Unicast	(everything else)		

Anycast addresses are taken from the unicast address spaces (of any scope) and are not syntactically distinguishable from unicast addresses.

The general format of Global Unicast addresses is described in Section 2.5.4. Some special-purpose subtypes of Global Unicast



addresses that contain embedded IPv4 addresses (for the purposes of IPv4-IPv6 interoperation) are described in Section 2.5.5.

Future specifications may redefine one or more sub-ranges of the Global Unicast space for other purposes, but unless and until that happens, implementations must treat all addresses that do not start with any of the above-listed prefixes as Global Unicast addresses.

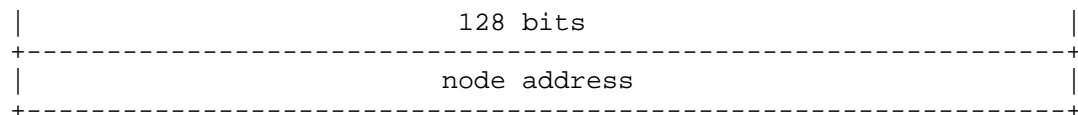
The current assigned IPv6 prefixes and references to their usage can be found in the IANA Internet Protocol Version 6 Address Space registry [IANA-AD] and the IANA IPv6 Special-Purpose Address Registry [IANA-SP].

## 2.4. Unicast Addresses

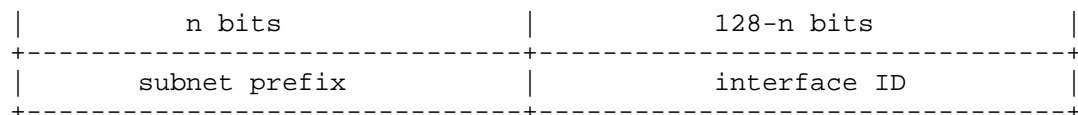
IPv6 unicast addresses are aggregatable with prefixes of arbitrary bit-length, similar to IPv4 addresses under Classless Inter-Domain Routing.

There are several types of unicast addresses in IPv6, in particular, Global Unicast, site-local unicast (deprecated, see Section 2.5.7), and Link-Local unicast. There are also some special-purpose subtypes of Global Unicast, such as IPv6 addresses with embedded IPv4 addresses. Additional address types or subtypes can be defined in the future.

IPv6 nodes may have considerable or little knowledge of the internal structure of the IPv6 address, depending on the role the node plays (for instance, host versus router). At a minimum, a node may consider that unicast addresses (including its own) have no internal structure:



A slightly sophisticated host (but still rather simple) may additionally be aware of subnet prefix(es) for the link(s) it is attached to, where different addresses may have different values for n:



Though a very simple router may have no knowledge of the internal structure of IPv6 unicast addresses, routers will more generally have knowledge of one or more of the hierarchical boundaries for the operation of routing protocols. The known boundaries will differ from router to router, depending on what positions the router holds in the routing hierarchy.

Except for the knowledge of the subnet boundary discussed in the previous paragraphs, nodes should not make any assumptions about the structure of an IPv6 address.

#### 2.4.1. Interface Identifiers

Interface identifiers in IPv6 unicast addresses are used to identify interfaces on a link. They are required to be unique within a subnet prefix. It is recommended that the same interface identifier not be assigned to different nodes on a link. They may also be unique over a broader scope. The same interface identifier may be used on multiple interfaces on a single node, as long as they are attached to different subnets.

Interface IDs must be viewed outside of the node that created Interface ID as an opaque bit string without any internal structure.

Note that the uniqueness of interface identifiers is independent of the uniqueness of IPv6 addresses. For example, a Global Unicast address may be created with a local scope interface identifier and a Link-Local address may be created with a universal scope interface identifier.

For all unicast addresses, except those that start with the binary value 000, Interface IDs are required to be 64 bits long. If derived from an IEEE MAC-layer address, they must be constructed in Modified EUI-64 format.

Modified EUI-64 format-based interface identifiers may have universal scope when derived from a universal token (e.g., IEEE 802 48-bit MAC or IEEE EUI-64 identifiers [EUI64]) or may have local scope where a global token is not being used (e.g., serial links, tunnel end-points) or where global tokens are undesirable (e.g., temporary tokens for privacy [RFC4941]).

Modified EUI-64 format interface identifiers are formed by inverting the "u" bit (universal/local bit in IEEE EUI-64 terminology) when forming the interface identifier from IEEE EUI-64 identifiers. In the resulting Modified EUI-64 format, the "u" bit is set to one (1) to indicate universal scope, and it is set to zero (0) to indicate

local scope. The first three octets in binary of an IEEE EUI-64 identifier are as follows:

0	0 0	1 1	2
0	7 8	5 6	3
+-----+-----+-----+-----+-----+			
cccc	ccug	cccc	cccc cccc
+-----+-----+-----+-----+-----+			

written in Internet standard bit-order, where "u" is the universal/local bit, "g" is the individual/group bit, and "c" is the bits of the company\_id. Appendix A, "Creating Modified EUI-64 Format Interface Identifiers", provides examples on the creation of Modified EUI-64 format-based interface identifiers.

The motivation for inverting the "u" bit when forming an interface identifier is to make it easy for system administrators to hand configure non-global identifiers when hardware tokens are not available. This is expected to be the case for serial links and tunnel end-points, for example. The alternative would have been for these to be of the form 0200:0:0:1, 0200:0:0:2, etc., instead of the much simpler 0:0:0:1, 0:0:0:2, etc.

IPv6 nodes are not required to validate that interface identifiers created with modified EUI-64 tokens with the "u" bit set to universal are unique.

The details of forming interface identifiers are defined in the appropriate "IPv6 over <link>" specification, such as "IPv6 over Ethernet" [RFC2464], and "IPv6 over FDDI" [RFC2467].

#### 2.4.2. The Unspecified Address

The address 0:0:0:0:0:0:0:0 is called the unspecified address. It must never be assigned to any node. It indicates the absence of an address. One example of its use is in the Source Address field of any IPv6 packets sent by an initializing host before it has learned its own address.

The unspecified address must not be used as the destination address of IPv6 packets or in IPv6 Routing headers. An IPv6 packet with a source address of unspecified must never be forwarded by an IPv6 router.

### 2.4.3. The Loopback Address

The unicast address 0:0:0:0:0:0:0:1 is called the loopback address. It may be used by a node to send an IPv6 packet to itself. It must not be assigned to any physical interface. It is treated as having Link-Local scope, and may be thought of as the Link-Local unicast address of a virtual interface (typically called the "loopback interface") to an imaginary link that goes nowhere.

The loopback address must not be used as the source address in IPv6 packets that are sent outside of a single node. An IPv6 packet with a destination address of loopback must never be sent outside of a single node and must never be forwarded by an IPv6 router. A packet received on an interface with a destination address of loopback must be dropped.

### 2.4.4. Global Unicast Addresses

The general format for IPv6 Global Unicast addresses is as follows:

n bits	m bits	128-n-m bits
+-----+-----+-----+		
global routing prefix	subnet ID	interface ID
+-----+-----+-----+		

where the global routing prefix is a (typically hierarchically-structured) value assigned to a site (a cluster of subnets/links), the subnet ID is an identifier of a link within the site, and the interface ID is as defined in Section 2.5.1.

All Global Unicast addresses other than those that start with binary 000 have a 64-bit interface ID field (i.e.,  $n + m = 64$ ), formatted as described in Section 2.5.1. Global Unicast addresses that start with binary 000 have no such constraint on the size or structure of the interface ID field.

Examples of Global Unicast addresses that start with binary 000 are the IPv6 address with embedded IPv4 addresses described in Section 2.5.5. An example of global addresses starting with a binary value other than 000 (and therefore having a 64-bit interface ID field) can be found in [RFC3587].

### 2.4.5. IPv6 Addresses with Embedded IPv4 Addresses

Two types of IPv6 addresses are defined that carry an IPv4 address in the low-order 32 bits of the address. These are the "IPv4-Compatible IPv6 address" and the "IPv4-mapped IPv6 address".

#### 2.4.5.1. IPv4-Compatible IPv6 Address

The "IPv4-Compatible IPv6 address" was defined to assist in the IPv6 transition. The format of the "IPv4-Compatible IPv6 address" is as follows:

80 bits	16	32 bits
-----		-----
0000.....0000	0000	IPv4 address
-----		-----

Note: The IPv4 address used in the "IPv4-Compatible IPv6 address" must be a globally-unique IPv4 unicast address.

The "IPv4-Compatible IPv6 address" is now deprecated because the current IPv6 transition mechanisms no longer use these addresses. New or updated implementations are not required to support this address type.

#### 2.4.5.2. IPv4-Mapped IPv6 Address

A second type of IPv6 address that holds an embedded IPv4 address is defined. This address type is used to represent the addresses of IPv4 nodes as IPv6 addresses. The format of the "IPv4-mapped IPv6 address" is as follows:

80 bits	16	32 bits
-----		-----
0000.....0000	ffff	IPv4 address
-----		-----

See [RFC4038] for background on the usage of the "IPv4-mapped IPv6 address".

#### 2.4.6. Link-Local IPv6 Unicast Addresses

Link-Local addresses are for use on a single link. Link-Local addresses have the following format:

10 bits	54 bits	64 bits
-----	-----	-----
1111111010	0	interface ID
-----	-----	-----

Link-Local addresses are designed to be used for addressing on a single link for purposes such as automatic address configuration, neighbor discovery, or when no routers are present.

Routers must not forward any packets with Link-Local source or destination addresses to other links.

#### 2.4.7. Site-Local IPv6 Unicast Addresses

Site-Local addresses were originally designed to be used for addressing inside of a site without the need for a global prefix. Site-local addresses are now deprecated as defined in [RFC3879].

Site-Local addresses have the following format:

10 bits	54 bits	64 bits	
+-----+	+-----+	+-----+	+-----+
1111111011	subnet ID	interface ID	
+-----+	+-----+	+-----+	+-----+

The special behavior of this prefix defined in [RFC3513] must no longer be supported in new implementations (i.e., new implementations must treat this prefix as Global Unicast).

Existing implementations and deployments may continue to use this prefix.

#### 2.5. Anycast Addresses

An IPv6 anycast address is an address that is assigned to more than one interface (typically belonging to different nodes), with the property that a packet sent to an anycast address is routed to the "nearest" interface having that address, according to the routing protocols' measure of distance.

Anycast addresses are allocated from the unicast address space, using any of the defined unicast address formats. Thus, anycast addresses are syntactically indistinguishable from unicast addresses. When a unicast address is assigned to more than one interface, thus turning it into an anycast address, the nodes to which the address is assigned must be explicitly configured to know that it is an anycast address.

For any assigned anycast address, there is a longest prefix P of that address that identifies the topological region in which all interfaces belonging to that anycast address reside. Within the region identified by P, the anycast address must be maintained as a separate entry in the routing system (commonly referred to as a "host route"); outside the region identified by P, the anycast address may be aggregated into the routing entry for prefix P.

Note that in the worst case, the prefix P of an anycast set may be the null prefix, i.e., the members of the set may have no topological locality. In that case, the anycast address must be maintained as a separate routing entry throughout the entire Internet, which presents a severe scaling limit on how many such "global" anycast sets may be supported. Therefore, it is expected that support for global anycast sets may be unavailable or very restricted.

One expected use of anycast addresses is to identify the set of routers belonging to an organization providing Internet service. Such addresses could be used as intermediate addresses in an IPv6 Routing header, to cause a packet to be delivered via a particular service provider or sequence of service providers.

Some other possible uses are to identify the set of routers attached to a particular subnet, or the set of routers providing entry into a particular routing domain.

#### 2.5.1. Required Anycast Address

The Subnet-Router anycast address is predefined. Its format is as follows:

	n bits		128-n bits	
+	-----	+	-----	+
	subnet prefix		00000000000000	
+	-----	+	-----	+

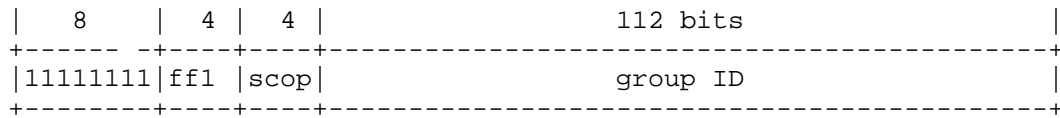
The "subnet prefix" in an anycast address is the prefix that identifies a specific link. This anycast address is syntactically the same as a unicast address for an interface on the link with the interface identifier set to zero.

Packets sent to the Subnet-Router anycast address will be delivered to one router on the subnet. All routers are required to support the Subnet-Router anycast addresses for the subnets to which they have interfaces.

The Subnet-Router anycast address is intended to be used for applications where a node needs to communicate with any one of the set of routers.

#### 2.6. Multicast Addresses

An IPv6 multicast address is an identifier for a group of interfaces (typically on different nodes). An interface may belong to any number of multicast groups. Multicast addresses have the following format:



binary 11111111 at the start of the address identifies the address as being a multicast address.

ffl is a set of 4 flags:

+--+--+--+
X R P T
+--+--+--+

The high-order flag is reserved, and must be initialized to 0.

T = 0 indicates a permanently-assigned ("well-known") multicast address, assigned by the Internet Assigned Numbers Authority (IANA).

T = 1 indicates a non-permanently-assigned ("transient" or "dynamically" assigned) multicast address.

The P flag's definition and usage can be found in [RFC3306] as updated by [RFC7371].

The R flag's definition and usage can be found in [RFC3956] as updated by [RFC7371].

The X flag's definition and usage can be found in [RFC3956] as updated by [RFC7371].

scop is a 4-bit multicast scope value used to limit the scope of the multicast group. The values are as follows:

- 0 reserved
- 1 Interface-Local scope
- 2 Link-Local scope
- 3 Realm-Local scope
- 4 Admin-Local scope
- 5 Site-Local scope
- 6 (unassigned)
- 7 (unassigned)
- 8 Organization-Local scope
- 9 (unassigned)
- A (unassigned)
- B (unassigned)



C (unassigned)  
D (unassigned)  
E Global scope  
F reserved

Interface-Local scope spans only a single interface on a node and is useful only for loopback transmission of multicast. Packets with interface-local scope received from another node must be discarded.

Link-Local multicast scope spans the same topological region as the corresponding unicast scope.

Interface-Local, Link-Local, and Realm-Local scope boundaries are automatically derived from physical connectivity or other non-multicast-related configurations. Global scope has no boundary. The boundaries of all other non-reserved scopes of Admin-Local or larger are administratively configured. For reserved scopes, the way of configuring their boundaries will be defined when the semantics of the scope are defined.

According to [RFC4007], the zone of a Realm-Local scope must fall within zones of larger scope. Because the zone of a Realm-Local scope is configured automatically while the zones of larger scopes are configured manually, care must be taken in the definition of those larger scopes to ensure that the inclusion constraint is met.

Realm-Local scopes created by different network technologies are considered to be independent and will have different zone indices (see Section 6 of [RFC4007]). A router with interfaces on links using different network technologies does not forward traffic between the Realm-Local multicast scopes defined by those technologies.

Site-Local scope is intended to span a single site.

Organization-Local scope is intended to span multiple sites belonging to a single organization.

scopes labeled "(unassigned)" are available for administrators to define additional multicast regions.

group ID identifies the multicast group, either permanent or transient, within the given scope. Additional definitions of the multicast group ID field structure are provided in [RFC3306].

The "meaning" of a permanently-assigned multicast address is independent of the scope value. For example, if the "NTP servers group" is assigned a permanent multicast address with a group ID of 101 (hex), then

ff01:0:0:0:0:0:0:101 means all NTP servers on the same interface (i.e., the same node) as the sender.

ff02:0:0:0:0:0:0:101 means all NTP servers on the same link as the sender.

ff05:0:0:0:0:0:0:101 means all NTP servers in the same site as the sender.

ff0e:0:0:0:0:0:0:101 means all NTP servers in the Internet.

Non-permanently-assigned multicast addresses are meaningful only within a given scope. For example, a group identified by the non-permanent, site-local multicast address ff15:0:0:0:0:0:0:101 at one site bears no relationship to a group using the same address at a different site, nor to a non-permanent group using the same group ID with a different scope, nor to a permanent group with the same group ID.

Multicast addresses must not be used as source addresses in IPv6 packets or appear in any Routing header.

Routers must not forward any multicast packets beyond the scope indicated by the scop field in the destination multicast address.

Nodes must not originate a packet to a multicast address whose scop field contains the reserved value 0; if such a packet is received, it must be silently dropped. Nodes should not originate a packet to a multicast address whose scop field contains the reserved value F; if such a packet is sent or received, it must be treated the same as packets destined to a global (scop E) multicast address.

#### 2.6.1. Pre-Defined Multicast Addresses

The following well-known multicast addresses are pre-defined. The group IDs defined in this section are defined for explicit scope values.

Use of these group IDs for any other scope values, with the T flag equal to 0, is not allowed.

reserved multicast addresses: ff00:0:0:0:0:0:0:0  
ff01:0:0:0:0:0:0:0  
ff02:0:0:0:0:0:0:0  
ff03:0:0:0:0:0:0:0  
ff04:0:0:0:0:0:0:0  
ff05:0:0:0:0:0:0:0  
ff06:0:0:0:0:0:0:0  
ff07:0:0:0:0:0:0:0  
ff08:0:0:0:0:0:0:0  
ff09:0:0:0:0:0:0:0  
ff0a:0:0:0:0:0:0:0  
ff0b:0:0:0:0:0:0:0  
ff0c:0:0:0:0:0:0:0  
ff0d:0:0:0:0:0:0:0  
ff0e:0:0:0:0:0:0:0  
ff0f:0:0:0:0:0:0:0

The above multicast addresses are reserved and shall never be assigned to any multicast group.

all nodes addresses: ff01:0:0:0:0:0:0:1  
ff02:0:0:0:0:0:0:1

The above multicast addresses identify the group of all IPv6 nodes, within scope 1 (interface-local) or 2 (link-local).

all routers addresses: ff01:0:0:0:0:0:0:2  
ff02:0:0:0:0:0:0:2  
ff05:0:0:0:0:0:0:2

The above multicast addresses identify the group of all IPv6 routers, within scope 1 (interface-local), 2 (link-local), or 5 (site-local).

Solicited-Node Address: ff02:0:0:0:0:1:ffxx:xxxx

Solicited-Node multicast address are computed as a function of a node's unicast and anycast addresses. A Solicited-Node multicast address is formed by taking the low-order 24 bits of an address (unicast or anycast) and appending those bits to the prefix FF02:0:0:0:0:1:FF00::/104 resulting in a multicast address in the range

ff02:0:0:0:0:1:ff00:0000

to

ff02:0:0:0:0:1:ffff:ffff

For example, the Solicited-Node multicast address corresponding to the IPv6 address 4037::01:800:200e:8c6c is ff02::1:ff0e:8c6c. IPv6 addresses that differ only in the high-order bits (e.g., due to multiple high-order prefixes associated with different aggregations) will map to the same Solicited-Node address, thereby reducing the number of multicast addresses a node must join.

A node is required to compute and join (on the appropriate interface) the associated Solicited-Node multicast addresses for all unicast and anycast addresses that have been configured for the node's interfaces (manually or automatically).

## 2.7. A Node's Required Addresses

A host is required to recognize the following addresses as identifying itself:

- o Its required Link-Local address for each interface.
- o Any additional Unicast and Anycast addresses that have been configured for the node's interfaces (manually or automatically).
- o The loopback address.
- o The All-Nodes multicast addresses defined in Section 2.7.1.
- o The Solicited-Node multicast address for each of its unicast and anycast addresses.
- o Multicast addresses of all other groups to which the node belongs.

A router is required to recognize all addresses that a host is required to recognize, plus the following addresses as identifying itself:

- o The Subnet-Router Anycast addresses for all interfaces for which it is configured to act as a router.
- o All other Anycast addresses with which the router has been configured.
- o The All-Routers multicast addresses defined in Section 2.7.1.

### 3. IANA Considerations

The "IPv4-Compatible IPv6 address" is deprecated by this document. The IANA should continue to list the address block containing these addresses at <http://www.iana.org/assignments/ipv6-address-space> as "Reserved by IETF" and not reassign it for any other purpose. For example:

0000::/8	Reserved by IETF	[RFC3513]	[1]
----------	------------------	-----------	-----

The IANA has added the following note and link to this address block.

[5] 0000::/96 was previously defined as the "IPv4-Compatible IPv6 address" prefix. This definition has been deprecated by [RFC4291].

The IANA has updated the references for the IPv6 Address Architecture in the IANA registries accordingly.

### 4. Security Considerations

IPv6 addressing documents do not have any direct impact on Internet infrastructure security. Authentication of IPv6 packets is defined in [RFC4302].

### 5. Acknowledgments

The authors would like to acknowledge the contributions of Paul Francis, Scott Bradner, Jim Bound, Brian Carpenter, Matt Crawford, Deborah Estrin, Roger Fajman, Bob Fink, Peter Ford, Bob Gilligan, Dmitry Haskin, Tom Harsch, Christian Huitema, Tony Li, Greg Minshall, Thomas Narten, Erik Nordmark, Yakov Rekhter, Bill Simpson, Sue Thomson, Markku Savela, Larry Masinter, Jun-ichiro Itojun Hagino, Tatuya Jinmei, Suresh Krishnan, and Mahmood Ali.

The authors would also like to acknowledge the authors of the updating RFCs that were incorporated in this version of the document to move IPv6 to Internet Standard. This includes Marcelo Bagnulo,

Congxiao Bao, Mohamed Boucadair, Brian Carpenter, Ralph Droms, Christian Huitema, Sheng Jiang, Seiichi Kawamura, Masanobu Kawashima, Xing Li, and Stig Venaas.

## 6. References

### 6.1. Normative References

- [I-D.hinden-6man-rfc2460bis]  
Deering, S. and B. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", draft-hinden-6man-rfc2460bis-07 (work in progress), September 2015.

### 6.2. Informative References

- [EUI64] "IEEE, "Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority"", March 1997, <<http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>>.
- [IANA-AD] "Internet Protocol Version 6 Address Space", <<https://www.iana.org/assignments/ipv6-address-space/ipv6-address-space.xhtml>>.
- [IANA-SP] "IANA IPv6 Special-Purpose Address Registry", <<https://www.iana.org/assignments/iana-ipv6-special-registry/iana-ipv6-special-registry.xhtml>>.
- [RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, DOI 10.17487/RFC2464, December 1998, <<http://www.rfc-editor.org/info/rfc2464>>.
- [RFC2467] Crawford, M., "Transmission of IPv6 Packets over FDDI Networks", RFC 2467, DOI 10.17487/RFC2467, December 1998, <<http://www.rfc-editor.org/info/rfc2467>>.
- [RFC3306] Haberman, B. and D. Thaler, "Unicast-Prefix-based IPv6 Multicast Addresses", RFC 3306, DOI 10.17487/RFC3306, August 2002, <<http://www.rfc-editor.org/info/rfc3306>>.
- [RFC3513] Hinden, R. and S. Deering, "Internet Protocol Version 6 (IPv6) Addressing Architecture", RFC 3513, DOI 10.17487/RFC3513, April 2003, <<http://www.rfc-editor.org/info/rfc3513>>.
- [RFC3587] Hinden, R., Deering, S., and E. Nordmark, "IPv6 Global Unicast Address Format", RFC 3587, DOI 10.17487/RFC3587, August 2003, <<http://www.rfc-editor.org/info/rfc3587>>.

- [RFC3879] Huitema, C. and B. Carpenter, "Deprecating Site Local Addresses", RFC 3879, DOI 10.17487/RFC3879, September 2004, <<http://www.rfc-editor.org/info/rfc3879>>.
- [RFC3956] Savola, P. and B. Haberman, "Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address", RFC 3956, DOI 10.17487/RFC3956, November 2004, <<http://www.rfc-editor.org/info/rfc3956>>.
- [RFC4007] Deering, S., Haberman, B., Jinmei, T., Nordmark, E., and B. Zill, "IPv6 Scoped Address Architecture", RFC 4007, DOI 10.17487/RFC4007, March 2005, <<http://www.rfc-editor.org/info/rfc4007>>.
- [RFC4038] Shin, M-K., Ed., Hong, Y-G., Hagino, J., Savola, P., and E. Castro, "Application Aspects of IPv6 Transition", RFC 4038, DOI 10.17487/RFC4038, March 2005, <<http://www.rfc-editor.org/info/rfc4038>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<http://www.rfc-editor.org/info/rfc4291>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<http://www.rfc-editor.org/info/rfc4302>>.
- [RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, <<http://www.rfc-editor.org/info/rfc4632>>.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<http://www.rfc-editor.org/info/rfc4941>>.
- [RFC7371] Boucadair, M. and S. Venaas, "Updates to the IPv6 Multicast Addressing Architecture", RFC 7371, DOI 10.17487/RFC7371, September 2014, <<http://www.rfc-editor.org/info/rfc7371>>.

#### Appendix A. Creating Modified EUI-64 Format Interface Identifiers

Depending on the characteristics of a specific link or node, there are a number of approaches for creating Modified EUI-64 format interface identifiers. This appendix describes some of these approaches.

## Links or Nodes with IEEE EUI-64 Identifiers

The only change needed to transform an IEEE EUI-64 identifier to an interface identifier is to invert the "u" (universal/local) bit. An example is a globally unique IEEE EUI-64 identifier of the form:

0	1	3	4	6
0	5	1	7	3
-----				
cccccc0gcccccccc ccccccccmmmmmmmmmm nnnnnnnnnnnnnnnnnnnnnn nnnnnnnnnnnnnnnnnnnnnn				
-----				

where "c" is the bits of the assigned company\_id, "0" is the value of the universal/local bit to indicate universal scope, "g" is individual/group bit, and "m" is the bits of the manufacturer-selected extension identifier. The IPv6 interface identifier would be of the form:

0	1	3	4	6
0	5	1	7	3
-----				
cccccc1gcccccccc ccccccccmmmmmmmmmm nnnnnnnnnnnnnnnnnnnnnn nnnnnnnnnnnnnnnnnnnnnn				
-----				

The only change is inverting the value of the universal/local bit.

## Links or Nodes with IEEE 802 48-bit MACs

[EUI64] defines a method to create an IEEE EUI-64 identifier from an IEEE 48-bit MAC identifier. This is to insert two octets, with hexadecimal values of 0xFF and 0xFE (see the Note at the end of appendix), in the middle of the 48-bit MAC (between the company\_id and vendor-supplied id). An example is the 48-bit IEEE MAC with Global scope:

0	1	3	4
0	5	1	7
-----			
cccccc0gcccccccc ccccccccmmmmmmmmmm nnnnnnnnnnnnnnnnnnnnnn			
-----			

where "c" is the bits of the assigned company\_id, "0" is the value of the universal/local bit to indicate Global scope, "g" is individual/group bit, and "m" is the bits of the manufacturer-selected extension identifier. The interface identifier would be of the form:



0	1	3	4	6
0	5	1	7	3
6	6	2	8	
-----				
cccccc gccccccc ccccccc 1111111 11111110mmmmmmmm mmmmmmmmmmmmmmmmmmmm				
-----				

When IEEE 802 48-bit MAC addresses are available (on an interface or a node), an implementation may use them to create interface identifiers due to their availability and uniqueness properties.

#### Links with Other Kinds of Identifiers

There are a number of types of links that have link-layer interface identifiers other than IEEE EUI-64 or IEEE 802 48-bit MACs. Examples include LocalTalk and Arcnet. The method to create a Modified EUI-64 format identifier is to take the link identifier (e.g., the LocalTalk 8-bit node identifier) and zero fill it to the left. For example, a LocalTalk 8-bit node identifier of hexadecimal value 0x4F results in the following interface identifier:

0	1	3	4	6
0	5	1	7	3
6	6	2	8	
-----				
0000000000000000 0000000000000000 0000000000000000 0000000001001111				
-----				

Note that this results in the universal/local bit set to "0" to indicate local scope.

#### Links without Identifiers

There are a number of links that do not have any type of built-in identifier. The most common of these are serial links and configured tunnels. Interface identifiers that are unique within a subnet prefix must be chosen.

When no built-in identifier is available on a link, the preferred approach is to use a universal interface identifier from another interface or one that is assigned to the node itself. When using this approach, no other interface connecting the same node to the same subnet prefix may use the same identifier.

If there is no universal interface identifier available for use on the link, the implementation needs to create a local-scope interface identifier. The only requirement is that it be unique within a subnet prefix. There are many possible approaches to select a subnet-prefix-unique interface identifier. These include the following:

Manual Configuration  
Node Serial Number  
Other Node-Specific Token

The subnet-prefix-unique interface identifier should be generated in a manner such that it does not change after a reboot of a node or if interfaces are added or deleted from the node.

The selection of the appropriate algorithm is link and implementation dependent. The details on forming interface identifiers are defined in the appropriate "IPv6 over <link>" specification. It is strongly recommended that a collision detection algorithm be implemented as part of any automatic algorithm.

Note: [EUI64] actually defines 0xFF and 0xFE as the bits to be inserted to create an IEEE EUI-64 identifier from an IEEE MAC-48 identifier. The 0xFF and 0xFE values are used when starting with an IEEE EUI-48 identifier. The incorrect value was used in earlier versions of the specification due to a misunderstanding about the differences between IEEE MAC-48 and EUI-48 identifiers.

This document purposely continues the use of 0xFF and 0xFE because it meets the requirements for IPv6 interface identifiers (i.e., that they must be unique on the link), IEEE EUI-48 and MAC-48 identifiers are syntactically equivalent, and that it doesn't cause any problems in practice.

#### Appendix B. CHANGES SINCE RFC 4291

This document has the following changes from RFC4291, "IP Version 6 Addressing Architecture". Numbers identify the Internet-Draft version that the change was made.:

- 06) Incorporate the updates made by RFC7371. The changes were to the flag bits and their definitions in Section 2.6.
- 05) Incorporate the updates made by RFC7346. The change was to add Realm-Local scope to the multicast scope table in Section 2.6, and add the updating text to the same section.
- 04) Incorporate the updates made by RFC6052. The change was to add a text in Section 2.3 that points to the IANA registries that records the prefix defined in RFC6052 and a number of other special use prefixes.

- 03) Incorporate the updates made by RFC7136 to deprecate the U and G bits in Modified EUI-64 format Internet IDs.
- 03) Add note to the reference section acknowledging the authors of the updating documents.
- 03) Editorial changes.
- 02) Updates to resolve the open Errata on RFC4291. These are:

Errata ID: 3480: Corrects the definition of Interface-Local multicast scope include that packets with interface-local scope received from another node must be discarded

Errata ID: 1627: Remove extraneous "of" in Section 2.7.

Errata ID: 2702: This errata is marked rejected. No change is required.

Errata ID: 2735: This errata is marked rejected. No change is required.

Errata ID: 4406: This errata is marked rejected. No change is required.

Errata ID: 2406: This errata is marked rejected. No change is required.

Errata ID: 863: This errata is marked rejected. No change is required.

Errata ID: 864: This errata is marked rejected. No change is required.

Errata ID: 866: This errata is marked rejected. No change is required.

- 02) Update references to current versions.
- 02) Editorial changes.
- 01) Incorporate the updates made by RFC5952 regarding the text format when outputting IPv6 addresses. A new section was added for this and addresses shown in this document were changed to lower case.

- 01) Revise this Section to document to show the the changes from RFC4291.
- 01) Editorial changes.
- 00) Establish a baseline from RFC4291. The only intended changes are formatting (XML is slightly different from .nroff), differences between an RFC and Internet Draft, fixing a few ID Nits, and updates to the authors information. There should not be any content changes to the specification.

#### Authors' Addresses

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

Email: bob.hinden@gmail.com

Stephen E. Deering  
Retired  
Vancouver, British Columbia  
Canada

IPv6 Maintenance  
Internet-Draft  
Updates: 4861 (if approved)  
Intended status: Standards Track  
Expires: April 10, 2017

F. Baker  
  
B. Carpenter  
Univ. of Auckland  
October 7, 2016

First-hop router selection by hosts in a multi-prefix network  
draft-ietf-6man-multi-homed-host-10

Abstract

This document describes expected IPv6 host behavior in a scenario that has more than one prefix, each allocated by an upstream network that is assumed to implement BCP 38 ingress filtering, when the host has multiple routers to choose from. It also applies to other scenarios such as the usage of stateful firewalls that effectively act as address-based filters. Host behavior in choosing a first-hop router may interact with source address selection in a given implementation. However, the selection of the source address for a packet is done before the first-hop router for that packet is chosen. Given that the network or host is, or appears to be, multihomed with multiple provider-allocated addresses, that the host has elected to use a source address in a given prefix, and that some but not all neighboring routers are advertising that prefix in their Router Advertisement Prefix Information Options, this document specifies to which router a host should present its transmission. It updates RFC 4861.

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 April 10, 2017.

## Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction and Applicability . . . . .	2
1.1. Host Model . . . . .	3
1.2. Requirements Language . . . . .	5
2. Sending context expected by the host . . . . .	5
2.1. Expectations the host has of the network . . . . .	5
2.2. Expectations of multihomed networks . . . . .	7
3. Reasonable expectations of the host . . . . .	7
3.1. Interpreting Router Advertisements . . . . .	7
3.2. Default Router Selection . . . . .	8
3.3. Source Address Selection . . . . .	9
3.4. Redirects . . . . .	9
3.5. History . . . . .	9
4. Residual issues . . . . .	10
5. IANA Considerations . . . . .	10
6. Security Considerations . . . . .	10
7. Acknowledgements . . . . .	10
8. References . . . . .	11
8.1. Normative References . . . . .	11
8.2. Informative References . . . . .	11
Appendix A. Change Log (RFC Editor: please delete) . . . . .	12
Authors' Addresses . . . . .	13

## 1. Introduction and Applicability

This document describes the expected behavior of an IPv6 [RFC2460] host in a network that has more than one prefix, each allocated by an upstream network that is assumed to implement BCP 38 [RFC2827] ingress filtering, and in which the host is presented with a choice of routers. It expects that the network will implement some form of egress routing, so that packets sent to a host outside the local network from a given ISP's prefix will go to that ISP. If the packet

is sent to the wrong egress, it is liable to be discarded by the BCP 38 filter. However, the mechanics of egress routing once the packet leaves the host are out of scope. The question here is how the host interacts with that network.

Various aspects of this issue, and possible solution approaches, are discussed in the document IPv6 Multihoming without Network Address Translation [RFC7157].

BCP 38 filtering by ISPs is not the only scenario where such behavior is valuable. Implementations that combine existing recommendations, such as [RFC6092] [RFC7084] can also result in such filtering. Another case is when the connections to the upstream networks include stateful firewalls, such that return packets in a stream will be discarded if they do not return via the firewall that created state for the outgoing packets. A similar cause of such discards is unicast reverse path forwarding (uRPF) [RFC3704].

In this document, the term "filter" is used for simplicity to cover all such cases. In any case, one cannot assume the host to be aware whether an ingress filter, a stateful firewall, or any other type of filter is in place. Therefore, the only known consistent solution is to implement the features defined in this document.

Note that, apart from ensuring that a message with a given source address is given to a first-hop router that appears to know about the prefix in question, this specification is consistent with [RFC4861]. Nevertheless, implementers of Sections 6.2.3, 6.3.4, 6.3.6 and 8.1 of RFC 4861 should extend their implementations accordingly. This specification is fully consistent with [RFC6724] and depends on support for its Rule 5.5 (see Section 3.3). Hosts that do not support these features may fail to communicate in the presence of filters as described above.

#### 1.1. Host Model

It could be argued that the proposal of this document, which is to send messages using a source address in a given prefix to the router that advertised the prefix in its Router Advertisement (RA), is a form of [RFC1122]'s Strong End System (ES, e.g. Host) Model, discussed in section 3.3.4.2 of that document. In short, [RFC1122] identifies two basic models, in which the "strong host" model models the host as a set of hosts in one chassis, each of which uses a single address on a single interface, and always both sends and receives on that interface, and the "weak host" model treats the host as one system with zero or more addresses on every interface, and capable of using any interface for any communication. As noted there, neither model is completely satisfactory. For example, a host

with a link-local-only interface and a default route pointing to that interface will necessarily send packets using that interface but with a source address derived from some other interface, and will therefore be a de facto weak host. If the router upstream from such a host implements BCP 38 Ingress Filtering [RFC2827], such as by implementing uRPF on each interface, the router might prevent communication by weak hosts.

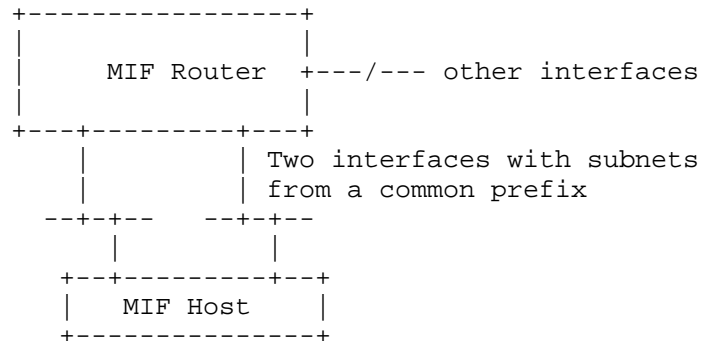


Figure 1: Hypothetical MIF interconnection

The proposal also differs slightly from [RFC1122]'s language for the Strong Host Model. The proposal is that the packet will go to a router that advertised a given prefix, but does not specify what interface that might happen on. Hence, if the router is a multi-interface (MIF) router and is using a common prefix spanning two or more LANs shared by the host (as in Figure 1), the host might use either of those LANs, according to this proposal. The Strong Host Model is not stated in those terms, but in terms of the interface used. A Strong host would treat such a MIF router as two separate routers when obeying the rules from RFC 1122 as they apply in the Strong case:

- (A) A host MUST silently discard an incoming datagram whose destination address does not correspond to the physical interface through which it is received.
- (B) A host MUST restrict itself to sending (non-source- routed) IP datagrams only through the physical interface that corresponds to the IP source address of the datagrams.

However, comparing the presumptive route lookup mechanisms in each model, this proposal is indeed most similar to the Strong Host Model, as is any source/destination routing paradigm.

Strong:    route(src IP addr, dest IP addr, TOS) -> gateway



Weak:    route(dest IP addr, TOS) -> gateway, interface

In the hypothetical MIF model suggested in Figure 1, the address fails to identify a single interface, but it does identify a single gateway.

## 1.2. 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 [RFC2119].

## 2. Sending context expected by the host

### 2.1. Expectations the host has of the network

A host receives prefixes in a Router Advertisement [RFC4861], which goes on to identify whether they are usable by SLAAC [RFC4862] with any type of interface identifier [RFC4941] [RFC7217]. When no prefixes are usable for SLAAC, the Router Advertisement would normally signal the availability of DHCPv6 [RFC3315] and the host would use it to configure its addresses. In the latter case (or if both SLAAC and DHCPv6 are used on the same link for some reason) it will generally be the case that the configured addresses match one of the prefixes advertised in a Router Advertisement that are supposed to be on-link for that link.

The simplest multihomed network implementation in which a host makes choices among routers might be a LAN with one or more hosts on it and two or more routers, one for each upstream network, or a host that is served by disjoint networks on separate interfaces. In such a network, especially the latter, there is not necessarily a routing protocol, and the two routers may not even know that the other is a router as opposed to a host, or may be configured to ignore its presence. One might expect that the routers may or may not receive each other's RAs and form an address in the other router's prefix (which is not per [RFC4862], but is implemented by some stub router implementations). However, all hosts in such a network might be expected to create an address in each prefix so advertised.

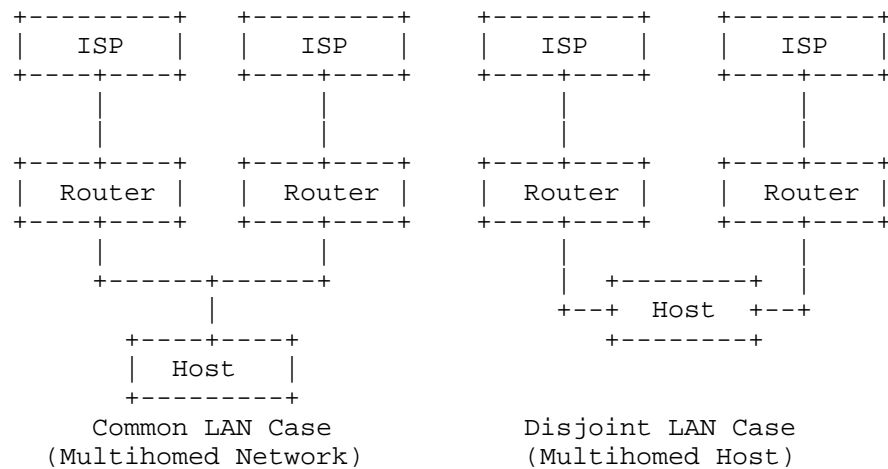


Figure 2: Two simple networks

If there is no routing protocol among those routers, there is no mechanism by which packets can be deterministically forwarded between the routers (as described in BCP 84 [RFC3704]) in order to avoid filters. Even if there was routing, it would result in an indirect route, rather than a direct route originating with the host; this is not "wrong", but can be inefficient. Therefore the host would do well to select the appropriate router itself.

Since the host derives fundamental default routing information from the Router Advertisement, this implies that, in any network with hosts using multiple prefixes, each prefix SHOULD be advertised via a Prefix Information Option (PIO) [RFC4861] by one of the attached routers, even if addresses are being assigned using DHCPv6. A router that advertises a prefix indicates that it is able to appropriately route packets with source addresses within that prefix, regardless of the setting of the L and A flags in the PIO.

In some circumstances both L and A might be zero. If SLAAC is not wanted (A=0) and there is no reason to announce an on-link prefix (L=0), a PIO SHOULD be sent to inform hosts that they should use the router in question as the first hop for packets with source addresses in the PIO prefix. An example case is the MIF router in Figure 1, which could send PIOs with A=L=0 for the common prefix. Although this does not violate the existing standard [RFC4861], such a PIO has not previously been common, and it is possible that existing host implementations simply ignore such a PIO or that existing router implementations are not capable of sending such a PIO. Newer implementations that support this mechanism should be updated accordingly:

- o A host SHOULD NOT ignore a PIO simply because both L and A flags are cleared (extending Section 6.3.4 of [RFC4861]).
- o A router SHOULD be able to send such a PIO (extending Section 6.2.3 of [RFC4861]).

## 2.2. Expectations of multihomed networks

Networking equipment needs to support source/destination routing for at least some of the routes in the Forwarding Information Base (FIB), such as default egress routes differentiated by source prefix. Installation of source/destination routes in the FIB might be accomplished using static routes, SDN technologies, or dynamic routing protocols.

## 3. Reasonable expectations of the host

### 3.1. Interpreting Router Advertisements

As described in [RFC4191] and [RFC4861], a Router Advertisement may contain zero or more Prefix information Options (PIOs), or zero or more Route Information Options (RIOs). In their original intent, these indicate general information to a host: "the router whose address is found in the source address field of this packet is one of your default routers", "you might create an address in this prefix", or "this router would be a good place to send traffic directed to a given destination prefix". In a multi-prefix network with multiple exits, the host's characterization of each default router SHOULD include the prefixes it has announced (extending Section 6.3.4 of [RFC4861]). In other words, the PIO is reinterpreted to also imply that the advertising router would be a reasonable first hop for any packet using a source address in any advertised prefix, regardless of Default Router Preference.

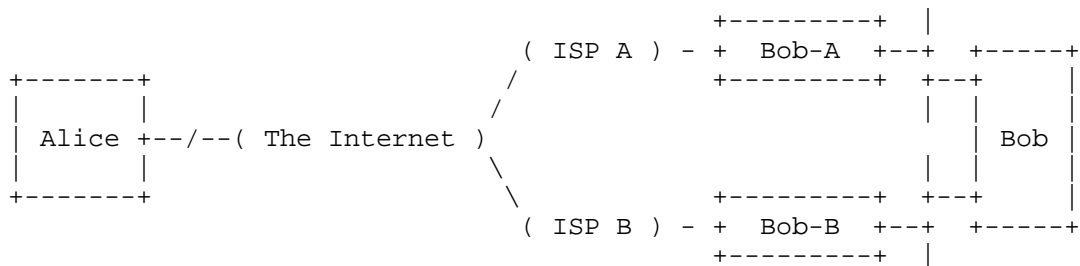


Figure 3: PIOs, RIOs, and Default Routes

The implications bear consideration. Imagine, Figure 3, that hosts Alice and Bob are in communication. Bob's network consists at least

of Bob (the computer), 2 routers (Bob-A and Bob-B), and the links between them; it may be much larger, for example a campus or corporate network. Bob's network is therefore multihomed, and Bob's first hop routers are Bob-A (to upstream ISP A advertising prefix PA) and Bob-B (to upstream network B and advertising prefix PB). We assume that Bob is not applying Rule 5.5 of [RFC6724]. If Bob is responding to a message from Alice, his choice of source address is forced to be the address Alice used as a destination (which we may presume to have been in prefix PA). Hence, Bob created or was assigned an address in PA, and can only reasonably send traffic using it to Bob-A as a first hop router. If there are several routers in Bob's network advertising the prefix PA (referred to as Bob-Ax routers), then Bob should choose its first-hop router only from among those routers. From among the multiple Bob-Ax routers, Bob should choose the first-hop router based on the criteria specified in Section 3 of [RFC4191]. If none of the Bob-Ax routers has advertised an RA with a non-zero Router Lifetime or an RIO with a non-zero Route Lifetime that includes Alice, but router Bob-B has, it is irrelevant. Bob is using the address allocated in PA and courts a BCP 38 discard if he doesn't send the packet to Bob-A.

In the special case that Bob is initiating the conversation, an RIO might, however, influence source address choice. Bob could presumably use any address allocated to him, in this case his address in PA or PB. If Bob-B has advertised an RIO for Alice's prefix and Bob-A has not, Bob MAY take that fact into account in address selection - choosing an address that would allow him to make use of the RIO.

### 3.2. Default Router Selection

Default Router Selection (Section 6.3.6 of [RFC4861]) is extended as follows: A host SHOULD select default routers for each prefix it is assigned an address in. Routers that have advertised the prefix in its Router Advertisement message SHOULD be preferred over routers that do not advertise the prefix, regardless of Default Router Preference. Note that this document does not change the way in which default router preferences are communicated [RFC4191].

If no router has advertised the prefix in an RA, normal routing metrics will apply. An example is a host connected to the Internet via one router, and at the same time connected by a VPN to a private domain which is also connected to the global Internet.

As a result of this, when a host sends a packet using a source address in one of those prefixes and has no history directing it otherwise, it SHOULD send it to the indicated default router. In the "simplest" network described in Section 2.1, that would get it to the

only router that is directly capable of getting it to the right ISP. This will also apply in more complex networks, even when more than one physical or virtual interface is involved.

In more complex cases, wherein routers advertise RAs for multiple prefixes whether or not they have direct or isolated upstream connectivity, the host is dependent on the routing system already. If the host gives the packet to a router advertising its source prefix, it should be able to depend on the router to do the right thing.

### 3.3. Source Address Selection

There is an interaction with Default Address Selection [RFC6724]. A host following the recommendation in the previous section will store information about which next-hops advertised which prefixes. Rule 5.5 of RFC 6724 states that the source address used to send to a given destination address should if possible be chosen from a prefix known to be advertised by the next-hop router for that destination. This selection rule **SHOULD** therefore be implemented in a host following the recommendation in the previous section.

### 3.4. Redirects

There is potential for adverse interaction with any off-link Redirect (Redirect for a destination that is not on-link) message sent by a router in accordance with Section 8 of [RFC4861]. Hosts **SHOULD** apply off-link redirects only for the specific pair of source and destination addresses concerned, so the host's Destination Cache might need to contain appropriate source-specific entries. This extends the validity check specified in Section 8.1 of [RFC4861].

### 3.5. History

Some modern hosts maintain history, in terms of what has previously worked or not worked for a given address or prefix and in some cases the effective window and MSS values for TCP or other protocols. This might include a next hop address for use when a packet is sent to the indicated address.

When such a host makes a successful exchange with a remote destination using a particular address pair, and the host has previously received a PIO that matches the source address, then the host **SHOULD** include the prefix in such history, whatever the setting of the L and A flags in the PIO. On subsequent attempts to communicate with that destination, if it has an address in that prefix at that time, a host **MAY** use an address in the remembered prefix for the session.

#### 4. Residual issues

Consider a network where routers on a link run a routing protocol and are configured with the same information. Thus, on each link all routers advertise all prefixes on the link. The assumption that packets will be forwarded to the appropriate egress by the local routing system might cause at least one extra hop in the local network (from the host to the wrong router, and from there to another router on the same link).

In a slightly more complex situation such as the disjoint LAN case of Figure 2, for example a home plus corporate home-office configuration, the two upstream routers might be on different LANs and therefore different subnets (e.g., the host is itself multi-homed). In that case, there is no way for the "wrong" router to detect the existence of the "right" router, or to route to it.

In such a case it is particularly important that hosts take the responsibility to memorize and select the best first-hop as described in Section 3.

#### 5. IANA Considerations

This memo asks the IANA for no new parameters.

#### 6. Security Considerations

This document is intended to avoid connectivity issues in the presence of BCP 38 ingress filters or stateful firewalls combined with multihoming. It does not in itself create any new security or privacy exposures. However, since the solution is designed to ensure that routing occurs correctly in situations where it previously failed, this might result in unexpected exposure of networks that were previously unreachable.

There might be a small privacy improvement: with the current practice, a multihomed host that sends packets with the wrong address to an upstream router or network discloses the prefix of one upstream to the other upstream network. This practice reduces the probability of that occurrence.

#### 7. Acknowledgements

Comments were received from Jinmei Tatuya and Ole Troan, who have suggested important text, plus Mikael Abrahamsson, Steven Barth, Carlos Bernardos Cano, Chris Bowers, Zhen Cao, Juliusz Chroboczek, Toerless Eckert, David Farmer, Bob Hinden, Ben Laurie, Dusan Mudric,

Tadahisa Okimoto, Pierre Pfister, Behcet Sarikaya, Mark Smith and  
James Woodyatt.

## 8. References

### 8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<http://www.rfc-editor.org/info/rfc2460>>.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <<http://www.rfc-editor.org/info/rfc4191>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<http://www.rfc-editor.org/info/rfc4861>>.
- [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <<http://www.rfc-editor.org/info/rfc6724>>.

### 8.2. Informative References

- [RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <<http://www.rfc-editor.org/info/rfc1122>>.
- [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, <<http://www.rfc-editor.org/info/rfc2827>>.
- [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, <<http://www.rfc-editor.org/info/rfc3315>>.

- [RFC3704] Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks", BCP 84, RFC 3704, DOI 10.17487/RFC3704, March 2004, <<http://www.rfc-editor.org/info/rfc3704>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<http://www.rfc-editor.org/info/rfc4862>>.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<http://www.rfc-editor.org/info/rfc4941>>.
- [RFC6092] Woodyatt, J., Ed., "Recommended Simple Security Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service", RFC 6092, DOI 10.17487/RFC6092, January 2011, <<http://www.rfc-editor.org/info/rfc6092>>.
- [RFC7084] Singh, H., Beebee, W., Donley, C., and B. Stark, "Basic Requirements for IPv6 Customer Edge Routers", RFC 7084, DOI 10.17487/RFC7084, November 2013, <<http://www.rfc-editor.org/info/rfc7084>>.
- [RFC7157] Troan, O., Ed., Miles, D., Matsushima, S., Okimoto, T., and D. Wing, "IPv6 Multihoming without Network Address Translation", RFC 7157, DOI 10.17487/RFC7157, March 2014, <<http://www.rfc-editor.org/info/rfc7157>>.
- [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, <<http://www.rfc-editor.org/info/rfc7217>>.

Appendix A. Change Log (RFC Editor: please delete)

Initial Version: 2015-08-05

Version 01: Update text on PIOs, added text on Redirects, and clarified the concept of a "simple" network, 2015-08-13.

Version 02: Clarifications after WG discussions, 2015-08-19.

Version 03: More clarifications after more WG discussions, especially adding stateful firewalls, uRPF, and more precise discussion of RFC 4861, 2015-09-03.



Version 04:    Responds to various comments including

- \*    Questions regarding RFC 1122's strong and weak host models.  
     This model is, strictly speaking, neither, but is most similar  
     to the strong host model.
- \*    Some wording errors.
- \*    Requests for discussion of the handling of the RIO, PIO, and  
     Default Router List in an RA.

WG Versions 00-02:    More clarifications after more WG discussions,  
2015-11-03.

WG Version 03:    A final clarification re uRPF, 2015-12-15.

WG Versions 04-07:    Various clarifications and review comments,  
2016-06-23.

WG Version 08-10:    Fixes for IETF Last Call and IESG comments,  
2016-10-07.

#### Authors' Addresses

Fred Baker  
Santa Barbara, California 93117  
USA

Email: FredBaker.IETF@gmail.com

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

Email: brian.e.carpenter@gmail.com

Network Working Group  
Internet-Draft  
Obsoletes: 2460 (if approved)  
Intended status: Standards Track  
Expires: November 20, 2017

S. Deering  
Retired  
R. Hinden  
Check Point Software  
May 19, 2017

Internet Protocol, Version 6 (IPv6) Specification  
draft-ietf-6man-rfc2460bis-13

Abstract

This document specifies version 6 of the Internet Protocol (IPv6).  
It obsoletes RFC2460

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 November 20, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the  
document authors. All rights reserved.

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

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

## Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	4
3. IPv6 Header Format . . . . .	5
4. IPv6 Extension Headers . . . . .	6
4.1. Extension Header Order . . . . .	8
4.2. Options . . . . .	9
4.3. Hop-by-Hop Options Header . . . . .	12
4.4. Routing Header . . . . .	12
4.5. Fragment Header . . . . .	14
4.6. Destination Options Header . . . . .	21
4.7. No Next Header . . . . .	22
4.8. Defining New Extension Headers and Options . . . . .	22
5. Packet Size Issues . . . . .	23
6. Flow Labels . . . . .	24
7. Traffic Classes . . . . .	24
8. Upper-Layer Protocol Issues . . . . .	24
8.1. Upper-Layer Checksums . . . . .	25
8.2. Maximum Packet Lifetime . . . . .	26
8.3. Maximum Upper-Layer Payload Size . . . . .	27
8.4. Responding to Packets Carrying Routing Headers . . . . .	27
9. IANA Considerations . . . . .	27
10. Security Considerations . . . . .	28
11. Acknowledgments . . . . .	30
12. References . . . . .	30
12.1. Normative References . . . . .	30
12.2. Informative References . . . . .	31
Appendix A. Formatting Guidelines for Options . . . . .	33
Appendix B. Changes Since RFC2460 . . . . .	36
B.1. Change History Since RFC2460 . . . . .	39
Authors' Addresses . . . . .	45

## 1. Introduction

IP version 6 (IPv6) is a new version of the Internet Protocol (IP), designed as the successor to IP version 4 (IPv4) [RFC0791]. The changes from IPv4 to IPv6 fall primarily into the following categories:

- o Expanded Addressing Capabilities

IPv6 increases the IP address size from 32 bits to 128 bits, to support more levels of addressing hierarchy, a much greater number of addressable nodes, and simpler auto-configuration of addresses. The scalability of multicast routing is improved by adding a "scope" field to multicast addresses. And a new type of address called an "anycast address" is defined, used to send a packet to any one of a group of nodes.

- o Header Format Simplification

Some IPv4 header fields have been dropped or made optional, to reduce the common-case processing cost of packet handling and to limit the bandwidth cost of the IPv6 header.

- o Improved Support for Extensions and Options

Changes in the way IP header options are encoded allows for more efficient forwarding, less stringent limits on the length of options, and greater flexibility for introducing new options in the future.

- o Flow Labeling Capability

A new capability is added to enable the labeling of sequences of packets that the sender requests to be treated in the network as a single flow.

- o Authentication and Privacy Capabilities

Extensions to support authentication, data integrity, and (optional) data confidentiality are specified for IPv6.

This document specifies the basic IPv6 header and the initially-defined IPv6 extension headers and options. It also discusses packet size issues, the semantics of flow labels and traffic classes, and the effects of IPv6 on upper-layer protocols. The format and semantics of IPv6 addresses are specified separately in [RFC4291].

The IPv6 version of ICMP, which all IPv6 implementations are required to include, is specified in [RFC4443]

The data transmission order for IPv6 is the same as for IPv4 as defined in Appendix B of [RFC0791].

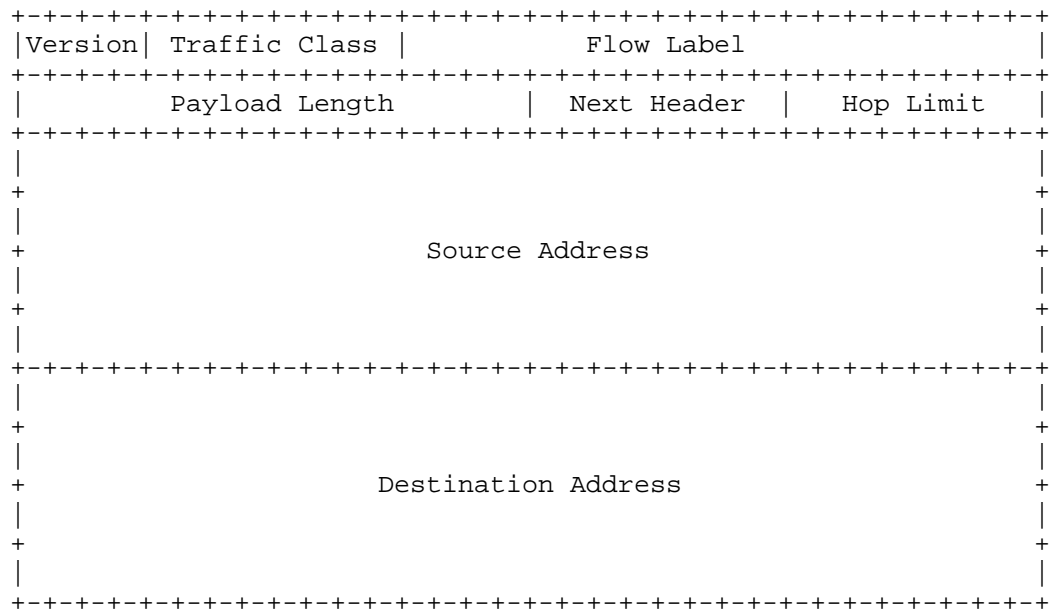
Note: As this document obsoletes [RFC2460], any document referenced in this document that includes pointers to RFC2460, should be interpreted as referencing this document.

## 2. Terminology

node	a device that implements IPv6.
router	a node that forwards IPv6 packets not explicitly addressed to itself. [See Note below].
host	any node that is not a router. [See Note below].
upper layer	a protocol layer immediately above IPv6. Examples are transport protocols such as TCP and UDP, control protocols such as ICMP, routing protocols such as OSPF, and internet or lower-layer protocols being "tunneled" over (i.e., encapsulated in) IPv6 such as IPX, AppleTalk, or IPv6 itself.
link	a communication facility or medium over which nodes can communicate at the link layer, i.e., the layer immediately below IPv6. Examples are Ethernet (simple or bridged); PPP links; X.25, Frame Relay, or ATM networks; and internet (or higher) layer "tunnels", such as tunnels over IPv4 or IPv6 itself.
neighbors	nodes attached to the same link.
interface	a node's attachment to a link.
address	an IPv6-layer identifier for an interface or a set of interfaces.
packet	an IPv6 header plus payload.
link MTU	the maximum transmission unit, i.e., maximum packet size in octets, that can be conveyed over a link.
path MTU	the minimum link MTU of all the links in a path between a source node and a destination node.

Note: it is possible for a device with multiple interfaces to be configured to forward non-self-destined packets arriving from some set (fewer than all) of its interfaces, and to discard non-self-destined packets arriving from its other interfaces. Such a device must obey the protocol requirements for routers when receiving packets from, and interacting with neighbors over, the former (forwarding) interfaces. It must obey the protocol requirements for hosts when receiving packets from, and interacting with neighbors over, the latter (non-forwarding) interfaces.

### 3. IPv6 Header Format



Version	4-bit Internet Protocol version number = 6.
Traffic Class	8-bit traffic class field. See section 7.
Flow Label	20-bit flow label. See section 6.
Payload Length	16-bit unsigned integer. Length of the IPv6 payload, i.e., the rest of the packet following this IPv6 header, in octets. (Note that any extension headers [Section 4] present are considered part of the payload, i.e., included in the length count.)

Next Header	8-bit selector. Identifies the type of header immediately following the IPv6 header. Uses the same values as the IPv4 Protocol field [IANA-PN].
Hop Limit	8-bit unsigned integer. Decremented by 1 by each node that forwards the packet. When forwarding, the packet is discarded if Hop Limit was zero when received or is decremented to zero. A node that is the destination of a packet should not discard a packet with hop limit equal to zero, it should process the packet normally.
Source Address	128-bit address of the originator of the packet. See [RFC4291].
Destination Address	128-bit address of the intended recipient of the packet (possibly not the ultimate recipient, if a Routing header is present). See [RFC4291] and section 4.4.

#### 4. IPv6 Extension Headers

In IPv6, optional internet-layer information is encoded in separate headers that may be placed between the IPv6 header and the upper-layer header in a packet. There is a small number of such extension headers, each one identified by a distinct Next Header value.

Extension Headers are numbered from IANA IP Protocol Numbers [IANA-PN], the same values used for IPv4 and IPv6. When processing a sequence of Next Header values in a packet, the first one that is not an Extension Header [IANA-EH] indicates that the next item in the packet is the corresponding upper-layer header. A special "No Next Header" value is used if there is no upper-layer header.

As illustrated in these examples, an IPv6 packet may carry zero, one, or more extension headers, each identified by the Next Header field of the preceding header:

IPv6 header	TCP header + data
Next Header = TCP	

IPv6 header	Routing header	TCP header + data
Next Header = Routing	Next Header = TCP	

IPv6 header	Routing header	Fragment header	fragment of TCP header + data
Next Header = Routing	Next Header = Fragment	Next Header = TCP	

Extension headers (except for the Hop-by-Hop Options header) are not processed, inserted, or deleted 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 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.

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.

At the Destination node, normal demultiplexing on the Next Header field of the IPv6 header invokes the module to process the first extension header, or the upper-layer header if no extension header is present. The contents and semantics of each extension header determine whether or not to proceed to the next header. Therefore, extension headers must be processed strictly in the order they appear in the packet; a receiver must not, for example, scan through a



packet looking for a particular kind of extension header and process that header prior to processing all preceding ones.

If, as a result of processing a header, the destination node is required to proceed to the next header but the Next Header value in the current header is unrecognized by the node, it should discard the packet and send an ICMP Parameter Problem message to the source of the packet, with an ICMP Code value of 1 ("unrecognized Next Header type encountered") and the ICMP Pointer field containing the offset of the unrecognized value within the original packet. The same action should be taken if a node encounters a Next Header value of zero in any header other than an IPv6 header.

Each extension header is an integer multiple of 8 octets long, in order to retain 8-octet alignment for subsequent headers. Multi-octet fields within each extension header are aligned on their natural boundaries, i.e., fields of width *n* octets are placed at an integer multiple of *n* octets from the start of the header, for *n* = 1, 2, 4, or 8.

A full implementation of IPv6 includes implementation of the following extension headers:

- Hop-by-Hop Options
- Fragment
- Destination Options
- Routing
- Authentication
- Encapsulating Security Payload

The first four are specified in this document; the last two are specified in [RFC4302] and [RFC4303], respectively. The current list of IPv6 extension headers can be found at [IANA-EH].

#### 4.1. Extension Header Order

When more than one extension header is used in the same packet, it is recommended that those headers appear in the following order:

- IPv6 header
- Hop-by-Hop Options header
- Destination Options header (note 1)
- Routing header
- Fragment header
- Authentication header (note 2)
- Encapsulating Security Payload header (note 2)
- Destination Options header (note 3)
- upper-layer header

note 1: for options to be processed by the first destination that appears in the IPv6 Destination Address field plus subsequent destinations listed in the Routing header.

note 2: additional recommendations regarding the relative order of the Authentication and Encapsulating Security Payload headers are given in [RFC4303].

note 3: for options to be processed only by the final destination of the packet.

Each extension header should occur at most once, except for the Destination Options header which should occur at most twice (once before a Routing header and once before the upper-layer header).

If the upper-layer header is another IPv6 header (in the case of IPv6 being tunneled over or encapsulated in IPv6), it may be followed by its own extension headers, which are separately subject to the same ordering recommendations.

If and when other extension headers are defined, their ordering constraints relative to the above listed headers must be specified.

IPv6 nodes must accept and attempt to process extension headers in any order and occurring any number of times in the same packet, except for the Hop-by-Hop Options header which is restricted to appear immediately after an IPv6 header only. Nonetheless, it is strongly advised that sources of IPv6 packets adhere to the above recommended order until and unless subsequent specifications revise that recommendation.

## 4.2. Options

Two of the currently-defined extension headers defined in this document -- the Hop-by-Hop Options header and the Destination Options header -- carry a variable number of type-length-value (TLV) encoded "options", of the following format:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
| Option Type | Opt Data Len | Option Data |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Option Type                      8-bit identifier of the type of option.

Opt Data Len                    8-bit unsigned integer. Length of the Option Data field of this option, in octets.

Option Data                      Variable-length field.    Option-Type-specific data.

The sequence of options within a header must be processed strictly in the order they appear in the header; a receiver must not, for example, scan through the header looking for a particular kind of option and process that option prior to processing all preceding ones.

The Option Type identifiers are internally encoded such that their highest-order two bits specify the action that must be taken if the processing IPv6 node does not recognize the Option Type:

- 00 - skip over this option and continue processing the header.
- 01 - discard the packet.
- 10 - discard the packet and, regardless of whether or not the packet's Destination Address was a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.
- 11 - discard the packet and, only if the packet's Destination Address was not a multicast address, send an ICMP Parameter Problem, Code 2, message to the packet's Source Address, pointing to the unrecognized Option Type.

The third-highest-order bit of the Option Type specifies whether or not the Option Data of that option can change en-route to the packet's final destination. When an Authentication header is present in the packet, for any option whose data may change en-route, its entire Option Data field must be treated as zero-valued octets when computing or verifying the packet's authenticating value.

- 0 - Option Data does not change en-route
- 1 - Option Data may change en-route

The three high-order bits described above are to be treated as part of the Option Type, not independent of the Option Type. That is, a particular option is identified by a full 8-bit Option Type, not just the low-order 5 bits of an Option Type.

The same Option Type numbering space is used for both the Hop-by-Hop Options header and the Destination Options header. However, the specification of a particular option may restrict its use to only one of those two headers.

Individual options may have specific alignment requirements, to ensure that multi-octet values within Option Data fields fall on natural boundaries. The alignment requirement of an option is specified using the notation  $xn+y$ , meaning the Option Type must appear at an integer multiple of  $x$  octets from the start of the header, plus  $y$  octets. For example:

$2n$  means any 2-octet offset from the start of the header.  
 $8n+2$  means any 8-octet offset from the start of the header, plus 2 octets.

There are two padding options which are used when necessary to align subsequent options and to pad out the containing header to a multiple of 8 octets in length. These padding options must be recognized by all IPv6 implementations:

Pad1 option (alignment requirement: none)

```
+-----+
|           0           |
+-----+
```

NOTE! the format of the Pad1 option is a special case -- it does not have length and value fields.

The Pad1 option is used to insert one octet of padding into the Options area of a header. If more than one octet of padding is required, the PadN option, described next, should be used, rather than multiple Pad1 options.

PadN option (alignment requirement: none)

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|           1           | Opt Data Len | Option Data |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

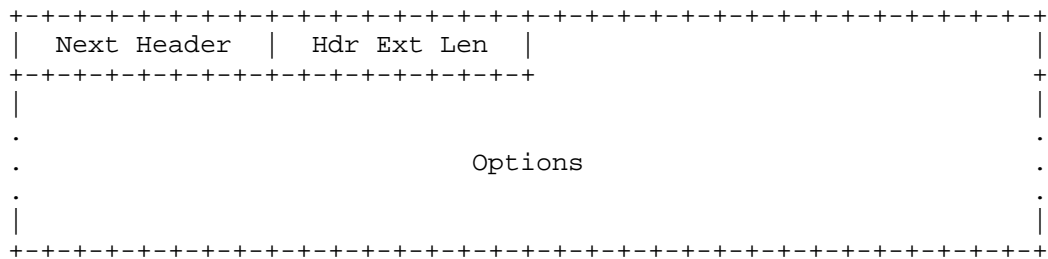
The PadN option is used to insert two or more octets of padding into the Options area of a header. For  $N$  octets of padding, the

Opt Data Len field contains the value N-2, and the Option Data consists of N-2 zero-valued octets.

Appendix A contains formatting guidelines for designing new options.

#### 4.3. Hop-by-Hop Options Header

The Hop-by-Hop Options header is used to carry optional information that may be examined and processed by every node along a packet's delivery path. The Hop-by-Hop Options header is identified by a Next Header value of 0 in the IPv6 header, and has the following format:



Next Header	8-bit selector. Identifies the type of header immediately following the Hop-by-Hop Options header. Uses the same values as the IPv4 Protocol field [IANA-PN].
Hdr Ext Len	8-bit unsigned integer. Length of the Hop-by-Hop Options header in 8-octet units, not including the first 8 octets.
Options	Variable-length field, of length such that the complete Hop-by-Hop Options header is an integer multiple of 8 octets long. Contains one or more TLV-encoded options, as described in section 4.2.

The only hop-by-hop options defined in this document are the Pad1 and PadN options specified in section 4.2.

#### 4.4. Routing Header

The Routing header is used by an IPv6 source to list one or more intermediate nodes to be "visited" on the way to a packet's destination. This function is very similar to IPv4's Loose Source

and Record Route option. The Routing header is identified by a Next Header value of 43 in the immediately preceding header, and has the following format:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
| Next Header | Hdr Ext Len | Routing Type | Segments Left |
+-----+-----+-----+-----+-----+-----+-----+-----+
|
|
.                               .
.                               .
|                               |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Next Header	8-bit selector. Identifies the type of header immediately following the Routing header. Uses the same values as the IPv4 Protocol field [IANA-PN].
Hdr Ext Len	8-bit unsigned integer. Length of the Routing header in 8-octet units, not including the first 8 octets.
Routing Type	8-bit identifier of a particular Routing header variant.
Segments Left	8-bit unsigned integer. Number of route segments remaining, i.e., number of explicitly listed intermediate nodes still to be visited before reaching the final destination.
type-specific data	Variable-length field, of format determined by the Routing Type, and of length such that the complete Routing header is an integer multiple of 8 octets long.

If, while processing a received packet, a node encounters a Routing header with an unrecognized Routing Type value, the required behavior of the node depends on the value of the Segments Left field, 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.

If, after processing a Routing header of a received packet, an intermediate node determines that the packet is to be forwarded onto a link whose link MTU is less than the size of the packet, the node must discard the packet and send an ICMP Packet Too Big message to the packet's Source Address.

The currently defined IPv6 Routing Headers and their status can be found at [IANA-RH]. Allocation guidelines for IPv6 Routing Headers can be found in [RFC5871].

#### 4.5. Fragment Header

The Fragment header is used by an IPv6 source to send a packet larger than would fit in the path MTU to its destination. (Note: unlike IPv4, fragmentation in IPv6 is performed only by source nodes, not by routers along a packet's delivery path -- see section 5.) The Fragment header is identified by a Next Header value of 44 in the immediately preceding header, and has the following format:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
| Next Header | Reserved | Fragment Offset | Res|M|
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Identification                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Next Header	8-bit selector. Identifies the initial header type of the Fragmentable Part of the original packet (defined below). Uses the same values as the IPv4 Protocol field [IANA-PN].
Reserved	8-bit reserved field. Initialized to zero for transmission; ignored on reception.
Fragment Offset	13-bit unsigned integer. The offset, in 8-octet units, of the data following this header, relative to the start of the Fragmentable Part of the original packet.
Res	2-bit reserved field. Initialized to zero for transmission; ignored on reception.
M flag	1 = more fragments; 0 = last fragment.

Identification            32 bits.   See description below.

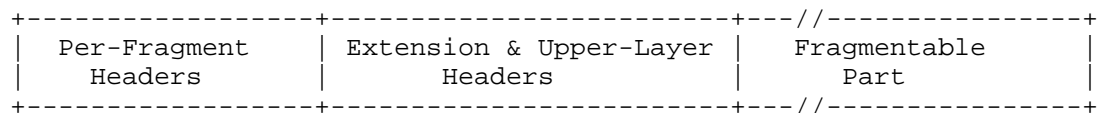
In order to send a packet that is too large to fit in the MTU of the path to its destination, a source node may divide the packet into fragments and send each fragment as a separate packet, to be reassembled at the receiver.

For every packet that is to be fragmented, the source node generates an Identification value. The Identification must be different than that of any other fragmented packet sent recently\* with the same Source Address and Destination Address. If a Routing header is present, the Destination Address of concern is that of the final destination.

- \* "recently" means within the maximum likely lifetime of a packet, including transit time from source to destination and time spent awaiting reassembly with other fragments of the same packet. However, it is not required that a source node knows the maximum packet lifetime. Rather, it is assumed that the requirement can be met by implementing an algorithm that results in a low identification reuse frequency. Examples of algorithms that can meet this requirement are described in [RFC7739].

The initial, large, unfragmented packet is referred to as the "original packet", and it is considered to consist of three parts, as illustrated:

original packet:



The Per-Fragment Headers must consist of the IPv6 header plus any extension headers that must be processed by nodes en route to the destination, that is, all headers up to and including the Routing header if present, else the Hop-by-Hop Options header if present, else no extension headers.

The Extension Headers are all other extension headers that are not included in the Per-Fragment headers part of the packet. For this purpose, the Encapsulating Security Payload (ESP) is not considered an extension header. The Upper-Layer Header is the first upper-layer header that is not an IPv6 extension header.



Examples of upper-layer headers include TCP, UDP, IPv4, IPv6, ICMPv6, and as noted ESP.

The Fragmentable Part consists of the rest of the packet after the upper-layer header or after any header (i.e., initial IPv6 header or extension header) that contains a Next Header value of No Next Header.

The Fragmentable Part of the original packet is divided into fragments. The lengths of the fragments must be chosen such that the resulting fragment packets fit within the MTU of the path to the packets' destination(s). Each complete fragment, except possibly the last ("rightmost") one, being an integer multiple of 8 octets long.

The fragments are transmitted in separate "fragment packets" as illustrated:

original packet:

Per-Fragment Headers	Ext & Upper-Layer Headers	first fragment	second fragment	...	last fragment
-------------------------	------------------------------	-------------------	--------------------	-----	------------------

fragment packets:

Per-Fragment Headers	Fragment Header	Ext & Upper-Layer Headers	first fragment
-------------------------	--------------------	------------------------------	-------------------

Per-Fragment Headers	Fragment Header	second fragment
-------------------------	--------------------	--------------------

o  
o  
o

Per-Fragment Headers	Fragment Header	last fragment
-------------------------	--------------------	------------------

The first fragment packet is composed of:

- (1) The Per-Fragment Headers of the original packet, with the Payload Length of the original IPv6 header changed to contain the length of this fragment packet only (excluding the length of the

IPv6 header itself), and the Next Header field of the last header of the Per-Fragment Headers changed to 44.

(2) A Fragment header containing:

The Next Header value that identifies the first header after the Per-Fragment Headers of the original packet.

A Fragment Offset containing the offset of the fragment, in 8-octet units, relative to the start of the Fragmentable Part of the original packet. The Fragment Offset of the first ("leftmost") fragment is 0.

An M flag value of 1 as this is the first fragment.

The Identification value generated for the original packet.

(3) Extension Headers, if any, and the Upper-Layer header. These headers must be in the first fragment. Note: This restricts the size of the headers through the Upper-Layer header to the MTU of the path to the packets' destinations(s).

(4) The first fragment.

The subsequent fragment packets are composed of:

(1) The Per-Fragment Headers of the original packet, with the Payload Length of the original IPv6 header changed to contain the length of this fragment packet only (excluding the length of the IPv6 header itself), and the Next Header field of the last header of the Per-Fragment Headers changed to 44.

(2) A Fragment header containing:

The Next Header value that identifies the first header after the Per-Fragment Headers of the original packet.

A Fragment Offset containing the offset of the fragment, in 8-octet units, relative to the start of the Fragmentable part of the original packet.

An M flag value of 0 if the fragment is the last ("rightmost") one, else an M flag value of 1.

The Identification value generated for the original packet.

(3) The fragment itself.

Fragments must not be created that overlap with any other fragments created from the original packet.

At the destination, fragment packets are reassembled into their original, unfragmented form, as illustrated:

reassembled original packet:

```

+-----+-----+-----+-----+//--+-----+
| Per-Fragment | Ext & Upper-Layer | first | second |      | last |
|   Headers    |   Headers      | frag data | fragment | .... | fragment |
+-----+-----+-----+-----+//--+-----+

```

The following rules govern reassembly:

An original packet is reassembled only from fragment packets that have the same Source Address, Destination Address, and Fragment Identification.

The Per-Fragment Headers of the reassembled packet consists of all headers up to, but not including, the Fragment header of the first fragment packet (that is, the packet whose Fragment Offset is zero), with the following two changes:

The Next Header field of the last header of the Per-Fragment Headers is obtained from the Next Header field of the first fragment's Fragment header.

The Payload Length of the reassembled packet is computed from the length of the Per-Fragment Headers and the length and offset of the last fragment. For example, a formula for computing the Payload Length of the reassembled original packet is:

$$PL.orig = PL.first - FL.first - 8 + (8 * FO.last) + FL.last$$

where

PL.orig = Payload Length field of reassembled packet.

PL.first = Payload Length field of first fragment packet.

FL.first = length of fragment following Fragment header of first fragment packet.  
FO.last = Fragment Offset field of Fragment header of last fragment packet.  
FL.last = length of fragment following Fragment header of last fragment packet.

The Fragmentable Part of the reassembled packet is constructed from the fragments following the Fragment headers in each of the fragment packets. The length of each fragment is computed by subtracting from the packet's Payload Length the length of the headers between the IPv6 header and fragment itself; its relative position in Fragmentable Part is computed from its Fragment Offset value.

The Fragment header is not present in the final, reassembled packet.

If the fragment is a whole datagram (that is, both the Fragment Offset field and the M flag are zero), then it does not need any further reassembly and should be processed as a fully reassembled packet (i.e., updating Next Header, adjust Payload Length, removing the Fragmentation Header, etc.). Any other fragments that match this packet (i.e., the same IPv6 Source Address, IPv6 Destination Address, and Fragment Identification) should be processed independently.

The following error conditions may arise when reassembling fragmented packets:

- o If insufficient fragments are received to complete reassembly of a packet within 60 seconds of the reception of the first-arriving fragment of that packet, reassembly of that packet must be abandoned and all the fragments that have been received for that packet must be discarded. If the first fragment (i.e., the one with a Fragment Offset of zero) has been received, an ICMP Time Exceeded -- Fragment Reassembly Time Exceeded message should be sent to the source of that fragment.
- o If the length of a fragment, as derived from the fragment packet's Payload Length field, is not a multiple of 8 octets and the M flag of that fragment is 1, then that fragment must be discarded and an ICMP Parameter Problem, Code 0, message should be sent to the source of the fragment, pointing to the Payload Length field of the fragment packet.

- o If the length and offset of a fragment are such that the Payload Length of the packet reassembled from that fragment would exceed 65,535 octets, then that fragment must be discarded and an ICMP Parameter Problem, Code 0, message should be sent to the source of the fragment, pointing to the Fragment Offset field of the fragment packet.
- o If the first fragment does not include all headers through an Upper-Layer header, then that fragment should be discarded and an ICMP Parameter Problem, Code 3, message should be sent to the source of the fragment, with the Pointer field set to zero.
- o If any of the fragments being reassembled overlaps with any other fragments being reassembled for the same packet, reassembly of that packet must be abandoned and all the fragments that have been received for that packet must be discarded and no ICMP error messages should be sent.

It should be noted that fragments may be duplicated in the network. Instead of treating these exact duplicate fragments as overlapping fragments, an implementation may choose to detect this case and drop exact duplicate fragments while keeping the other fragments belonging to the same packet.

The following conditions are not expected to occur frequently, but are not considered errors if they do:

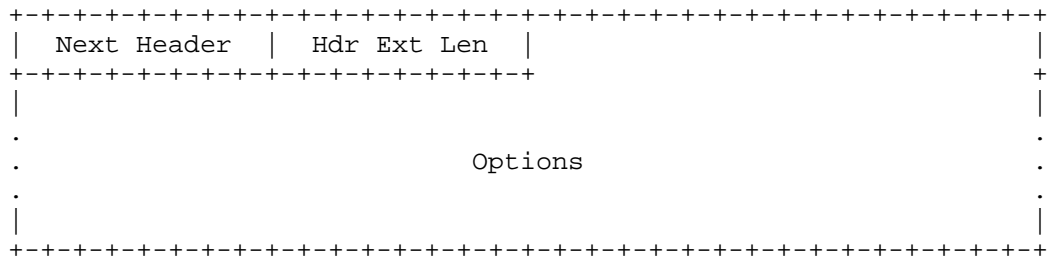
The number and content of the headers preceding the Fragment header of different fragments of the same original packet may differ. Whatever headers are present, preceding the Fragment header in each fragment packet, are processed when the packets arrive, prior to queueing the fragments for reassembly. Only those headers in the Offset zero fragment packet are retained in the reassembled packet.

The Next Header values in the Fragment headers of different fragments of the same original packet may differ. Only the value from the Offset zero fragment packet is used for reassembly.

Other fields in the IPv6 header may also vary across the fragments being reassembled. Specifications that use these fields may provide additional instructions if the basic mechanism of using the values from the Offset zero fragment is not sufficient. For example, Section 5.3 of [RFC3168] describes how to combine the Explicit Congestion Notification (ECN) bits from different fragments to derive the ECN bits of the reassembled packet.

#### 4.6. Destination Options Header

The Destination Options header is used to carry optional information that need be examined only by a packet's destination node(s). The Destination Options header is identified by a Next Header value of 60 in the immediately preceding header, and has the following format:



Next Header	8-bit selector. Identifies the type of header immediately following the Destination Options header. Uses the same values as the IPv4 Protocol field [IANA-PN].
Hdr Ext Len	8-bit unsigned integer. Length of the Destination Options header in 8-octet units, not including the first 8 octets.
Options	Variable-length field, of length such that the complete Destination Options header is an integer multiple of 8 octets long. Contains one or more TLV-encoded options, as described in section 4.2.

The only destination options defined in this document are the Pad1 and PadN options specified in section 4.2.

Note that there are two possible ways to encode optional destination information in an IPv6 packet: either as an option in the Destination Options header, or as a separate extension header. The Fragment header and the Authentication header are examples of the latter approach. Which approach can be used depends on what action is desired of a destination node that does not understand the optional information:

- o If the desired action is for the destination node to discard the packet and, only if the packet's Destination Address is not a multicast address, send an ICMP Unrecognized Type message to the packet's Source Address, then the information may be encoded either as a separate header or as an option in the Destination Options header whose Option Type has the value 11 in its highest-order two bits. The choice may depend on such factors as which takes fewer octets, or which yields better alignment or more efficient parsing.
- o If any other action is desired, the information must be encoded as an option in the Destination Options header whose Option Type has the value 00, 01, or 10 in its highest-order two bits, specifying the desired action (see section 4.2).

#### 4.7. No Next Header

The value 59 in the Next Header field of an IPv6 header or any extension header indicates that there is nothing following that header. If the Payload Length field of the IPv6 header indicates the presence of octets past the end of a header whose Next Header field contains 59, those octets must be ignored, and passed on unchanged if the packet is forwarded.

#### 4.8. Defining New Extension Headers and Options

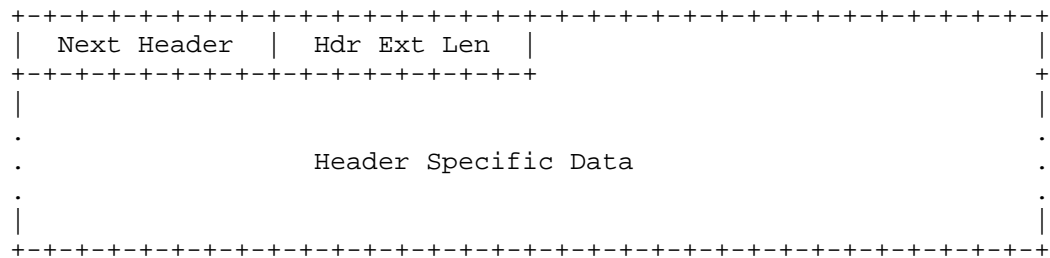
Defining new IPv6 extension headers is not recommended, unless there are no existing IPv6 extension headers that can be used by specifying a new option for that IPv6 extension header. A proposal to specify a new IPv6 extension header must include a detailed technical explanation of why an existing IPv6 extension header can not be used for the desired new function. See [RFC6564] for additional background information.

Note: New extension headers that require hop-by-hop behavior must not be defined because, as specified in Section 4 of this document, the only Extension Header that has hop-by-hop behavior is the Hop-by-Hop Options header.

New hop-by-hop options are not recommended because nodes may be configured to ignore the Hop-by-Hop Option header, drop packets containing a hop-by-hop header, or assign packets containing a hop-by-hop header to a slow processing path. Designers considering defining new hop-by-hop options need to be aware of this likely behaviour. There has to be a very clear justification why any new hop-by-hop option is needed before it is standardized.

Instead of defining new Extension Headers, it is recommended that the Destination Options header is used to carry optional information that must be examined only by a packet's destination node(s), because they provide better handling and backward compatibility.

If new Extension Headers are defined, they need to use the following format:



Next Header	8-bit selector. Identifies the type of header immediately following the extension header. Uses the same values as the IPv4 Protocol field [IANA-PN].
Hdr Ext Len	8-bit unsigned integer. Length of the Destination Options header in 8-octet units, not including the first 8 octets.
Header Specific Data	Variable-length field. Fields specific to the extension header.

## 5. Packet Size Issues

IPv6 requires that every link in the internet have an MTU of 1280 octets or greater. This is known as the IPv6 minimum link MTU. On any link that cannot convey a 1280-octet packet in one piece, link-specific fragmentation and reassembly must be provided at a layer below IPv6.

Links that have a configurable MTU (for example, PPP links [RFC1661]) must be configured to have an MTU of at least 1280 octets; it is recommended that they be configured with an MTU of 1500 octets or greater, to accommodate possible encapsulations (i.e., tunneling) without incurring IPv6-layer fragmentation.

From each link to which a node is directly attached, the node must be able to accept packets as large as that link's MTU.



It is strongly recommended that IPv6 nodes implement Path MTU Discovery [RFC1981], in order to discover and take advantage of path MTUs greater than 1280 octets. However, a minimal IPv6 implementation (e.g., in a boot ROM) may simply restrict itself to sending packets no larger than 1280 octets, and omit implementation of Path MTU Discovery.

In order to send a packet larger than a path's MTU, a node may use the IPv6 Fragment header to fragment the packet at the source and have it reassembled at the destination(s). However, the use of such fragmentation is discouraged in any application that is able to adjust its packets to fit the measured path MTU (i.e., down to 1280 octets).

A node must be able to accept a fragmented packet that, after reassembly, is as large as 1500 octets. A node is permitted to accept fragmented packets that reassemble to more than 1500 octets. An upper-layer protocol or application that depends on IPv6 fragmentation to send packets larger than the MTU of a path should not send packets larger than 1500 octets unless it has assurance that the destination is capable of reassembling packets of that larger size.

## 6. Flow Labels

The 20-bit Flow Label field in the IPv6 header is used by a source to label sequences of packets to be treated in the network as a single flow.

The current definition of the IPv6 Flow Label can be found in [RFC6437].

## 7. Traffic Classes

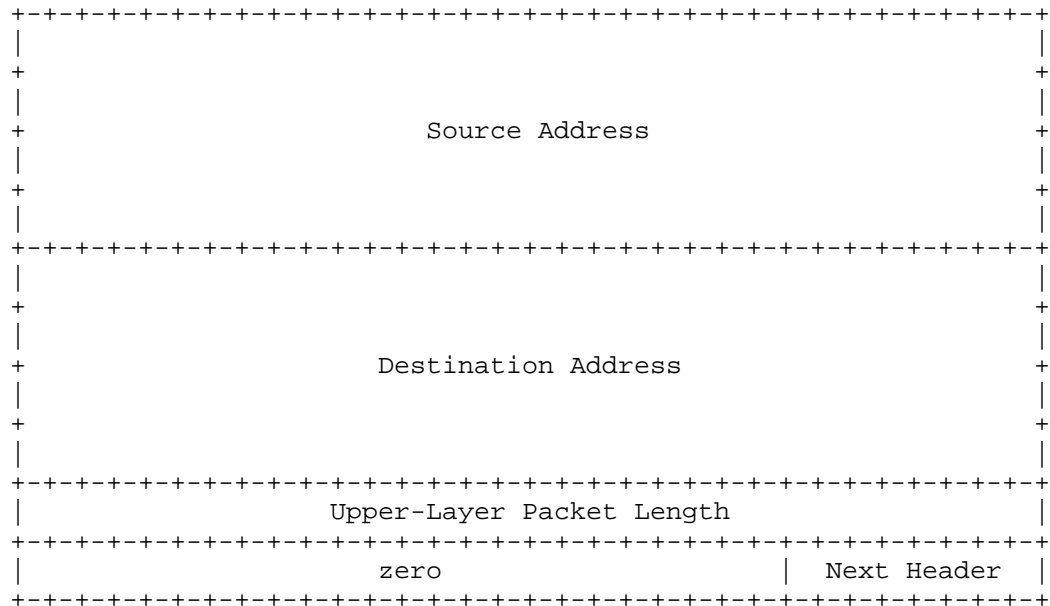
The 8-bit Traffic Class field in the IPv6 header is used by the network for traffic management. The value of the Traffic Class bits in a received packet or fragment might be different from the value sent by the packet's source.

The current use of the Traffic Class field for Differentiated Services and Explicit Congestion Notification is specified in [RFC2474] and [RFC3168].

## 8. Upper-Layer Protocol Issues

### 8.1. Upper-Layer Checksums

Any transport or other upper-layer protocol that includes the addresses from the IP header in its checksum computation must be modified for use over IPv6, to include the 128-bit IPv6 addresses instead of 32-bit IPv4 addresses. In particular, the following illustration shows the TCP and UDP "pseudo-header" for IPv6:



- o If the IPv6 packet contains a Routing header, the Destination Address used in the pseudo-header is that of the final destination. At the originating node, that address will be in the last element of the Routing header; at the recipient(s), that address will be in the Destination Address field of the IPv6 header.
- o The Next Header value in the pseudo-header identifies the upper-layer protocol (e.g., 6 for TCP, or 17 for UDP). It will differ from the Next Header value in the IPv6 header if there are extension headers between the IPv6 header and the upper-layer header.
- o The Upper-Layer Packet Length in the pseudo-header is the length of the upper-layer header and data (e.g., TCP header plus TCP data). Some upper-layer protocols carry their own

length information (e.g., the Length field in the UDP header); for such protocols, that is the length used in the pseudo-header. Other protocols (such as TCP) do not carry their own length information, in which case the length used in the pseudo-header is the Payload Length from the IPv6 header, minus the length of any extension headers present between the IPv6 header and the upper-layer header.

- o Unlike IPv4, the default behavior when UDP packets are originated by an IPv6 node is that the UDP checksum is not optional. That is, whenever originating a UDP packet, an IPv6 node must compute a UDP checksum over the packet and the pseudo-header, and, if that computation yields a result of zero, it must be changed to hex FFFF for placement in the UDP header. IPv6 receivers must discard UDP packets containing a zero checksum, and should log the error.
- o As an exception to the default behaviour, protocols that use UDP as a tunnel encapsulation may enable zero-checksum mode for a specific port (or set of ports) for sending and/or receiving. Any node implementing zero-checksum mode must follow the requirements specified in "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums" [RFC6936].

The IPv6 version of ICMP [RFC4443] includes the above pseudo-header in its checksum computation; this is a change from the IPv4 version of ICMP, which does not include a pseudo-header in its checksum. The reason for the change is to protect ICMP from misdelivery or corruption of those fields of the IPv6 header on which it depends, which, unlike IPv4, are not covered by an internet-layer checksum. The Next Header field in the pseudo-header for ICMP contains the value 58, which identifies the IPv6 version of ICMP.

## 8.2. Maximum Packet Lifetime

Unlike IPv4, IPv6 nodes are not required to enforce maximum packet lifetime. That is the reason the IPv4 "Time to Live" field was renamed "Hop Limit" in IPv6. In practice, very few, if any, IPv4 implementations conform to the requirement that they limit packet lifetime, so this is not a change in practice. Any upper-layer protocol that relies on the internet layer (whether IPv4 or IPv6) to limit packet lifetime ought to be upgraded to provide its own mechanisms for detecting and discarding obsolete packets.

### 8.3. Maximum Upper-Layer Payload Size

When computing the maximum payload size available for upper-layer data, an upper-layer protocol must take into account the larger size of the IPv6 header relative to the IPv4 header. For example, in IPv4, TCP's MSS option is computed as the maximum packet size (a default value or a value learned through Path MTU Discovery) minus 40 octets (20 octets for the minimum-length IPv4 header and 20 octets for the minimum-length TCP header). When using TCP over IPv6, the MSS must be computed as the maximum packet size minus 60 octets, because the minimum-length IPv6 header (i.e., an IPv6 header with no extension headers) is 20 octets longer than a minimum-length IPv4 header.

### 8.4. Responding to Packets Carrying Routing Headers

When an upper-layer protocol sends one or more packets in response to a received packet that included a Routing header, the response packet(s) must not include a Routing header that was automatically derived by "reversing" the received Routing header UNLESS the integrity and authenticity of the received Source Address and Routing header have been verified (e.g., via the use of an Authentication header in the received packet). In other words, only the following kinds of packets are permitted in response to a received packet bearing a Routing header:

- o Response packets that do not carry Routing headers.
- o Response packets that carry Routing headers that were NOT derived by reversing the Routing header of the received packet (for example, a Routing header supplied by local configuration).
- o Response packets that carry Routing headers that were derived by reversing the Routing header of the received packet IF AND ONLY IF the integrity and authenticity of the Source Address and Routing header from the received packet have been verified by the responder.

## 9. IANA Considerations

RFC2460 is referenced in a number of IANA registries. These include:

- o Internet Protocol Version 6 (IPv6) Parameters [IANA-6P]

- o Assigned Internet Protocol Numbers [IANA-PN]
- o ONC RPC Network Identifiers (netids) [IANA-NI]
- o Technical requirements for authoritative name servers [IANA-NS]
- o Network Layer Protocol Identifiers (NLPIDs) of Interest [IANA-NL]
- o Protocol Registries [IANA-PR]
- o Structure of Management Information (SMI) Numbers (MIB Module Registrations) [IANA-MI]

The IANA should update these references to point to this document.

## 10. Security Considerations

IPv6, from the viewpoint of the basic format and transmission of packets, has security properties that are similar to IPv4. These security issues include:

- o Eavesdropping, On-path elements can observe the whole packet (including both contents and metadata) of each IPv6 datagram.
- o Replay, where attacker records a sequence of packets off of the wire and plays them back to the party which originally received them.
- o Packet insertion, where the attacker forges a packet with some chosen set of properties and injects it into the network.
- o Packet deletion, where the attacker remove a packet from the wire.
- o Packet modification, where the attacker removes a packet from the wire, modifies it, and re-injects it into the network.
- o Man in the Middle attacks, where the attacker subverts the communication stream in order to pose as the sender to receiver and the receiver to the sender.
- o Denial of Service Attacks, where the attacker sends large amounts of legitimate traffic to a destination to overwhelm it.

IPv6 packets can be protected from eavesdropping, replay, packet insertion, packet modification, and man in the middle attacks by use of the "Security Architecture for the Internet Protocol" [RFC4301]. In addition, upper-layer protocols such as TLS or SSH can be used to protect the application layer traffic running on top of IPv6.

There is not any mechanism to protect against "denial of service attacks". Defending against these type of attacks is outside the scope of this specification.

IPv6 addresses are significantly larger than IPv4 address making it much harder to scan the address space across the Internet and even on a single network link (e.g., Local Area Network). See [RFC7707] for more information.

IPv6 addresses of nodes are expected to be more visible on the Internet as compared with IPv4 since the use of address translation technology is reduced. This creates some additional privacy issues such as making it easier to distinguish endpoints. See [RFC7721] for more information.

The design of IPv6 extension headers architecture, while adding a lot of flexibility, also creates new security challenges. As noted below, issues relating the fragment extension header have been resolved, but it's clear that for any new extension header designed in the future, the security implications need to be examined thoroughly, and this needs to include how the new extension header works with existing extension headers. See [RFC7045] for more information.

This version of the IPv6 specification resolves a number of security issues that were found with the previous version [RFC2460] of the IPv6 specification. These include:

- o Revised the text to handle the case of fragments that are whole datagrams (i.e., both the Fragment Offset field and the M flag are zero). If received they should be processed as a reassembled packet. Any other fragments that match should be processed independently. The Fragment creation process was modified to not create whole datagram fragments (Fragment Offset field and the M flag are zero). See [RFC6946] and [RFC8021] for more information.
- o Changed the text to require that IPv6 nodes must not create overlapping fragments. Also, when reassembling an IPv6 datagram, if one or more its constituent fragments is determined to be an overlapping fragment, the entire datagram (and any constituent fragments) must be silently discarded. Includes clarification that no ICMP error message should be sent if overlapping fragments are received. See [RFC5722] for more information.

- 0 Revised the text to require that all headers through the first Upper-Layer Header are in the first fragment. See [RFC6946] for more information.
- o Removed the paragraph in Section 5 that required including a fragment header to outgoing packets if a ICMP Packet Too Big message reporting a Next-Hop MTU less than 1280. See [RFC7112] for more information.
- o Incorporated the updates from [RFC5095] and [RFC5871] to remove the description of the RH0 Routing Header, that the allocations guidelines for routing headers are specified in RFC5871, and removed RH0 Routing Header from the list of required extension headers.

Security issues relating to other parts of IPv6 including addressing, ICMPv6, Path MTU Discovery, etc., are discussed in the appropriate specifications.

## 11. Acknowledgments

The authors gratefully acknowledge the many helpful suggestions of the members of the IPng working group, the End-to-End Protocols research group, and the Internet Community At Large.

The authors would also like to acknowledge the authors of the updating RFCs that were incorporated in this version of the document to move the IPv6 specification to Internet Standard. They are Joe Abley, Shane Amante, Jari Arkko, Manav Bhatia, Ronald P. Bonica, Scott Bradner, Brian Carpenter, P.F. Chimento, Marshall Eubanks, Fernando Gont, James Hoagland, Sheng Jiang, Erik Kline, Suresh Krishnan, Vishwas Manral, George Neville-Neil, Jarno Rajahalme, Pekka Savola, Magnus Westerlund, and James Woodyatt.

## 12. References

### 12.1. Normative References

- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<http://www.rfc-editor.org/info/rfc791>>.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <<http://www.rfc-editor.org/info/rfc2474>>.

- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<http://www.rfc-editor.org/info/rfc3168>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<http://www.rfc-editor.org/info/rfc4291>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, DOI 10.17487/RFC4443, March 2006, <<http://www.rfc-editor.org/info/rfc4443>>.
- [RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <<http://www.rfc-editor.org/info/rfc6437>>.

## 12.2. Informative References

- [IANA-6P] "Internet Protocol Version 6 (IPv6) Parameters", <<https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml>>.
- [IANA-EH] "IPv6 Extension Header Types", <<https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#extension-header>>.
- [IANA-MI] "Structure of Management Information (SMI) Numbers (MIB Module Registrations)", <<http://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml>>.
- [IANA-NI] "ONC RPC Network Identifiers (netids)", <<http://www.iana.org/assignments/rpc-netids/rpc-netids.xhtml>>.
- [IANA-NL] "Network Layer Protocol Identifiers (NLPIDs) of Interest", <<http://www.iana.org/assignments/nlpids/nlpids.xhtml>>.
- [IANA-NS] "Technical requirements for authoritative name servers", <<https://www.iana.org/help/nameserver-requirements>>.
- [IANA-PN] "Assigned Internet Protocol Numbers", <<https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>>.



- [IANA-PR] "Protocol Registries", <<https://www.iana.org/protocols>>.
- [IANA-RH] "IANA Routing Types Parameter Registry",  
<<https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml#ipv6-parameters-3>>.
- [RFC1661] Simpson, W., Ed., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, DOI 10.17487/RFC1661, July 1994,  
<<http://www.rfc-editor.org/info/rfc1661>>.
- [RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August 1996, <<http://www.rfc-editor.org/info/rfc1981>>.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<http://www.rfc-editor.org/info/rfc2460>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<http://www.rfc-editor.org/info/rfc4301>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<http://www.rfc-editor.org/info/rfc4302>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<http://www.rfc-editor.org/info/rfc4303>>.
- [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, <<http://www.rfc-editor.org/info/rfc5095>>.
- [RFC5722] Krishnan, S., "Handling of Overlapping IPv6 Fragments", RFC 5722, DOI 10.17487/RFC5722, December 2009, <<http://www.rfc-editor.org/info/rfc5722>>.
- [RFC5871] Arkko, J. and S. Bradner, "IANA Allocation Guidelines for the IPv6 Routing Header", RFC 5871, DOI 10.17487/RFC5871, May 2010, <<http://www.rfc-editor.org/info/rfc5871>>.
- [RFC6564] Krishnan, S., Woodyatt, J., Kline, E., Hoagland, J., and M. Bhatia, "A Uniform Format for IPv6 Extension Headers", RFC 6564, DOI 10.17487/RFC6564, April 2012, <<http://www.rfc-editor.org/info/rfc6564>>.

- [RFC6936] Fairhurst, G. and M. Westerlund, "Applicability Statement for the Use of IPv6 UDP Datagrams with Zero Checksums", RFC 6936, DOI 10.17487/RFC6936, April 2013, <<http://www.rfc-editor.org/info/rfc6936>>.
- [RFC6946] Gont, F., "Processing of IPv6 "Atomic" Fragments", RFC 6946, DOI 10.17487/RFC6946, May 2013, <<http://www.rfc-editor.org/info/rfc6946>>.
- [RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, DOI 10.17487/RFC7045, December 2013, <<http://www.rfc-editor.org/info/rfc7045>>.
- [RFC7112] Gont, F., Manral, V., and R. Bonica, "Implications of Oversized IPv6 Header Chains", RFC 7112, DOI 10.17487/RFC7112, January 2014, <<http://www.rfc-editor.org/info/rfc7112>>.
- [RFC7707] Gont, F. and T. Chown, "Network Reconnaissance in IPv6 Networks", RFC 7707, DOI 10.17487/RFC7707, March 2016, <<http://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, <<http://www.rfc-editor.org/info/rfc7721>>.
- [RFC7739] Gont, F., "Security Implications of Predictable Fragment Identification Values", RFC 7739, DOI 10.17487/RFC7739, February 2016, <<http://www.rfc-editor.org/info/rfc7739>>.
- [RFC8021] Gont, F., Liu, W., and T. Anderson, "Generation of IPv6 Atomic Fragments Considered Harmful", RFC 8021, DOI 10.17487/RFC8021, January 2017, <<http://www.rfc-editor.org/info/rfc8021>>.

#### Appendix A. Formatting Guidelines for Options

This appendix gives some advice on how to lay out the fields when designing new options to be used in the Hop-by-Hop Options header or the Destination Options header, as described in section 4.2. These guidelines are based on the following assumptions:

- o One desirable feature is that any multi-octet fields within the Option Data area of an option be aligned on their natural

boundaries, i.e., fields of width  $n$  octets should be placed at an integer multiple of  $n$  octets from the start of the Hop-by-Hop or Destination Options header, for  $n = 1, 2, 4$ , or  $8$ .

- o Another desirable feature is that the Hop-by-Hop or Destination Options header take up as little space as possible, subject to the requirement that the header be an integer multiple of 8 octets long.
- o It may be assumed that, when either of the option-bearing headers are present, they carry a very small number of options, usually only one.

These assumptions suggest the following approach to laying out the fields of an option: order the fields from smallest to largest, with no interior padding, then derive the alignment requirement for the entire option based on the alignment requirement of the largest field (up to a maximum alignment of 8 octets). This approach is illustrated in the following examples:

#### Example 1

If an option X required two data fields, one of length 8 octets and one of length 4 octets, it would be laid out as follows:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     | Option Type=X |Opt Data Len=12|
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |
|                                     | 4-octet field |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |
|                                     | 8-octet field |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Its alignment requirement is  $8n+2$ , to ensure that the 8-octet field starts at a multiple-of-8 offset from the start of the enclosing header. A complete Hop-by-Hop or Destination Options header containing this one option would look as follows:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
| Next Header | Hdr Ext Len=1 | Option Type=X | Opt Data Len=12|
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     4-octet field                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     8-octet field                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

#### Example 2

If an option Y required three data fields, one of length 4 octets, one of length 2 octets, and one of length 1 octet, it would be laid out as follows:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Option Type=Y                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Opt Data Len=7 | 1-octet field |           2-octet field           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     4-octet field                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Its alignment requirement is  $4n+3$ , to ensure that the 4-octet field starts at a multiple-of-4 offset from the start of the enclosing header. A complete Hop-by-Hop or Destination Options header containing this one option would look as follows:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
| Next Header | Hdr Ext Len=1 | Pad1 Option=0 | Option Type=Y |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Opt Data Len=7 | 1-octet field |           2-octet field           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     4-octet field                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
| PadN Option=1 | Opt Data Len=2 |           0           |           0           |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

#### Example 3

A Hop-by-Hop or Destination Options header containing both options X and Y from Examples 1 and 2 would have one of the two following formats, depending on which option appeared first:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
| Next Header | Hdr Ext Len=3 | Option Type=X | Opt Data Len=12 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     4-octet field                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     8-octet field                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
| PadN Option=1 | Opt Data Len=1 |          0          | Option Type=Y |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Opt Data Len=7 | 1-octet field |          2-octet field          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     4-octet field                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
| PadN Option=1 | Opt Data Len=2 |          0          |          0          |
+-----+-----+-----+-----+-----+-----+-----+-----+

+-----+-----+-----+-----+-----+-----+-----+-----+
| Next Header | Hdr Ext Len=3 | Pad1 Option=0 | Option Type=Y |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Opt Data Len=7 | 1-octet field |          2-octet field          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     4-octet field                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
| PadN Option=1 | Opt Data Len=4 |          0          |          0          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          0          |          0          | Option Type=X | Opt Data Len=12 |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     4-octet field                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     8-octet field                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

## Appendix B. Changes Since RFC2460

This memo has the following changes from RFC2460.

- o Removed IP Next Generation from the Abstract.
- o Added text in Section 1 that the Data Transmission Order is the same as IPv4 as defined in RFC791.
- o Clarified the text in Section 3 about decrementing the hop limit.

- o Clarification that extension headers (except for the hop-by-hop options header) are not processed, inserted, or deleted by any node along a packet's delivery path.
- o Changed requirement for the Hop-by-Hop Options header to a may, and added a note to indicate what is expected regarding the Hop-by-Hop Options header.
- o Added paragraph to Section 4 to clarify how Extension Headers are numbered and which are upper-layer headers.
- o Add reference to the end of Section 4 to IPv6 Extension Header IANA registry.
- o Incorporate the updates from RFC5095 and RFC5871 to remove the description of the RH0 Routing Header, that the allocations guidelines for routing headers are specified in RFC5871, and removed RH0 Routing Header from the list of required extension headers.
- o Revised Section 4.5 on IPv6 Fragmentation based on updates from RFC5722, RFC6946 RFC7112, and RFC8021. This include:
  - Revised the text to handle the case of fragments that are whole datagrams (i.e., both the Fragment Offset field and the M flag are zero). If received they should be processed as a reassembled packet. Any other fragments that match should be processed independently. The revised Fragment creation process was modified to not create whole datagram fragments (Fragment Offset field and the M flag are zero).
  - Changed the text to require that IPv6 nodes must not create overlapping fragments. Also, when reassembling an IPv6 datagram, if one or more its constituent fragments is determined to be an overlapping fragment, the entire datagram (and any constituent fragments) must be silently discarded. Includes a clarification that no ICMP error message should be sent if overlapping fragments are received.
  - Revised the text to require that all headers through the first Upper-Layer Header are in the first fragment. This changed the text describing how packets are fragmented and reassembled, and added a new error case.
  - Added text to Fragment Header process on handling exact duplicate fragments.

- Updated the Fragmentation header text to correct the inclusion of AH and note no next header case.
  - Change terminology in Fragment header section from "Unfragmentable Headers" to "Per-Fragment Headers".
  - Removed the paragraph in Section 5 that required including a fragment header to outgoing packets if a ICMP Packet Too Big message reporting a Next-Hop MTU less than 1280.
  - Changed the text to clarify MTU restriction and 8-byte restrictions, and noting the restriction on headers in first fragment.
- o In Section 4.5 added clarification noting that some fields in the IPv6 header may also vary across the fragments being reassembled and that other specifications may provide additional instructions for how they should be reassembled. For example, Section 5.3 of [RFC3168].
  - o Incorporated the update from RFC6564 to add a new Section 4.8 that describes recommendations for defining new Extension headers and options.
  - o Added text to Section 5 to define "IPv6 minimum link MTU".
  - o Simplify the text in Section 6 about Flow Labels and remove Appendix A, and instead point to the current specifications of the IPv6 Flow Label field as defined in [RFC6437] and the Traffic Class as defined in [RFC2474] and [RFC3168].
  - o Incorporate the update in made by RFC6935 "UDP Checksums for Tunneled Packets" in Section 8. Added an exception to the default behaviour for the handling of handling UDP packets with zero checksums for tunnels.
  - o Add instruction to Section 9 "IANA Considerations" to change references to RFC2460 to this document
  - o Revised and expanded Section 10 "Security Considerations".
  - o Add a paragraph to the acknowledgement section acknowledging the authors of the updating documents
  - o Update references to current versions and assign references to normative and informative.
  - o Changes to resolve the open Errata on RFC2460. These are:

Errata ID: 2541: This errata notes that RFC2460 didn't update RFC2205 when the length of the Flow Label was changed from 24 to 20 bits from RFC1883. This issue was resolved in RFC6437 where the Flow Label is defined. This draft now references RFC6437. No change is required.

Errata ID: 4279: This errata noted that the specification doesn't handle the case of a forwarding node receiving a packet with a zero Hop Limit. This is fixed in Section 3 of this draft.

Errata ID: 2843: This errata is marked rejected. No change was made.

#### B.1. Change History Since RFC2460

NOTE TO RFC EDITOR: Please remove this subsection prior to RFC Publication

This section describes change history made in each Internet Draft that went into producing this version. The numbers identify the Internet-Draft version in which the change was made.

Working Group Internet Drafts

- 13) Added link to reference to RFC6564 in Section 4.8.
- 13) Added text to Section 5 to define "IPv6 minimum link MTU".
- 13) Editorial changes.
- 12) Editorial changes (remove old duplicate paragraph).
- 11) In Section 4.5 added clarification noting that some fields in the IPv6 header may also vary across the fragments being reassembled and that other specifications may provide additional instructions for how they should be reassembled. For example, Section 5.3 of [RFC3168].
- 11) In Section 4 restructured text including separated behaviors of extension headers and the hop-by-hop option header, removed "examine" from first paragraph about extension headers, and removed reference to RFC7045 because "examine" was removed (RFC7045 is referenced in Security Considerations). Also removed "including the source and



destination nodes" from paragraph about the hop-by-hop options header.

- 11) Revised Section 4.8 to make it closer to the update done by RFC6554 that updated it and reordered the paragraphs.
- 11) Reordered items in Appendix B "Changes Since RFC2460" to match the order of the document.
- 11) Editorial changes.
- 10) Revised and expanded Security Consideration Section based on IESG Discuss comments.
- 10) Editorial changes.
- 09) Based on results of IETF last call, changed text in Section 4 to add clarification that extension headers are not examined, processed, inserted, or deleted by any node along a packet's delivery path.
- 09) Changed reference from draft-ietf-6man-rfc4291bis to RFC4291 because the bis draft won't be advanced as the same time.
- 09) Revised "Changes since RFC2460" Section to have a summary of changes since RFC2460 and a separate subsection with a change history of each Internet Draft. This subsection will be removed when the RFC is published.
- 09) Editorial changes.
- 08) Revised header insertion text in Section 4 based on the results of w.g. survey that concluded to describe the problems with header insertion.
- 08) Editorial changes.
- 07) Expanded Security Considerations section to include both IPsec and encryption at higher levels in the protocol stack as ways to mitigate IP level security issues.
- 07) Added paragraph to Section 4 to clarify how Extension Headers are numbered and which are upper-layer headers.
- 07) Moved the text regarding network duplicated fragments to the received fragment error section.

- 07) Added clarification that no ICMP error message should be sent if overlapping fragments are received.
- 07) Revised the text in Section 4.8 regarding new hop-by-hop options and new Extension headers to be closer to the -05 version.
- 07) Added additional registries to the IANA Considerations section that IANA needs to update.
- 07) Editorial changes.
- 06) Added the Routing Header to the list required extension headers that a full implementation includes.
- 06) Moved the text in Section 4.5 regarding the handling of received overlapping fragments to the list of error conditions
- 06) Rewrote the text in Section 4.8 "Defining New Extension Headers and Options" to be clearer and remove redundant text.
- 06) Editorial changes.
- 05) Changed requirement for the Hop-by-Hop Options header from a should to a may, and added a note to indicate what is expected.
- 05) Corrected reference to point to draft-ietf-6man-rfc4291bis instead of draft-hinden-6man-rfc4291bis.
- 05) Change to text regarding not inserting extension headers to cite using encapsulation as an example.
- 04) Changed text discussing Fragment ID selection to refer to RFC7739 for example algorithms.
- 04) Editorial changes.
- 03) Clarified the text about decrementing the hop limit.
- 03) Removed IP Next Generation from the Abstract.
- 03) Add reference to the end of Section 4 to IPv6 Extension Header IANA registry.
- 03) Editorial changes.

- 02) Added text to Section 4.8 "Defining New Extension Headers and Options" clarifying why no new hop by hop extension headers should be defined.
- 02) Added text to Fragment Header process on handling exact duplicate fragments.
- 02) Editorial changes.
- 01) Added text that Extension headers must never be inserted by any node other than the source of the packet.
- 01) Change "must" to "should" in Section 4.3 on the Hop-by-Hop header.
- 01) Added text that the Data Transmission Order is the same as IPv4 as defined in RFC791.
- 01) Updated the Fragmentation header text to correct the inclusion of AH and note no next header case.
- 01) Change terminology in Fragment header section from "Unfragmentable Headers" to "Per-Fragment Headers".
- 01) Removed paragraph in Section 5 that required including a fragment header to outgoing packets if a ICMP Packet Too Big message reporting a Next-Hop MTU less than 1280. This is based on the update in RFC8021.
- 01) Changed to Fragmentation Header section to clarify MTU restriction and 8-byte restrictions, and noting the restriction on headers in first fragment.
- 01) Editorial changes.
- 00) Add instruction to the IANA to change references to RFC2460 to this document
- 00) Add a paragraph to the acknowledgement section acknowledging the authors of the updating documents
- 00) Remove old paragraph in Section 4 that should have been removed when incorporating the update from RFC7045.
- 00) Editorial changes.

Individual Internet Drafts

07) Update references to current versions and assign references to normative and informative.

07) Editorial changes.

06) The purpose of this draft is to incorporate the updates dealing with Extension headers as defined in RFC6564, RFC7045, and RFC7112. The changes include:

RFC6564: Added new Section 4.8 that describe recommendations for defining new Extension headers and options

RFC7045: The changes were to add a reference to RFC7045, change the requirement for processing the hop-by-hop option to a should, and added a note that due to performance restrictions some nodes won't process the Hop-by-Hop Option header.

RFC7112: The changes were to revise the Fragmentation Section (Section 4.5) to require that all headers through the first Upper-Layer Header are in the first fragment. This changed the text describing how packets are fragmented and reassembled and added a new error case.

06) Editorial changes.

05) The purpose of this draft is to incorporate the updates dealing with fragmentation as defined in RFC5722 and RFC6946. Note: The issue relating to the handling of exact duplicate fragments identified on the mailing list is left open.

05) Fix text in the end of Section 4 to correct the number of extension headers defined in this document.

05) Editorial changes.

04) The purpose of this draft is to update the document to incorporate the update made by RFC6935 "UDP Checksums for Tunneled Packets".

- 04) Remove Routing (Type 0) header from the list of required extension headers.
- 04) Editorial changes.
- 03) The purpose of this draft is to update the document for the deprecation of the RH0 Routing Header as specified in RFC5095 and the allocations guidelines for routing headers as specified in RFC5871. Both of these RFCs updated RFC2460.
- 02) The purpose of this version of the draft is to update the document to resolve the open Errata on RFC2460.

Errata ID: 2541: This errata notes that RFC2460 didn't update RFC2205 when the length of the Flow Label was changed from 24 to 20 bits from RFC1883. This issue was resolved in RFC6437 where the Flow Label is defined. This draft now references RFC6437. No change is required.

Errata ID: 4279: This errata noted that the specification doesn't handle the case of a forwarding node receiving a packet with a zero Hop Limit. This is fixed in Section 3 of this draft. Note: No change was made regarding host behaviour.

Errata ID: 2843: This errata is marked rejected. No change is required.

- 02) Editorial changes to the Flow Label and Traffic Class text.
- 01) The purpose of this version of the draft is to update the document to point to the current specifications of the IPv6 Flow Label field as defined in [RFC6437] and the Traffic Class as defined in [RFC2474] and [RFC3168].
- 00) The purpose of this version is to establish a baseline from RFC2460. The only intended changes are formatting (XML is slightly different from .nroff), differences between an RFC

and Internet Draft, fixing a few ID Nits, and updates to the authors information. There should not be any content changes to the specification.

#### Authors' Addresses

Stephen E. Deering  
Retired  
Vancouver, British Columbia  
Canada

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

Email: bob.hinden@gmail.com

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: April 16, 2016

J. Jeong  
S. Lee  
Sungkyunkwan University  
J. Park  
ETRI  
October 14, 2015

DNS Name Autoconfiguration for Internet of Things Devices  
draft-jeong-6man-iot-dns-autoconf-00

Abstract

This document specifies an autoconfiguration scheme for the global (or local) DNS names of Internet of Things (IoT) devices, such as appliances and sensors. By this scheme, the DNS name of an IoT device can be autoconfigured with the device's category and model in wired and wireless target networks (e.g., home, office, shopping mall, smart grid, and road network). This DNS name lets IoT users (e.g., home residents and customers) in the Internet (or local network) easily identify each device for monitoring and remote-controlling it in the target network.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at  
<http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at  
<http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on April 16, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	3
1.1. Applicability Statements . . . . .	4
2. Requirements Language . . . . .	4
3. Terminology . . . . .	4
4. Overview . . . . .	4
5. DNS Name Autoconfiguration . . . . .	5
5.1. DNS Name Format . . . . .	5
5.2. Procedure of DNS Name Autoconfiguration . . . . .	6
5.2.1. DNS Name Generation . . . . .	6
5.2.2. DNS Name Collection . . . . .	7
5.2.3. DNS Name Retrieval . . . . .	8
6. Location-Aware DNS Name Configuration . . . . .	8
6.1. Macro-Location-Aware DNS Name . . . . .	8
6.2. Micro-Location-Aware DNS Name . . . . .	9
7. DNS Name Management for Mobile IoT Devices . . . . .	10
8. Security Considerations . . . . .	10
9. Acknowledgements . . . . .	10
10. References . . . . .	11
10.1. Normative References . . . . .	11
10.2. Informative References . . . . .	11



## 1. Introduction

Many Internet of Things (IoT) devices (e.g., appliances and sensors) have begun to have wireless communication capability (e.g., WiFi, Bluetooth, and ZigBee) for monitoring and remote-controlling in a local network or the Internet. According to the capacity, such IoT devices can be categorized into high-capacity devices and low-capacity devices. High-capacity devices have a high-power processor and a large storage, such as appliances (e.g., television, refrigerator, air conditioner, and washing machine) and smart devices (smartphone and tablet). They are placed in environments (e.g., home, office, shopping mall, smart grid, and road network) for the direct use for human users, and they require the interaction with human users. Low-capacity devices have a low-power processor and a small storage, such as sensors (e.g., light, meter, room temperature controller, and sensors). They are installed for the easy management of environments (e.g., home, office, store, and factory), and they do not require the interaction with human users.

For the Internet connectivity of IoT devices, a variety of parameters (e.g., address prefixes, default routers, and DNS servers) can be automatically configured by Neighbor Discovery (ND) for IP Version 6, IPv6 Stateless Address Autoconfiguration, and IPv6 Router Advertisement (RA) Options for DNS Configuration [RFC4861][RFC4862][RFC6106].

For these IoT devices, the manual configuration of DNS names will be cumbersome and time-consuming as the number of them increases rapidly in a network. It will be good for such DNS names to be automatically configured such that they are readable to human users.

Multicast DNS (mDNS) in [RFC6762] can provide DNS service for networked devices on a local link (e.g., home network and office network) without any conventional recursive DNS server. mDNS also supports the autoconfiguration of a device's DNS name without the intervention of the user. mDNS aims at the DNS naming service for the local DNS names of the networked devices on the local link rather than the DNS naming service for the global DNS names of such devices in the Internet. However, for IoT devices accessible from the Internet, mDNS cannot be used. Thus, a new autoconfiguration scheme becomes required for the global DNS names of IoT devices.

This document proposes an autoconfiguration scheme for the global (or local) DNS names of IoT devices. Since an autoconfigured DNS name contains the device category and model of a device, IoT users in the Internet (or local network) can easily identify the device. With this device category and model, they will be able to monitor and remote-control each device with mobile smart devices (e.g.,

smartphone and tablet) by resolving its DNS name into the corresponding IPv6 address.

### 1.1. Applicability Statements

It is assumed that IoT devices have networking capability through wired or wireless communication media, such as Ethernet [IEEE-802.3], WiFi [IEEE-802.11] [IEEE-802.11a] [IEEE-802.11b][IEEE-802.11g] [IEEE-802.11n], Bluetooth [IEEE-802.15.1], and ZigBee [IEEE-802.15.4] in a local area network (LAN) or personal area network (PAN).

Also, it is assumed that each IoT device has a factory configuration (called device configuration) having device category (e.g., smart TV, smartphone, tablet, and refrigerator) and model (i.e., a specific model name of the device). This device configuration can be read by the device for DNS name autoconfiguration.

## 2. 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].

## 3. Terminology

This document uses the terminology described in [RFC4861] and [RFC4862]. In addition, four new terms are defined below:

- o Device Configuration: A factory configuration that has device category (e.g., smart TV, smartphone, tablet, and refrigerator) and model (i.e., a specific model name of the device).
- o DNS Search List (DNSSL): The list of DNS suffix domain names used by IPv6 hosts when they perform DNS query searches for short, unqualified domain names [RFC6106].
- o DNSSL Option: IPv6 RA option to deliver the DNSSL information to IPv6 hosts [RFC6106].

## 4. Overview

This document specifies an autoconfiguration scheme for an IoT device using device configuration and DNS search list. Device configuration has device category and device model. DNS search list has DNS suffix domain names that represent the DNS domains of a network having the IoT device [RFC6106].

As an IPv6 host, the IoT device can obtain DNS search list through

IPv6 Router Advertisement (RA) with DNS Search List (DNSSL) Option [RFC4861][RFC6106] or DHCPv6 with Domain Search List Option [RFC3315][RFC3736][RFC3646].

The IoT device can construct its DNS name with the concatenation of device category, device model, and domain name. Since there exist more than one device with the same model, the DNS name should have a unique identification to differentiate multiple devices with the same model.

Since both RA and DHCPv6 can be simultaneously used for the parameter configuration for IPv6 hosts, this document considers the DNS name autoconfiguration in the coexistence of RA and DHCP.

## 5. DNS Name Autoconfiguration

The DNS name autoconfiguration for an IoT device needs the acquisition of DNS search list through either RA [RFC6106] or DHCPv6 [RFC3646]. Once the DNS search list is obtained, the IoT device autonomously constructs its DNS name(s) with the DNS search list and its device information.

### 5.1. DNS Name Format

A DNS name for an IoT device has the following format as in Figure 1:

```
+-----+
| unique_id.device_model.device_category.domain_name |
+-----+
```

Figure 1: IoT Device DNS Name Format

#### Fields:

unique_id	unique identifier to guarantee the uniqueness of the DNS name in ASCII characters. The identifier MAY be a sequence number or alphanumeric with readability, such as product name.
device_model	device's model name in ASCII characters. It is a product model name provided by the manufacturer.
device_category	device's category name in ASCII characters, such as TV, refrigerator, air conditioner, smartphone, tablet, light, and meter.

domain\_name            DNS domain name that is encoded according to the specification of "Representation and use of domain name" of RFC 3315.

## 5.2. Procedure of DNS Name Autoconfiguration

The procedure of DNS name autoconfiguration is performed through a DNSSL option delivered by either RA [RFC6106] or DHCPv6 [RFC3646].

### 5.2.1. DNS Name Generation

When as an IPv6 host a device receives a DNSSL option through either RA or DHCPv6, it checks the validity of the DNSSL option. If the option is valid, the IPv6 host performs the DNS name autoconfiguration with each DNS suffix domain name in the DNSSL option as follows:

1. The host constructs its DNS name with the DNS suffix domain name along with device configuration (i.e., device category and device model) and a selected identifier (as unique\_id) that is considered unique, as shown in Figure 1.
2. The host constructs an IPv6 unicast address as a tentative address with a 64-bit network prefix and the last 64 bits of the MD5 hashed value of the above DNS name.
3. The host constructs the solicited-node multicast address in [RFC4861] corresponding to the tentative IPv6 address.
4. The host performs Duplicate Address Detection (DAD) for the IPv6 address with the solicited-node multicast address [RFC4861] [RFC4862].
5. If there is no response from the DAD, the host sets the IPv6 tentative address as its IPv6 unicast address and regards the constructed DNS name as unique on the local link. Otherwise, since the DAD fails because of DNS name conflict, go to Step 1 for a new DNS name generation with another identifier for unique\_id.
6. Since the DNS name is proven to be unique, it is used as the device's DNS name and the DNS autoconfiguration is done for the given DNS suffix domain name. Also, the host joins the solicited-node multicast address for the verified DNS name in order to prevent other hosts from using this DNS name.

When the DNS search list has more than one DNS suffix domain name, the IPv6 host repeats the above procedure until all of the DNS

suffixes are used for the DNS name autoconfiguration along with the IPv6 unicast autoconfiguration corresponding to the DNS name.

#### 5.2.2. DNS Name Collection

Once as IPv6 hosts the devices have autoconfigured their DNS names, as a collector, any IPv6 node (i.e., router or host) in the same subnet can collect the device DNS names using IPv6 Node Information (NI) protocol [RFC4620].

For a collector to collect the device DNS names without any prior node information, a new NI query needs to be defined. That is, a new ICMPv6 Code (e.g., 3) SHOULD be defined for the collection of the IPv6 host DNS names. The Data field is not included in the ICMPv6 header since the NI query is for all the IPv6 hosts in the same subnet. The Qtype field for NI type is set to 2 for Node Name.

The query SHOULD be transmitted by the collector to a link-local multicast address for this NI query. Assume that a link-local scope multicast address (e.g., all-nodes multicast address, FF02::1) SHOULD be defined for device DNS name collection such that all the IPv6 hosts join this link-local multicast address for the device DNS name collection service.

When an IPv6 host receives this query sent by the collector in multicast, it transmits its Reply with its DNS name with a random interval between zero and Query Response Interval, as defined by Multicast Listener Discovery Version 2 [RFC3810]. This randomly delayed Reply allows the collector to collect the device DNS names with less frame collision probability by spreading out the Reply time instants.

After the collector collects the device DNS names, it resolves the DNS names into the corresponding IPv6 addresses by NI protocol [RFC4620] with the ICMPv6 Code 1 of NI Query. This code indicates that the Data field of the NI Query has the DNS name of an IoT device. The IoT device that receives this NI query sends the collector an NI Reply with its IPv6 address in the Data field.

For DNS name resolution service, the collector can register the pair(s) of DNS name and IPv6 address for each IPv6 host into an appropriate designated DNS server for the DNS domain suffix of the DNS name. It is assumed that the collector is configured to register DNS names into the designated DNS server in a secure way based on DNSSEC [RFC4033][RFC6840]. This registration of the DNS name and IPv6 address can be performed by DNS dynamic update [RFC2136]. Before registering the DNS name into the designated DNS server, the collector SHOULD verify the uniqueness of the DNS name in the

intended DNS domain by sending a DNS query for the resolution of the DNS name. If there is no corresponding IPv6 address for the queried DNS name, the collector registers the DNS name and the corresponding IPv6 address into the designated DNS server. On the other hand, if there is such a corresponding IPv6 address, the DNS name is regarded as duplicate (i.e., not unique), and so the responder notifies the corresponding IoT device with the duplicate DNS name of an error message of DNS name duplication using NI protocol. When an IoT device receives such a DNS name duplication error, it needs to construct a new DNS name and repeats the procedure of device DNS name generation along with the uniqueness test of the device DNS name in its subnet.

The two separate procedures of the DNS name collection and IPv6 address resolution in the above NI protocol can be consolidated into a single collection for the pairs of DNS names and the corresponding IPv6 addresses. For such an optimization, a new ICMPv6 Code (e.g., 4) is defined for the NI Query to query the pair of a DNS name and the corresponding IPv6 address. With this code, the collector can collect the pairs of each IoT device's DNS name and IPv6 address in one NI query message rather than two NI query messages.

#### 5.2.3. DNS Name Retrieval

A smart device like smartphone can retrieve the DNS names of IoT devices by contacting a global (or local) DNS server having the IoT device DNS names. If the smart device can retrieve the zone file with the DNS names, it can display the information of IoT devices in a target network, such as home network and office network. With this information, the user can monitor and control the IoT devices in the Internet (or local network).

### 6. Location-Aware DNS Name Configuration

If the DNS name of an IoT device includes location information, it allows users to easily identify the physical location of each device. This document proposes the representation of location in a DNS name.

#### 6.1. Macro-Location-Aware DNS Name

If location information (such as living room, kitchen, and bedroom in an apartment) is available to an IoT device, a keyword for the location can be used to construct a DNS name as subdomain name. This location information lets users track the position of mobile devices (such as smartphone, tablet, and vacuum cleaning robot). The physical location of the device is defined as macro-location for DNS naming.

A subdomain name for macro-location (denoted as `mac_loc`) MAY be placed between `device_category` and `domain_name` of the DNS name format in Figure 2. A localization scheme for device location is beyond the scope of this document.

## 6.2. Micro-Location-Aware DNS Name

An IoT device can be located in the center, wall, or corner in a room that is specified by macro-location. For example, assume that a cleaning robot is located in the right-upper corner of a living room. If the DNS name for the cleaning robot contains the right-upper corner of the living room, a home resident can find it easily. In this document, for this DNS naming, the detailed location for an IoT device can be specified as a micro-location subdomain name.

A subdomain name for micro-location (denoted as `mic_loc`) MAY be placed between `device_category` and `domain_name` of the DNS name format in Figure 2. A localization scheme for micro-location is beyond the scope of this document.

To denote both macro-location (i.e., `mac_loc`) and micro-location (i.e., `mic_loc`) into a DNS name, the following format is described as in Figure 2:

```

+-----+
| unique_id.device_model.device_category.mic_loc.mac_loc.domain_name|
+-----+

```

Figure 2: Location-Aware Device DNS Name Format

### Fields:

<code>unique_id</code>	unique identifier to guarantee the uniqueness of the DNS name in ASCII characters. The identifier MAY be a sequence number or alphanumeric with readability, such as product name.
<code>device_model</code>	device's model name in ASCII characters. It is a product model name provided by the manufacturer.
<code>device_category</code>	device's category name in ASCII characters, such as TV, refrigerator, air conditioner, smartphone, tablet, light, and meter.
<code>mic_loc</code>	device's micro-location, such as center, wall, and corner.

mac_loc	device's macro-location, such as living room.
domain_name	DNS domain name that is encoded according to the specification of "Representation and use of domain name" of RFC 3315.

## 7. DNS Name Management for Mobile IoT Devices

Some IoT devices can have mobility, such as smartphone, tablet, laptop computer, and cleaning robot. This mobility allows the IoT devices to move from a subnet to another subnet where subnets can have different domain suffixes, such as living\_room.home and garage.home. The DNS name change (or addition) due to the mobility should be considered.

To deal with DNS name management in mobile environments, whenever an IoT device enters a new subnet and receives DNS suffix domain names, it generates its new DNS names and registers them into a designated DNS server, specified by RDNSS option.

When the IoT device recognizes the movement to another subnet, it can delete its previous DNS name(s) from the DNS server having the DNS name(s), using DNS dynamic update [RFC2136]. For at least one DNS name to remain in a DNS server for the location management in Mobile IPv6 [RFC6275], the IoT device does not delete its default DNS name in its home network in Mobile IPv6.

## 8. Security Considerations

This document shares all the security issues of the NI protocol that are specified in the "Security Considerations" section of [RFC4620].

To prevent the disclosure of location information for privacy concern, the subdomains related to location can be encrypted by a shared key or public-and-private keys. For example, a DNS name of smartphonel.living\_room.home can be represented as smartphonel.xxx.home where xxx is a string of the encrypted representation of the subdomain living\_room.

## 9. Acknowledgements

This work was partly supported by the ICT R&D program of MSIP/IITP [10041244, SmartTV 2.0 Software Platform] and ETRI.

## 10. References



## 10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 4861, September 2007.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.
- [RFC6106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 6106, November 2010.
- [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.
- [RFC3736] Droms, R., "Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6", RFC 3736, April 2004.
- [RFC3646] Droms, R., Ed., "DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3646, December 2003.
- [RFC4033] Arends, R., Ed., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.
- [RFC6840] Weiler, S., Ed. and D. Blacka, Ed., "Clarifications and Implementation Notes for DNS Security (DNSSEC)", RFC 6840, February 2013.

## 10.2. Informative References

- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, February 2013.
- [RFC4620] Crawford, M. and B. Haberman, Ed., "IPv6 Node Information Queries", RFC 4620, August 2006.
- [RFC3810] Vida, R. and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, June 2004.

- [RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.
- [RFC6275] Perkins, C., Ed., Johnson, D., and J. Arkko, "Mobility Support in IPv6", RFC 6275, July 2011.
- [IEEE-802.3] IEEE Std 802.3, "IEEE Standard for Ethernet", December 2012.
- [IEEE-802.11] IEEE Std 802.11, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", March 2012.
- [IEEE-802.11a] IEEE Std 802.11a, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band", September 1999.
- [IEEE-802.11b] IEEE Std 802.11b, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band", September 1999.
- [IEEE-802.11g] IEEE P802.11g/D8.2, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Further Higher Data Rate Extension in the 2.4 GHz Band", April 2003.
- [IEEE-802.11n] IEEE P802.11n/D9.0, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 5: Enhancements for Higher Throughput", March 2009.
- [IEEE-802.15.1] IEEE Std 802.15.1, "Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Wireless Personal Area Networks (WPANs)", June 2005.
- [IEEE-802.15.4] IEEE Std 802.15.4, "Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)", September 2011.

## Authors' Addresses

Jaehoon Paul Jeong  
Department of Software  
Sungkyunkwan University  
2066 Seobu-Ro, Jangan-Gu  
Suwon, Gyeonggi-Do 440-746  
Republic of Korea

Phone: +82 31 299 4957  
Fax: +82 31 290 7996  
EMail: pauljeong@skku.edu  
URI: <http://cpslab.skku.edu/people-jaehoon-jeong.php>

Sejun Lee  
Department of Computer Science and Engineering  
Sungkyunkwan University  
2066 Seobu-Ro, Jangan-Gu  
Suwon, Gyeonggi-Do 440-746  
Republic of Korea

Phone: +82 31 299 4106  
Fax: +82 31 290 7996  
EMail: sejunlee@skku.edu

Jung-Soo Park  
Electronics and Telecommunications Research Institute  
218 Gajeong-Ro, Yuseong-Gu  
Daejeon, 305-700  
Republic of Korea

Phone: +82 42 860 6514  
EMail: pjs@etri.re.kr



IPv6 Maintenance  
Internet-Draft  
Updates: 4861 (if approved)  
Intended status: Standards Track  
Expires: April 20, 2016

S. Krishnan  
Ericsson  
J. Korhonen  
Broadcom  
S. Chakrabarti  
Ericsson  
E. Nordmark  
Arista Networks  
A. Yourtchenko  
cisco  
October 18, 2015

Support for adjustable maximum router lifetimes per-link  
draft-krishnan-6man-maxra-03

#### Abstract

The neighbor discovery protocol specifies the maximum time allowed between sending unsolicited multicast Router Advertisements from a router interface as well as the maximum router lifetime. It also allows the limits to be overridden by link-layer specific documents. This document allows for overriding these values on a per-link basis.

#### 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 April 20, 2016.

#### Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	2
3. Relationship between AdvDefaultLifetime and MaxRtrAdvInterval	3
4. Updates to RFC4861 . . . . .	4
5. Host Behavior . . . . .	4
6. Security Considerations . . . . .	4
7. IANA Considerations . . . . .	4
8. Acknowledgements . . . . .	4
9. References . . . . .	4
9.1. Normative References . . . . .	5
9.2. Informative References . . . . .	5
Authors' Addresses . . . . .	5

## 1. Introduction

IPv6 Neighbor Discovery relies on IP multicast with the expectation to be efficient with respect to available bandwidth and to avoid generating interrupts in the network nodes. On some datalink-layer network, for example IEEE 802.11 WiFi, this is not the case because of limitations in the services offered by the datalink-layer network [draft-vyncke-6man-mcast-not-efficient-01]. On such links any possible reduction of multicast traffic will be highly beneficial. Unfortunately, due to the fixed protocol constants specified in [RFC4861] it is difficult to relax the multicast timers for neighbor discovery. There are already link technology specific clarifications how to tune protocol constants for certain system with the expectation to reduce excess Neighbor Discovery Protocol (NDP) traffic. 3GPP cellular links are one existing example [RFC6459][RFC7066].

This document specifies updates to the IPv6 Neighbor Discovery Protocol [RFC4861] for relaxing the the maximum time allowed between sending unsolicited multicast Router Advertisements (RA) from a router interface as well as for the maximum router lifetime.

## 2. Terminology

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

### 3. Relationship between AdvDefaultLifetime and MaxRtrAdvInterval

MaxRtrAdvInterval is an upper bound on the time between the two successive Router Advertisement messages are sent, therefore one might reason about the relationship between these two values in terms of the ratio  $K = \text{AdvDefaultLifetime} / \text{MaxRtrAdvInterval}$ , which expresses how many Router Advertisements will be guaranteed to be sent before the router lifetime expiry.

Assuming unicast Solicited Router Advertisements or a perfectly stable network, on a theoretically perfect link with no losses, it would have been sufficient to have  $K$  just above 1 - so that the sent Router Advertisement refreshes the router entry just before it expires. On the real links which allow for some loss, one would need to use  $K > 2$  in order to minimize the chances of a single router advertisement loss causing a loss of the router entry.

The exact calculation will depend on the packet loss probability. An example: if we take a ballpark value of 1% probability of a packet loss, then  $K=2$  will give 0.01% percent chance of an outage due to a packet loss,  $K=3$  will give 0.0001% chance of an outage, and so forth. To reverse the numbers, with these parameters,  $K \sim 1$  gives 99% reliability,  $K \sim 2$  gives 99.99% reliability, and  $K \sim 3$  gives 99.9999% reliability - the latter should be good enough for a lot of scenarios.

In a network with higher packet loss or if the higher reliability is desired, the  $K$  might be chosen to be even higher. On the other hand, some of the data link layers provide reliable delivery at layer 2 - so there one might even consider using the "theoretical" value of  $K$  just above 1. Since the choice of these two parameters does not impact the interoperability per se, this document does not impose any specific constraints on their values other than providing the guidelines in this section, therefore each individual link can optimize accordingly to its use case.

Also AdvDefaultLifetime MUST be set to a value greater than or equal to the selected MaxRtrAdvInterval. Otherwise, a router lifetime is guaranteed to expire before the new Router Advertisement has a chance to be sent, thereby creating an outage.

#### 4. Updates to RFC4861

This document updates Section 6.2.1. of [RFC4861] to update the following router configuration variables. MaxRtrAdvInterval MUST be no greater than 65535. AdvDefaultLifetime MUST be between MaxRtrAdvInterval and 65535.

This document also updates Sections 6.2.1., 6.2.2., 6.2.4. and 6.2.5. of [RFC4861] so that AdvSendAdvertisements can be set "FALSE" but the router would still continue be a router and respond with RAs to solicited RSes .

#### 5. Host Behavior

Legacy hosts on a link with updated routers may have issues with a Router Lifetime of more than 9000 seconds. In the few implementations we have tested with general purpose operating systems, there does not seem to be any issues with setting this field to more than 9000, but there might be implementations that incorrectly (since RFC4861 requires receivers to handle any value) reject such RAs.

#### 6. Security Considerations

On a link where router advertisements are few and far between, the attack window for a rogue router to send an unsolicited RA is greatly increased. These attacks can easily be prevented by using SeND [RFC3971]

#### 7. IANA Considerations

This document does not require any IANA action.

#### 8. Acknowledgements

The authors would like to thank the members of the 6man efficient ND design team for their comments that led to the creation of this draft. The authors would also like to thank Lorenzo Colitti, Erik Kline and Jeena Rachel John for their comments and suggestions that improved this document.

#### 9. References



## 9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

## 9.2. Informative References

- [RFC6459] Korhonen, J., Soinen, J., Patil, B., Savolainen, T., Bajko, G., and K. Iisakkila, "IPv6 in 3rd Generation Partnership Project (3GPP) Evolved Packet System (EPS)", RFC 6459, January 2012.
- [RFC7066] Korhonen, J., Arkko, J., Savolainen, T., and S. Krishnan, "IPv6 for Third Generation Partnership Project (3GPP) Cellular Hosts", RFC 7066, November 2013.

## Authors' Addresses

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

Phone: +1 514 345 7900 x42871  
Email: suresh.krishnan@ericsson.com

Jouni Korhonen  
Broadcom  
Porkkalankatu 24  
FIN-00180 Helsinki  
Finland

Email: jouni.nospam@gmail.com

Samita Chakrabarti  
Ericsson  
USA

Email: [samita.chakrabarti@ericsson.com](mailto:samita.chakrabarti@ericsson.com)

Erik Nordmark  
Arista Networks  
Santa Clara, CA  
USA

Email: [nordmark@acm.org](mailto:nordmark@acm.org)

Andrew Yourtchenko  
cisco  
6b de Kleetlaan  
Diegem 1831  
Belgium

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

IPv6 Maintenance  
Internet-Draft  
Intended status: Standards Track  
Expires: January 7, 2016

S. Krishnan  
Ericsson  
July 6, 2015

Uplink access technology indications in Router Advertisements  
draft-krishnan-6man-uat-00

Abstract

In IPv6 networks Router Advertisements can be used for providing common configuration information to nodes that are attached. There are some scenarios where it is advantageous for routers to provide their uplink access technology information to attached hosts. This document describes a neighbor discovery option that will allow the routers to do so.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 7, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	2
3. Uplink Access Technology option . . . . .	3
4. Router Behavior . . . . .	3
5. Host Behavior . . . . .	4
6. Security Considerations . . . . .	4
7. IANA Considerations . . . . .	4
8. Acknowledgements . . . . .	4
9. Normative References . . . . .	4
Author's Address . . . . .	5

## 1. Introduction

In several IPv6 networks, the access technology used by routers on their uplinks is different from that used on their downlinks. There are some scenarios where it is advantageous for routers to provide their uplink access technology information to the hosts attached on the downlinks. One such example is a tethering scenario where a mobile phone that uses a cellular uplink such as LTE, shares its internet connection to hosts that connect over a local WiFi link. In this case it would be beneficial for hosts to know that the uplink connection is a cellular link and potentially modify their behavior based on their knowledge. e.g. Application and software updates (and similar bulk transfers) could be rescheduled based on administrative configuration.

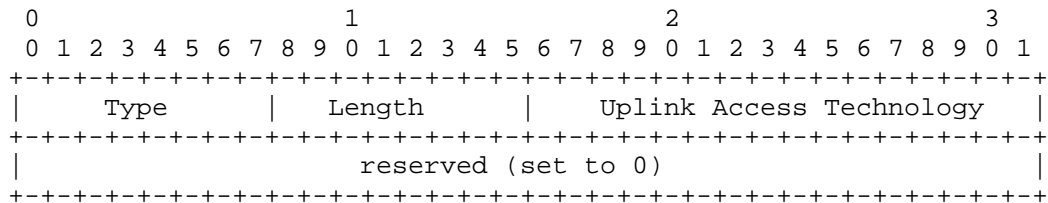
This document describes an IPv6 Neighbor discovery option [RFC4861] for routers to advertise their uplink access technology(ies) in a router advertisement message.

## 2. Terminology

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

### 3. Uplink Access Technology option

This option is to be carried in RA messages sent out by a router on a given link. It specifies the uplink type(s) that the router uses.



#### Type

8-bit identifier of the type of option. The option identifier for the UAT option will be allocated by the IANA.

#### Length

8-bit unsigned integer. The length of the option (including the type and length fields) in units of 8 octets. It MUST be set to 1.

#### Uplink Access Technology

A 16-bit field that specifies the uplink access technology used by the router sending the Router Advertisement carrying this option.

Figure 1: Uplink Access Technology (UAT) Option Layout

Multiple UAT options MAY be present in a single Router Advertisement message to allow for routers that use multiple uplinks. This document defines the following initial values for the UAT field that can be extended by adding new values to the IANA registry.

UAT	Access Technology
0x01	3GPP
0x02	DSL
0x03	Cable
0x04	802.3

Figure 2

### 4. Router Behavior

The value of the UAT(s) provided in this option can either be administratively configured or implicitly derived from the access technology type on the uplink interfaces.

## 5. Host Behavior

The value of the UAT(s) provided in this option is purely informational. It helps the hosts glean additional information about the router's uplink and perform different actions. Legacy hosts that do not recognize this option will simply ignore it.

## 6. Security Considerations

An attacker may attempt to modify the information provided inside this option to make hosts . These attacks can easily be prevented by using SeND [RFC3971]

## 7. IANA Considerations

This document defines a new IPv6 neighbor discovery option for carrying the uplink access technology type(s). IANA is requested to create a new registry for storing uplink access technology types and populate it with the following initial values.

UAT	Access Technology	Reference
0x01	3GPP	[RFC-krishnan-6man-uat-00.txt]
0x02	DSL	[RFC-krishnan-6man-uat-00.txt]
0x03	Cable	[RFC-krishnan-6man-uat-00.txt]
0x04	802.3	[RFC-krishnan-6man-uat-00.txt]

Figure 3

## 8. Acknowledgements

TBA.

## 9. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

Author's Address

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

Phone: +1 514 345 7900 x42871  
Email: suresh.krishnan@ericsson.com

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: April 4, 2016

S. Previdi, Ed.  
C. Filsfils  
Cisco Systems, Inc.  
B. Field  
Comcast  
I. Leung  
Rogers Communications  
J. Linkova  
Google  
E. Aries  
Facebook  
T. Kosugi  
NTT  
E. Vyncke  
Cisco Systems, Inc.  
D. Lebrun  
Universite Catholique de Louvain  
October 2, 2015

IPv6 Segment Routing Header (SRH)  
draft-previdi-6man-segment-routing-header-08

Abstract

Segment Routing (SR) allows a node to steer a packet through a controlled set of instructions, called segments, by prepending a SR header to the packet. A segment can represent any instruction, topological or service-based. SR allows to enforce a flow through any path (topological, or application/service based) while maintaining per-flow state only at the ingress node to the SR domain.

Segment Routing can be applied to the IPv6 data plane with the addition of a new type of Routing Extension Header. This draft describes the Segment Routing Extension Header Type and how it is used by SR capable nodes.

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 <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 April 4, 2016.

#### Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

#### Table of Contents

1. Segment Routing Documents . . . . .	3
2. Introduction . . . . .	3
2.1. Data Planes supporting Segment Routing . . . . .	4
2.2. Segment Routing (SR) Domain . . . . .	4
2.2.1. SR Domain in a Service Provider Network . . . . .	5
2.2.2. SR Domain in a Overlay Network . . . . .	6
2.3. Illustration . . . . .	8
3. IPv6 Instantiation of Segment Routing . . . . .	10
3.1. Segment Identifiers (SIDs) . . . . .	10
3.1.1. Node-SID . . . . .	10
3.1.2. Adjacency-SID . . . . .	11
3.2. Segment Routing Extension Header (SRH) . . . . .	11
3.2.1. SRH and RFC2460 behavior . . . . .	14
4. SRH Procedures . . . . .	15
4.1. Segment Routing Node Functions . . . . .	15
4.1.1. Source SR Node . . . . .	16
4.1.2. SR Domain Ingress Node . . . . .	17
4.1.3. Transit Node . . . . .	17
4.1.4. SR Segment Endpoint Node . . . . .	17

5.	Security Considerations . . . . .	18
5.1.	Threat model . . . . .	19
5.1.1.	Source routing threats . . . . .	19
5.1.2.	Applicability of RFC 5095 to SRH . . . . .	19
5.1.3.	Service stealing threat . . . . .	20
5.1.4.	Topology disclosure . . . . .	20
5.1.5.	ICMP Generation . . . . .	20
5.2.	Security fields in SRH . . . . .	21
5.2.1.	Selecting a hash algorithm . . . . .	22
5.2.2.	Performance impact of HMAC . . . . .	22
5.2.3.	Pre-shared key management . . . . .	23
5.3.	Deployment Models . . . . .	23
5.3.1.	Nodes within the SR domain . . . . .	23
5.3.2.	Nodes outside of the SR domain . . . . .	24
5.3.3.	SR path exposure . . . . .	24
5.3.4.	Impact of BCP-38 . . . . .	25
6.	IANA Considerations . . . . .	25
7.	Manageability Considerations . . . . .	25
8.	Contributors . . . . .	25
9.	Acknowledgements . . . . .	26
10.	References . . . . .	26
10.1.	Normative References . . . . .	26
10.2.	Informative References . . . . .	26
	Authors' Addresses . . . . .	28

## 1. Segment Routing Documents

Segment Routing terminology is defined in  
[I-D.ietf-spring-segment-routing].

Segment Routing use cases are described in  
[I-D.ietf-spring-problem-statement] and  
[I-D.ietf-spring-ipv6-use-cases].

Segment Routing protocol extensions are defined in  
[I-D.ietf-isis-segment-routing-extensions], and  
[I-D.ietf-ospf-ospfv3-segment-routing-extensions].

## 2. Introduction

Segment Routing (SR), defined in [I-D.ietf-spring-segment-routing], allows a node to steer a packet through a controlled set of instructions, called segments, by prepending a SR header to the packet. A segment can represent any instruction, topological or service-based. SR allows to enforce a flow through any path (topological or service/application based) while maintaining per-flow state only at the ingress node to the SR domain. Segments can be derived from different components: IGP, BGP, Services, Contexts,

Locators, etc. The list of segment forming the path is called the Segment List and is encoded in the packet header.

SR allows the use of strict and loose source based routing paradigms without requiring any additional signaling protocols in the infrastructure hence delivering an excellent scalability property.

The source based routing model described in [I-D.ietf-spring-segment-routing] is inherited from the ones proposed by [RFC1940] and [RFC2460]. The source based routing model offers the support for explicit routing capability.

## 2.1. Data Planes supporting Segment Routing

Segment Routing (SR), can be instantiated over MPLS ([I-D.ietf-spring-segment-routing-mpls]) and IPv6. This document defines its instantiation over the IPv6 data-plane based on the use-cases defined in [I-D.ietf-spring-ipv6-use-cases].

This document defines a new type of Routing Header (originally defined in [RFC2460]) called the Segment Routing Header (SRH) in order to convey the Segment List in the packet header as defined in [I-D.ietf-spring-segment-routing]. Mechanisms through which segment are known and advertised are outside the scope of this document.

A segment is materialized by an IPv6 address. A segment identifies a topological instruction or a service instruction. A segment can be either:

- o global: a global segment represents an instruction supported by all nodes in the SR domain and it is instantiated through an IPv6 address globally known in the SR domain.
- o local: a local segment represents an instruction supported only by the node who originates it and it is instantiated through an IPv6 address that is known only by the local node.

## 2.2. Segment Routing (SR) Domain

We define the concept of the Segment Routing Domain (SR Domain) as the set of nodes participating into the source based routing model. These nodes may be connected to the same physical infrastructure (e.g.: a Service Provider's network) as well as nodes remotely connected to each other (e.g.: an enterprise VPN or an overlay).

A non-exhaustive list of examples of SR Domains is:

- o The network of an operator, service provider, content provider, enterprise including nodes, links and Autonomous Systems.
- o A set of nodes connected as an overlay over one or more transit providers. The overlay nodes exchange SR-enabled traffic with segments belonging solely to the overlay routers (the SR domain). None of the segments in the SR-enabled packets exchanged by the overlay belong to the transit networks

The source based routing model through its instantiation of the Segment Routing Header (SRH) defined in this document equally applies to all the above examples.

While the source routing model defined in [RFC2460] doesn't mandate which node is allowed to insert (or modify) the SRH, it is assumed in this document that the SRH is inserted in the packet by its source. For example:

- o At the node originating the packet (host, server).
- o At the ingress node of a SR domain where the ingress node receives an IPv6 packet and encapsulates it into an outer IPv6 header followed by a Segment Routing header.

### 2.2.1. SR Domain in a Service Provider Network

The following figure illustrates an SR domain consisting of an operator's network infrastructure.

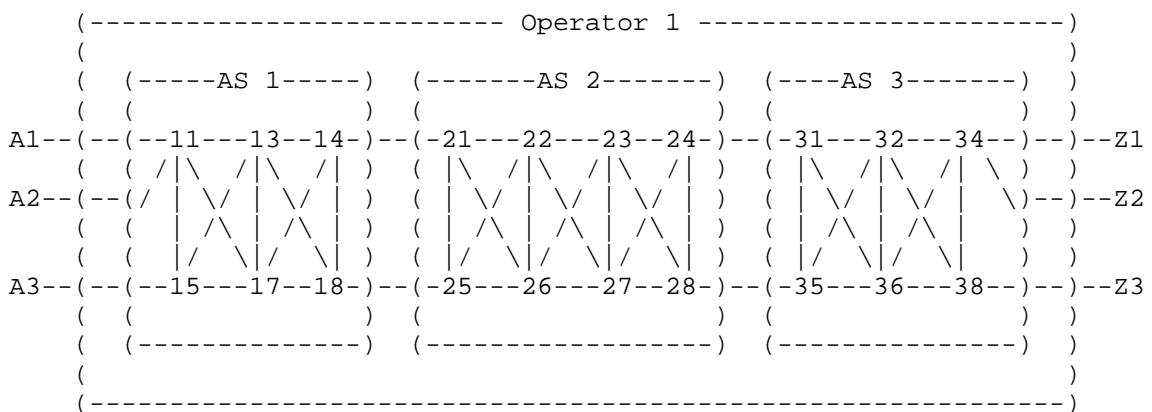


Figure 1: Service Provider SR Domain

Figure 1 describes an operator network including several ASes and delivering connectivity between endpoints. In this scenario, Segment

Routing is used within the operator networks and across the ASes boundaries (all being under the control of the same operator). In this case segment routing can be used in order to address use cases such as end-to-end traffic engineering, fast re-route, egress peer engineering, data-center traffic engineering as described in [I-D.ietf-spring-problem-statement], [I-D.ietf-spring-ipv6-use-cases] and [I-D.ietf-spring-resiliency-use-cases].

Typically, an IPv6 packet received at ingress (i.e.: from outside the SR domain), is classified according to network operator policies and such classification results into an outer header with an SRH applied to the incoming packet. The SRH contains the list of segment representing the path the packet must take inside the SR domain. Thus, the SA of the packet is the ingress node, the DA (due to SRH procedures described in Section 4) is set as the first segment of the path and the last segment of the path is the egress node of the SR domain.

The path may include intra-AS as well as inter-AS segments. It has to be noted that all nodes within the SR domain are under control of the same administration. When the packet reaches the egress point of the SR domain, the outer header and its SRH are removed so that the destination of the packet is unaware of the SR domain the packet has traversed.

The outer header with the SRH is no different from any other tunneling encapsulation mechanism and allows a network operator to implement traffic engineering mechanisms so to efficiently steer traffic across his infrastructure.

#### 2.2.2. SR Domain in a Overlay Network

The following figure illustrates an SR domain consisting of an overlay network over multiple operator's networks.

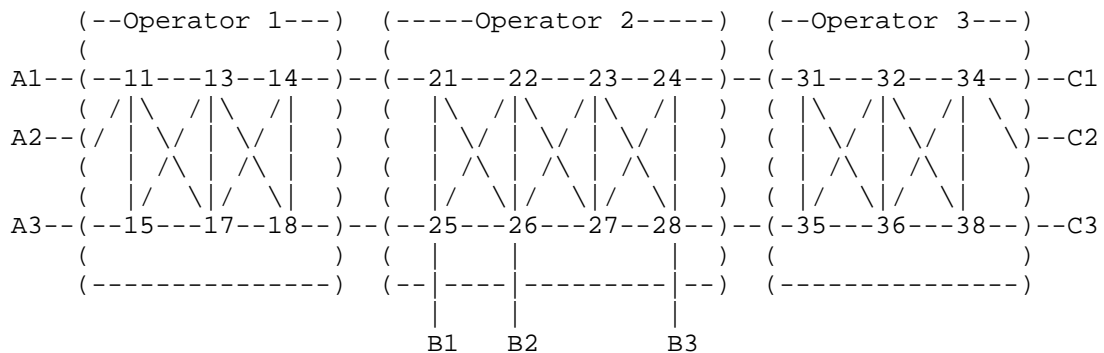


Figure 2: Overlay SR Domain

Figure 2 describes an overlay consisting of nodes connected to three different network operators and forming a single overlay network where Segment routing packets are exchanged.

The overlay consists of nodes A1, A2, A3, B1, B2, B3, C1, C2 and C3. These nodes are connected to their respective network operator and form an overlay network.

Each node may originate packets with an SRH which contains, in the segment list of the SRH or in the DA, segments identifying other overlay nodes. This implies that packets with an SRH may traverse operator's networks but, obviously, these SRHs cannot contain an address/segment of the transit operators 1, 2 and 3. The SRH originated by the overlay can only contain address/segment under the administration of the overlay (e.g. address/segments supported by A1, A2, A3, B1, B2, B3, C1, C2 or C3).

In this model, the operator network nodes are transit nodes and, according to [RFC2460], MUST NOT inspect the routing extension header since there are not the DA of the packet.

It is a common practice in operators networks to filter out, at ingress, any packet whose DA is the address of an internal node and it is also possible that an operator would filter out any packet destined to an internal address and having an extension header in it.

This common practice does not impact the SR-enabled traffic between the overlay nodes as the intermediate transit networks do never see a destination address belonging to their infrastructure. These SR-enabled overlay packets will thus never be filtered by the transit operators.

In all cases, transit packets (i.e.: packets whose DA is outside the domain of the operator's network) will be forwarded accordingly without introducing any security concern in the operator's network. This is similar to tunneled packets.

### 2.3. Illustration

In the context of Figure 3 we illustrate an example of how segment routing can be used within a SR domain in order to engineer traffic. Let's assume that the SR domain is configured as a single AS and the IGP (OSPF or IS-IS) is configured using the same cost on every link. Let's also assume that a packet P enters the SR domain at an ingress edge router I and that the operator requests the following requirements for packet P:

- o The local service S offered by node B must be applied to packet P.
- o The links AB and CE cannot be used to transport the packet P.
- o Any node N along the journey of the packet should be able to determine where the packet P entered the SR domain and where it will exit. The intermediate node should be able to determine the paths from the ingress edge router to itself, and from itself to the egress edge router.
- o Per-flow State for packet P should only be created at the ingress edge router.
- o The operator can forbid, for security reasons, anyone outside the operator domain to exploit its intra-domain SR capabilities.

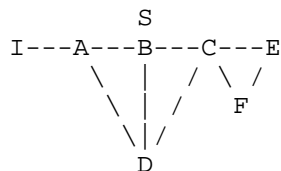


Figure 3: An illustration of SR properties

All these properties may be realized by instructing the ingress SR edge router I to create a SRH with the list of segments the packet must traverse: D, B, S, F, E. Therefore, the ingress router I creates an outer header where:

- o the SA is the IPv6 address of I

- o the final destination of the packet is the SR egress node E however, D being the first segment of the path, the DA is set to D IPv6 address.
- o the SRH is inserted with the segment list consisting of following IPv6 addresses: D, B, S, F, E

The SRH contains a source route encoded as a list of segments (D, B, S, F, E). The ingress and egress nodes are identified in the packet respectively by the SA and the last segment of the segment list.

The packet P reaches the ingress SR node I. Node I pushes the newly created outer header and SRH with the Segment List as illustrated above (D, B, S, F, E)

D is the IPv6 address of node D and it is recognized by all nodes in the SR domain as the forwarding instruction "forward to D according to D route in the IPv6 routing table". The routing table being built through IGPs (OSPF or IS-IS) it is equivalent to say "forward according to shortest path to D".

Once at D, the next segment is inspected and executed (segment B).

B is an instruction recognized by all the nodes in the SR domain which causes the packet to be forwarded along the shortest path to B.

Once at B, the next segment is executed (segment S).

S is an instruction only recognized by node B which causes the packet to receive service S.

Once the service S is applied, the next segment is executed (segment F) which causes the packet to be forwarded along the shortest path to F.

Once at F, the next segment is executed (segment E).

E is an instruction recognized by all the nodes in the SR domain which causes the packet to be forwarded along the shortest path to E.

E being the destination of the packet, removes the outer header and the SRH. Then, it inspects the inner packet header and forwards the packet accordingly.

All of the requirements are met:

- o First, the packet P has not used links AB and CE: the shortest-path from I to D is I-A-D, the shortest-path from D to B is D-B,



the shortest-path from B to F is B-C-F and the shortest-path from F to E is F-E, hence the packet path through the SR domain is I-A-D-B-C-F-E and the links AB and CE have been avoided.

- o Second, the service S supported by B has been applied on packet P.
- o Third, any node along the packet path is able to identify the service and topological journey of the packet within the SR domain by inspecting the SRH and SA/DA fields of the packet header.
- o Fourth, only node I maintains per-flow state for packet P. The entire program of topological and service instructions to be executed by the SR domain on packet P is encoded by the ingress edge router I in the SR header in the form of a list of segments where each segment identifies a specific instruction. No further per-flow state is required along the packet path. Intermediate nodes only hold states related to the global node segments and their local segments. These segments are not per-flow specific and hence scale very well. Typically, an intermediate node would maintain in the order of 100's to 1000's global node segments and in the order of 10's to 100 of local segments.
- o Fifth, the SR header (and its outer header) is inserted at the entrance to the domain and removed at the exit of the operator domain. For security reasons, the operator can forbid anyone outside its domain to use its intra-domain SR capability (e.g. configuring ACL that deny any packet with a DA towards its infrastructure segment).

### 3. IPv6 Instantiation of Segment Routing

#### 3.1. Segment Identifiers (SIDs)

Segment Routing, as described in [I-D.ietf-spring-segment-routing], defines Node-SID and Adjacency-SID. When SR is used over IPv6 data-plane the following applies.

##### 3.1.1. Node-SID

The Node-SID identifies a node. With SR-IPv6 the Node-SID is an IPv6 address that the operator configured on the node and that is used as the node identifier. Typically, in case of a router, this is the IPv6 address of the node loopback interface. Therefore, SR-IPv6 does not require any additional SID advertisement for the Node Segment. The Node-SID is in fact the IPv6 address of the node.

### 3.1.2. Adjacency-SID

Adjacency-SIDs can be either globally scoped IPv6 addresses or IPv6 addresses known locally by the node but not advertised in any control plane (in other words an Adjacency-SID may well be any 128-bit identifier). Obviously, in the latter case, the scope of the Adjacency-SID is local to the router and any packet with the a such Adjacency-SID would need first to reach the node through the node's Segment Identifier (i.e.: Node-SID) prior for the node to process the Adjacency-SID. In other words, two segments (SIDs) would then be required: the first is the node's Node-SID that brings the packet to the node and the second is the Adjacency-SID that will make the node to forward the packet through the interface the Adjacency-SID is allocated to.

In the SR architecture defined in [I-D.ietf-spring-segment-routing] a node may advertise one (or more) Adj-SIDs allocated to the same interface as well as a node can advertise the same Adj-SID for multiple interfaces. Use cases of Adj-SID advertisements are described in [I-D.ietf-spring-segment-routing] The semantic of the Adj-SID is:

Send out the packet to the interface this Adj-SID is allocated to.

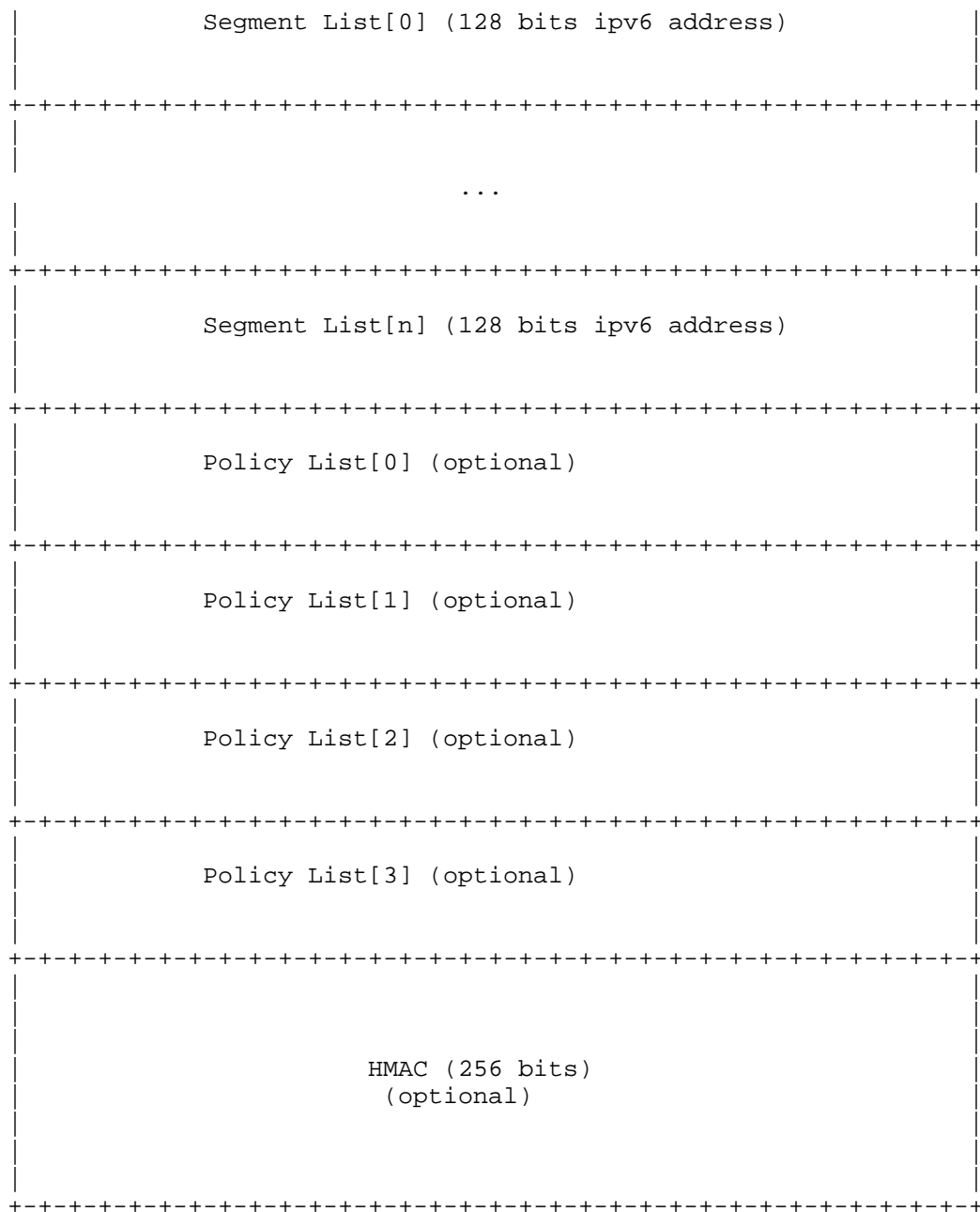
Advertisement of Adj-SID may be done using multiple mechanisms among which the ones described in ISIS and OSPF protocol extensions: [I-D.ietf-isis-segment-routing-extensions] and [I-D.ietf-ospf-ospfv3-segment-routing-extensions]. The distinction between local and global significance of the Adj-SID is given in the encoding of the Adj-SID advertisement.

### 3.2. Segment Routing Extension Header (SRH)

A new type of the Routing Header (originally defined in [RFC2460]) is defined: the Segment Routing Header (SRH) which has a new Routing Type, (suggested value 4) to be assigned by IANA.

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

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
Next Header	Hdr Ext Len	Routing Type	Segments Left
First Segment	Flags	HMAC Key ID	



where:

- o Next Header: 8-bit selector. Identifies the type of header immediately following the SRH.
- o Hdr Ext Len: 8-bit unsigned integer, is the length of the SRH header in 8-octet units, not including the first 8 octets.
- o Routing Type: TBD, to be assigned by IANA (suggested value: 4).
- o Segments Left. Defined in [RFC2460], it contains the index, in the Segment List, of the next segment to inspect. Segments Left is decremented at each segment.
- o First Segment: contains the index, in the Segment List, of the first segment of the path which is in fact the last element of the Segment List.
- o Flags: 16 bits of flags. Following flags are defined:

```

                                1
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+---+
|C|P|R|R|   Policy Flags   |
+---+---+---+---+---+---+---+---+

```

C-flag: Clean-up flag. Set when the SRH has to be removed from the packet when packet reaches the last segment.

P-flag: Protected flag. Set when the packet has been rerouted through FRR mechanism by a SR endpoint node.

R-flags. Reserved and for future use.

Policy Flags. Define the type of the IPv6 addresses encoded into the Policy List (see below). The following have been defined:

Bits 4-6: determine the type of the first element after the segment list.

Bits 7-9: determine the type of the second element.

Bits 10-12: determine the type of the third element.

Bits 13-15: determine the type of the fourth element.

The following values are used for the type:

0x0: Not present. If value is set to 0x0, it means the element represented by these bits is not present.

0x1: SR Ingress.

0x2: SR Egress.

0x3: Original Source Address.

0x4 to 0x7: currently unused and SHOULD be ignored on reception.

- o HMAC Key ID and HMAC field, and their use are defined in Section 5.
- 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 path. I.e., the first element of the segment list (Segment List [0]) contains the last segment of the path while the last segment of the Segment List (Segment List[n]) contains the first segment of the path. The index contained in "Segments Left" identifies the current active segment.
- o Policy List. Optional addresses representing specific nodes in the SR path such as:

SR Ingress: a 128 bit generic identifier representing the ingress in the SR domain (i.e.: it needs not to be a valid IPv6 address).

SR Egress: a 128 bit generic identifier representing the egress in the SR domain (i.e.: it needs not to be a valid IPv6 address).

Original Source Address: IPv6 address originally present in the SA field of the packet.

The segments in the Policy List are encoded after the segment list and they are optional. If none are in the SRH, all bits of the Policy List Flags MUST be set to 0x0.

### 3.2.1. SRH and RFC2460 behavior

The SRH being a new type of the Routing Header, it also has the same properties:

SHOULD only appear once in the packet.

Only the router whose address is in the DA field of the packet header MUST inspect the SRH.

Therefore, Segment Routing in IPv6 networks implies that the segment identifier (i.e.: the IPv6 address of the segment) is moved into the DA of the packet.

The DA of the packet changes at each segment termination/completion and therefore the original DA of the packet MUST be encoded as the last segment of the path.

As illustrated in Section 2.3, nodes that are within the path of a segment will forward packets based on the DA of the packet without inspecting the SRH. This ensures full interoperability between SR-capable and non-SR-capable nodes.

#### 4. SRH Procedures

In this section we describe the different procedures on the SRH.

##### 4.1. Segment Routing Node Functions

SR packets are forwarded to segments endpoints (i.e.: the segment endpoint is the node representing the segment and whose address is in the segment list and in the DA of the packet when traveling in the segment). The segment endpoint, when receiving a SR packet destined to itself, does:

- o Inspect the SRH.
- o Determine the next active segment.
- o Update the Segments Left field (or, if requested, remove the SRH from the packet).
- o Update the DA.
- o Forward the packet to the next segment.

The procedures applied to the SRH are related to the node function. Following nodes functions are defined:

Source SR Node.

SR Domain Ingress Node.

Transit Node.

SR Endpoint Node.

#### 4.1.1.1. Source SR Node

A Source SR Node can be any node originating an IPv6 packet with its IPv6 and Segment Routing Headers. This include either:

A host originating an IPv6 packet

A SR domain ingress router encapsulating a received IPv6 packet into an outer IPv6 header followed by a SRH

The mechanism through which a Segment List is derived is outside of the scope of this document. As an example, the Segment List may be obtained through:

Local path computation.

Local configuration.

Interaction with a centralized controller delivering the path.

Any other mechanism.

The following are the steps of the creation of the SRH:

Next Header and Hdr Ext Len fields are set according to [RFC2460].

Routing Type field is set as TBD (SRH).

The Segment List is built with the FIRST segment of the path encoded in the LAST element of the Segment List. Subsequent segments are encoded on top of the first segment. Finally, the LAST segment of the path is encoded in the FIRST element of the Segment List. In other words, the Segment List is encoded in the reverse order of the path.

The original DA of the packet is encoded as the last segment of the path (encoded in the first element of the Segment List).

The DA of the packet is set with the value of the first segment (found in the last element of the segment list).

The Segments Left field is set to n-1 where n is the number of elements in the Segment List.

The First Segment field is set to n-1 where n is the number of elements in the Segment List.

The packet is sent out towards the first segment (i.e.: represented in the packet DA).

HMAC and HMAC Key ID may be set according to Section 5.

#### 4.1.2. SR Domain Ingress Node

The SR Domain Ingress Node is the node where ingress policies are applied and where the packet path (and processing) is determined.

After policies are applied and packet classification is done, the result may be instantiated into a Segment List representing the path the packet should take. In such case, the SR Domain Ingress Node instantiate a new outer IPv6 header to which the SRH is appended (with the computed Segment List). The procedures for the creation and insertion of the new SRH are described in Section 4.1.1.

#### 4.1.3. Transit Node

According to [RFC2460], the only node who is 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 towards the DA and according to the IPv6 routing table.

In the example case described in Section 2.2.2, when SR capable nodes are connected through an overlay spanning multiple third-party infrastructure, it is safe to send SRH packets (i.e.: packet having a Segment Routing Header) between each other overlay/SR-capable nodes as long as the segment list does not include any of the transit provider nodes. In addition, as a generic security measure, any service provider will block any packet destined to one of its internal routers, especially if these packets have an extended header in it.

#### 4.1.4. SR Segment Endpoint Node

The SR segment endpoint node is the node whose address is in the DA. The segment endpoint node inspects the SRH and does:



1. IF DA = myself (segment endpoint)
2. IF Segments Left > 0 THEN  
    decrement Segments Left  
    update DA with Segment List[Segments Left]
3. IF Segments Left == 0 THEN  
    IF Clean-up bit is set THEN remove the SRH
4. ELSE give the packet to next PID (application)  
    End of processing.
5. Forward the packet out

## 5. Security Considerations

This section analyzes the security threat model, the security issues and mitigation techniques of SRH.

SRH is simply another type of the routing header as described in RFC 2460 [RFC2460] and is:

- o added to a new outer IP header by the ingress router when entering the SR domain or by the originating node itself. The source host can be outside the SR domain;
- o inspected and acted upon when reaching the destination address of the IP header per RFC 2460 [RFC2460].

Per RFC2460 [RFC2460], routers on the path that simply forward an IPv6 packet (i.e. the IPv6 destination address is none of theirs) will never inspect and process the content of any routing header (including SRH). Routers whose one interface IPv6 address equals the destination address field of the IPv6 packet MUST to parse the SRH and, if supported and if the local configuration allows it, MUST act accordingly to the SRH content.

According to RFC2460 [RFC2460], non SR-capable (or non SR-configured) router upon receipt of an IPv6 packet with SRH destined to an address of its:

- o must ignore the SRH completely if the Segment Left field is 0 and proceed to process the next header in the IPv6 packet;
- o must discard the IPv6 packet if Segment Left field is greater than 0 and send a Parameter Problem ICMP message back to the Source Address.

## 5.1. Threat model

### 5.1.1. Source routing threats

Using a SRH is a specific case of loose source routing, therefore it has some well-known security issues as described in RFC4942 [RFC4942] section 2.1.1 and RFC5095 [RFC5095]:

- o amplification attacks: where a packet could be forged in such a way to cause looping among a set of SR-enabled routers causing unnecessary traffic, hence a Denial of Service (DoS) against bandwidth;
- o reflection attack: where a hacker could force an intermediate node to appear as the immediate attacker, hence hiding the real attacker from naive forensic;
- o bypass attack: where an intermediate node could be used as a stepping stone (for example in a De-Militarized Zone) to attack another host (for example in the datacenter or any back-end server).

### 5.1.2. Applicability of RFC 5095 to SRH

First of all, the reader must remember this specific part of section 1 of RFC5095 [RFC5095], "A side effect is that this also eliminates benign RH0 use-cases; however, such applications may be facilitated by future Routing Header specifications.". In short, it is not forbidden to create new secure type of Routing Header; for example, RFC 6554 (RPL) [RFC6554] also creates a new Routing Header type for a specific application confined in a single network.

The main use case for SR consists of the single administrative domain (or cooperating administrative domains) where only trusted nodes with SR enabled and explicitly configured participate in SR: this is the same model as in RFC6554 [RFC6554]. All non-trusted nodes do not participate as either SR processing is not enabled by default or because they only process SRH from nodes within their domain.

Moreover, all SR routers SHOULD ignore SRH created by outsiders based on topology information (received on a peering or internal interface) or on presence and validity of the HMAC field. Therefore, if intermediate SR routers ONLY act on valid and authorized SRH (such as within a single administrative domain), then there is no security threat similar to RH-0. Hence, the RFC 5095 [RFC5095] attacks are not applicable.

### 5.1.3. Service stealing threat

Segment routing is used for added value services, there is also a need to prevent non-participating nodes to use those services; this is called 'service stealing prevention'.

### 5.1.4. Topology disclosure

The SRH may also contains IPv6 addresses of some intermediate SR routers in the path towards the destination, this obviously reveals those addresses to the potentially hostile attackers if those attackers are able to intercept packets containing SRH. On the other hand, if the attacker can do a traceroute whose probes will be forwarded along the SR path, then there is little learned by intercepting the SRH itself. The clean-bit of SRH can help by removing the SRH before forwarding the packet to potentially a non-trusted part of the network; if the attacker can force the generation of an ICMP message during the transit in the SR domain, then the ICMP will probably contain the SRH header (totally or partially) depending on the ICMP-generating router behavior.

### 5.1.5. ICMP Generation

Per section 4.4 of RFC2460 [RFC2460], when destination nodes (i.e. where the destination address is one of theirs) receive a Routing Header with unsupported Routing Type, the required behavior is:

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

This required behavior could be used by an attacker to force the generation of ICMP message by any node. The attacker could send packets with SRH (with Segment Left different than 0) destined to a node not supporting SRH. Per RFC2460 [RFC2460], the destination node must then generate an ICMP message per RFC 2460, causing a local CPU utilization and if the source of the offending packet with SRH was spoofed could lead to a reflection attack without any amplification.

It must be noted that this is a required behavior for any unsupported Routing Type and not limited to SRH packets. So, it is not specific to SRH and the usual rate limiting for ICMP generation is required anyway for any IPv6 implementation and has been implemented and deployed for many years.

## 5.2. Security fields in SRH

This section summarizes the use of specific fields in the SRH. They are based on a key-hashed message authentication code (HMAC).

The security-related fields in SRH are:

- o HMAC Key-id, 8 bits wide;
- o HMAC, 256 bits wide (optional, exists only if HMAC Key-id is not 0).

The HMAC field is the output of the HMAC computation (per RFC 2104 [RFC2104]) using a pre-shared key and hashing algorithm identified by HMAC Key-id and of the text which consists of the concatenation of:

- o the source IPv6 address;
- o First Segment field;
- o an octet whose bit-0 is the clean-up bit flag and others are 0;
- o HMAC Key-id;
- o all addresses in the Segment List.

The purpose of the HMAC field is to verify the validity, the integrity and the authorization of the SRH itself. If an outsider of the SR domain does not have access to a current pre-shared secret, then it cannot compute the right HMAC field and the first SR router on the path processing the SRH and configured to check the validity of the HMAC will simply reject the packet.

The HMAC field is located at the end of the SRH simply because only the router on the ingress of the SR domain needs to process it, then all other SR nodes can ignore it (based on local policy) because they trust the upstream router. This is to speed up forwarding operations because SR routers which do not validate the SRH do not need to parse the SRH until the end.

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. This allows for pre-shared key roll-over when two pre-shared keys are supported for a while when all SR nodes converged to a fresher pre-shared key. The HMAC Key-id field is opaque, i.e., it has neither syntax nor semantic except as an index to the right combination of pre-shared key and hash algorithm and except that a value of 0 means that there is no HMAC field. It could

also allow for interoperation among different SR domains if allowed by local policy and assuming a collision-free Key Id allocation which is out of scope of this memo.

When a specific SRH is linked to a time-related service (such as turbo-QoS for a 1-hour period), then it is important to refresh the shared-secret frequently as the HMAC validity period expires only when the HMAC Key-id and its associated shared-secret expires.

#### 5.2.1. Selecting a hash algorithm

The HMAC field in the SRH is 256 bits wide. Therefore, the HMAC MUST be based on a hash function whose output is at least 256 bits. If the output of the hash function is 256, then this output is simply inserted in the HMAC field. If the output of the hash function is larger than 256 bits, then the output value is truncated to 256 by taking the least-significant 256 bits and inserting them in the HMAC field.

SRH implementations can support multiple hash functions but MUST implement SHA-2 [FIPS180-4] in its SHA-256 variant.

NOTE: SHA-1 is currently used by some early implementations used for quick interoperations testing, the 160-bit hash value must then be right-hand padded with 96 bits set to 0. The authors understand that this is not secure but is ok for limited tests.

#### 5.2.2. Performance impact of HMAC

While adding a HMAC to each and every SR packet increases the security, it has a performance impact. Nevertheless, it must be noted that:

- o the HMAC field SHOULD be used only when SRH is inserted by a device (such as a home set-up box) which is outside of the segment routing domain. If the SRH is added by a router in the trusted segment routing domain, then, there is no need for a HMAC field, hence no performance impact.
- o when present, the HMAC field MUST be checked and validated only by the first router of the segment routing domain, this router is named 'validating SR router'. Downstream routers may not inspect the HMAC field.
- o this validating router can also have a cache of <IPv6 header + SRH, HMAC field value> to improve the performance. It is not the same use case as in IPsec where HMAC value was unique per packet, in SRH, the HMAC value is unique per flow.

- o Last point, hash functions such as SHA-2 have been optimized for security and performance and there are multiple implementations with good performance.

With the above points in mind, the performance impact of using HMAC is minimized.

### 5.2.3. Pre-shared key management

The field HMAC Key-id allows for:

- o key roll-over: when there is a need to change the key (the hash pre-shared secret), then multiple pre-shared keys can be used simultaneously. The validating routing can have a table of <HMAC Key-id, pre-shared secret, hash algorithm> for the currently active and future keys.
- o different algorithm: by extending the previous table to <HMAC Key-id, hash function, pre-shared secret>, the validating router can also support simultaneously several hash algorithms (see section Section 5.2.1)

The pre-shared secret distribution can be done:

- o in the configuration of the validating routers, either by static configuration or any SDN oriented approach;
- o dynamically using a trusted key distribution such as [RFC6407]

The intent of this document is NOT to define yet-another-key-distribution-protocol.

## 5.3. Deployment Models

### 5.3.1. Nodes within the SR domain

The routers inside a SR domain can be trusted to generate the outer IP header and the SRH and to process SRH received on interfaces that are part of the SR domain. These nodes MUST drop all SRH packets received on any interface that is not part of the SR domain and containing a SRH whose HMAC field cannot be validated by local policies. This includes obviously packet with a SRH generated by a non-cooperative SR domain.

If the validation fails, then these packets MUST be dropped, ICMP error messages (parameter problem) SHOULD be generated (but rate limited) and SHOULD be logged.

### 5.3.2. Nodes outside of the SR domain

Nodes outside of the SR domain cannot be trusted for physical security; hence, they need to obtain by some trusted means (outside of the scope of this document) a complete SRH for each new connection (i.e. new destination address). The received SRH MUST include a HMAC Key-id and HMAC field which has been computed correctly (see Section 5.2).

When a outside the SR domain sends a packet with a SRH and towards a SR domain ingress node, the packet MUST contain the HMAC Key-id and HMAC field and the destination address MUST be an address of a SR domain ingress node .

The ingress SR router, i.e., the router with an interface address equals to the destination address, MUST verify the HMAC field with respect to the HMAC Key-id.

If the validation is successful, then the packet is simply forwarded as usual for a SR packet. As long as the packet travels within the SR domain, no further HMAC check needs to be done. Subsequent routers in the SR domain MAY verify the HMAC field when they process the SRH (i.e. when they are the destination).

If the validation fails, then this packet MUST be dropped, an ICMP error message (parameter problem) SHOULD be generated (but rate limited) and SHOULD be logged.

### 5.3.3. SR path exposure

As the intermediate SR nodes addresses appears in the SRH, if this SRH is visible to an outsider then he/she could reuse this knowledge to launch an attack on the intermediate SR nodes or get some insider knowledge on the topology. This is especially applicable when the path between the source node and the first SR domain ingress router is on the public Internet.

The first remark is to state that 'security by obscurity' is never enough; in other words, the security policy of the SR domain SHOULD assume that the internal topology and addressing is known by the attacker.

IPsec Encapsulating Security Payload [RFC4303] cannot be use to protect the SRH as per RFC4303 the ESP header must appear after any routing header (including SRH).

When the SRH is not generated by the actual source node but by an SR domain ingress router, it is added after a new outer IP header, this

means that a normal traceroute will not reveal the routers in the SR domain (pretty much like in a MPLS network) and that if ICMP are generated by routers in the SR domain they will be sent to the ingress router of the SR domain without revealing anything to the outside of the SR domain.

To prevent a user to leverage the gained knowledge by intercepting SRH, it is recommended to apply an infrastructure Access Control List (iACL) at the edge of the SR domain. This iACL will drop all packets from outside the SR-domain whose destination is any address of any router inside the domain. This security policy should be tuned for local operations.

#### 5.3.4. Impact of BCP-38

BCP-38 [RFC2827], also known as "Network Ingress Filtering", checks whether the source address of packets received on an interface is valid for this interface. The use of loose source routing such as SRH forces packets to follow a path which differs from the expected routing. Therefore, if BCP-38 was implemented in all routers inside the SR domain, then SR packets could be received by an interface which is not expected one and the packets could be dropped.

As a SR domain is usually a subset of one administrative domain, and as BCP-38 is only deployed at the ingress routers of this administrative domain and as packets arriving at those ingress routers have been normally forwarded using the normal routing information, then there is no reason why this ingress router should drop the SRH packet based on BCP-38. Routers inside the domain commonly do not apply BCP-38; so, this is not a problem.

#### 6. IANA Considerations

TBD but should at least require a new type for routing header

#### 7. Manageability Considerations

TBD should we talk about traceroute? about SRH in ICMP replies?

#### 8. Contributors

The authors would like to thank Dave Barach, John Leddy, John Brzozowski, Pierre Francois, Nagendra Kumar, Mark Townsley, Christian Martin, Roberta Maglione, James Connolly, Aloys Augustin and Fred Baker for their contribution to this document.



## 9. Acknowledgements

TBD

## 10. References

### 10.1. 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>>.

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

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <<http://www.rfc-editor.org/info/rfc2460>>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<http://www.rfc-editor.org/info/rfc4303>>.

[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, <<http://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, <<http://www.rfc-editor.org/info/rfc6407>>.

### 10.2. Informative References

#### [I-D.ietf-isis-segment-routing-extensions]

Previdi, S., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-05 (work in progress), June 2015.

- [I-D.ietf-ospf-ospfv3-segment-routing-extensions]  
Psenak, P., Previdi, S., Filsfils, C., Gredler, H.,  
Shakir, R., Henderickx, W., and J. Tantsura, "OSPFv3  
Extensions for Segment Routing", draft-ietf-ospf-ospfv3-  
segment-routing-extensions-03 (work in progress), June  
2015.
- [I-D.ietf-spring-ipv6-use-cases]  
Brzozowski, J., Leddy, J., Leung, I., Previdi, S.,  
Townsend, W., Martin, C., Filsfils, C., and R. Maglione,  
"IPv6 SPRING Use Cases", draft-ietf-spring-ipv6-use-  
cases-05 (work in progress), September 2015.
- [I-D.ietf-spring-problem-statement]  
Previdi, S., Filsfils, C., Decraene, B., Litkowski, S.,  
Horneffer, M., and R. Shakir, "SPRING Problem Statement  
and Requirements", draft-ietf-spring-problem-statement-04  
(work in progress), April 2015.
- [I-D.ietf-spring-resiliency-use-cases]  
Francois, P., Filsfils, C., Decraene, B., and R. Shakir,  
"Use-cases for Resiliency in SPRING", draft-ietf-spring-  
resiliency-use-cases-01 (work in progress), March 2015.
- [I-D.ietf-spring-segment-routing]  
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S.,  
and r. rjs@rob.sh, "Segment Routing Architecture", draft-  
ietf-spring-segment-routing-05 (work in progress),  
September 2015.
- [I-D.ietf-spring-segment-routing-mpls]  
Filsfils, C., Previdi, S., Bashandy, A., Decraene, B.,  
Litkowski, S., Horneffer, M., Shakir, R., Tantsura, J.,  
and E. Crabbe, "Segment Routing with MPLS data plane",  
draft-ietf-spring-segment-routing-mpls-01 (work in  
progress), May 2015.
- [RFC1940] Estrin, D., Li, T., Rekhter, Y., Varadhan, K., and D.  
Zappala, "Source Demand Routing: Packet Format and  
Forwarding Specification (Version 1)", RFC 1940,  
DOI 10.17487/RFC1940, May 1996,  
<<http://www.rfc-editor.org/info/rfc1940>>.
- [RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-  
Hashing for Message Authentication", RFC 2104,  
DOI 10.17487/RFC2104, February 1997,  
<<http://www.rfc-editor.org/info/rfc2104>>.

- [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, <<http://www.rfc-editor.org/info/rfc2827>>.
- [RFC4942] Davies, E., Krishnan, S., and P. Savola, "IPv6 Transition/ Co-existence Security Considerations", RFC 4942, DOI 10.17487/RFC4942, September 2007, <<http://www.rfc-editor.org/info/rfc4942>>.
- [RFC6554] Hui, J., Vasseur, JP., Culler, D., and V. Manral, "An IPv6 Routing Header for Source Routes with the Routing Protocol for Low-Power and Lossy Networks (RPL)", RFC 6554, DOI 10.17487/RFC6554, March 2012, <<http://www.rfc-editor.org/info/rfc6554>>.

## Authors' Addresses

Stefano Previdi (editor)  
Cisco Systems, Inc.  
Via Del Serafico, 200  
Rome 00142  
Italy

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

Clarence Filsfils  
Cisco Systems, Inc.  
Brussels  
BE

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

Brian Field  
Comcast  
4100 East Dry Creek Road  
Centennial, CO 80122  
US

Email: [Brian\\_Field@cable.comcast.com](mailto:Brian_Field@cable.comcast.com)

Ida Leung  
Rogers Communications  
8200 Dixie Road  
Brampton, ON L6T 0C1  
CA

Email: [Ida.Leung@rci.rogers.com](mailto:Ida.Leung@rci.rogers.com)

Jen Linkova  
Google  
1600 Amphitheatre Parkway  
Mountain View, CA 94043  
US

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

Ebben Aries  
Facebook  
US

Email: [exa@fb.com](mailto:exa@fb.com)

Tomoya Kosugi  
NTT  
3-9-11, Midori-Cho Musashino-Shi,  
Tokyo 180-8585  
JP

Email: [kosugi.tomoya@lab.ntt.co.jp](mailto:kosugi.tomoya@lab.ntt.co.jp)

Eric Vyncke  
Cisco Systems, Inc.  
De Kleetlaann 6A  
Diegem 1831  
Belgium

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

David Lebrun  
Universite Catholique de Louvain  
Place Ste Barbe, 2  
Louvain-la-Neuve, 1348  
Belgium

Email: david.lebrun@uclouvain.be

MIF Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: April 17, 2016

M. Stenberg  
S. Barth  
Independent  
October 15, 2015

Multiple Provisioning Domains using Domain Name System  
draft-stenberg-mif-mpvd-dns-00

Abstract

This document describes a mechanism to transmit and secure provisioning domain information for IPv6 and IPv4 addresses by using reverse DNS resolution. In addition it specifies backwards-compatible extensions to IPv6 host configuration to support special-purpose global IPv6 prefixes which can only be used to access certain isolated services.

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 April 17, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
2.1. Requirements Language . . . . .	3
3. PVD information discovery using DNS . . . . .	3
3.1. PVD TXT Record Fomat . . . . .	4
3.2. PVD TXT Record Keys . . . . .	4
3.2.1. Reachable Services . . . . .	4
3.2.2. DNS Configuration . . . . .	5
3.2.3. Connectivity Characteristics . . . . .	5
3.2.4. Private Extensions . . . . .	6
4. Special-purpose IPv6 prefixes . . . . .	6
4.1. Extensions to Stateless Address Autoconfiguration . . . . .	6
4.2. Extensions to DHCPv6 . . . . .	7
5. Security Considerations . . . . .	9
6. IANA Considerations . . . . .	9
7. References . . . . .	10
7.1. Normative references . . . . .	10
7.2. Informative references . . . . .	11
Appendix A. This solution compared to doing this in DHCPv6/NDP [RFC                      Editor: please remove] . . . . .	11
Appendix B. Discussion Points [RFC Editor: please remove] . . . . .	12
Appendix C. Changelog [RFC Editor: please remove] . . . . .	13
Appendix D. Draft Source [RFC Editor: please remove] . . . . .	13
Appendix E. Acknowledgements . . . . .	13
Authors' Addresses . . . . .	13

## 1. Introduction

Given multiple address prefixes or multiple interfaces, hosts require more information to make educated choices about the interfaces and addresses to use. [RFC7556] describes the provision domains (PVDs) that provide the additional information the hosts need.

This document describes where and how the provision domain information is encoded in the Domain Name System (DNS). For optional authentication DNSSEC is used.

A backwards compatible way of adding IPv6 prefixes without generic internet connectivity is also provided so that the hosts that are not aware of the provisioning domain prefixes do not inadvertently use those for general network access.

## 2. Terminology

PVD	a provisioning domain, usually with a set of provisioning domain information; for more information, see [RFC7556].
special-purpose IPv6 prefix	a global IPv6 source prefix that cannot be used to reach the public IPv6 internet but instead only allows access to certain special services (e.g., VoIP, IPTV).

## 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 RFC 2119 [RFC2119].

### 3. PVD information discovery using DNS

Each PVD is stored within the DNS, encoded as a single TXT record [RFC1035]. Section 3.1 describes the syntax and Section 3.2 the semantics of the enclosed data.

To find the per-PVD TXT records that apply to a source address, the host queries the DNS for PTR records of the domain `_pvd.<domain>`. `<domain>` is a .arpa domain for reverse lookups derived from the respective prefix or subnet the source address is assigned from and generated as specified in [RFC3596] for IPv6 addresses and [RFC1034] for IPv4 addresses respectively. If the query returned any PTR records the host then subsequently queries the DNS for TXT records located in each domain indicated in the PTR records and handles their contents as individual PVDs.

As prefixes can be sub-delegated arbitrarily, PTR records SHOULD be provided for any subprefixes contained within a particular prefix. For example, given a prefix 2001:db8:8765:4321::/64, a host with an address of 2001:db8:8765:4321:1234:5678:abcd:beef queries for PTR records in `_pvd.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.2.3.4.5.6.7.8.8.b.d.0.1.0.0.2.ip6.arpa`. However, if the address is assigned from 2001:db8:8765:4321:1234::/80, the request would be made for `_pvd.0.0.0.0.0.0.0.0.0.0.0.0.0.4.3.2.1.1.2.3.4.5.6.7.8.8.b.d.0.1.0.0.2.ip6.arpa` instead.

If for example, the retrieved PTR record is assumed to point at the FQDN `_pvd.example.com`. The next query is then sent for TXT records in this domain, and if successful, the node has retrieved up-to-date information about PVDs applicable to that particular address.



A host MUST support handling multiple PTR records for the initial .arpa domain as well as multiple TXT records for all domains pointed to by the PTR records. This facilitates handling of multiple PVDs with minimal amount of state in the network. A host MUST honor both the time-to-live of the received records, and negative replies that conform to [RFC2308]. A host MUST NOT use addresses from a prefix as the source for new packet flows once the TTL has passed until it did successfully retrieve updated PVD information.

### 3.1. PVD TXT Record Fomat

PVD information within DNS is encoded using TXT records, similar to those of DNS-SD [RFC6763] and defined as follows. TXT records consist of key/value pairs, each encoded as a string of up to 255 octets preceded by a length byte storing the number of octets. The strings are in the form "key=value" or simply "key" (without quotation marks) where everything up to the first '=' character (if any, otherwise the whole string) is the key and everything after it (if any, including subsequent '=' characters) is the value. Due to the use of a length byte, quotation marks or similar are not required around the value. Keys are case-sensitive. Hosts MUST ignore TXT records which do not conform to these rules.

### 3.2. PVD TXT Record Keys

The following keys are defined to be used inside PVD TXT records. Unknown keys inside PVD Information MUST be ignored.

#### 3.2.1. Reachable Services

The following set of keys can be used to specify the set of services for which the respective PVD should be used. If present they MUST be honored by the client, i.e., if the PVD is marked as not usable for internet access it MUST NOT be used for internet access, if the usability is limited to a certain set of domain or address prefixes, then a different PVD MUST be used for other destinations.

Key	Description	Value	Example
n	User-visible service name	human-readable UTF-8 string	n=Fooobar Service
s	Internet inaccessible	(none)	s
z	DNS zones accessible	comma-separated list of DNS zone	z=foo.com,sub.bar.com
6	IPv6-prefixes accessible	comma-separated list of IPv6 prefixes	6=2001:db8:a::/48,2001:db8:b:c::/64
4	IPv4-prefixes accessible	comma-separated list of IPv4 prefixes in CIDR	4=1.2.3.0/24,2.3.0.0/16

### 3.2.2. DNS Configuration

The following set of keys can be used to specify the DNS configuration for the respective PVD. If present, they MUST be honored and used by the client whenever it wishes to access a resource of the PVD.

Key	Description	Value	Example
r	Recursive DNS server	comma-separated list of IPv6 and IPv4 addresses	r=2001:db8::1,192.0.2.2
d	DNS search domains	comma-separated list of search domains	d=foo.com,sub.bar.com

### 3.2.3. Connectivity Characteristics

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

Key	Description	Value	Example
bw	Maximum achievable bandwidth	1 symmetrical or 2 comma-separated ingress, egress values in kilobits per second	bw=5000 or bw=1000,100
lt	Minimum achievable latency	1 symmetrical or 2 comma-separated ingress, egress values in milliseconds	lt=20 or lt=20,100
rl	Maximum achievable reliability	1 symmetrical or 2 comma-separated ingress, egress values in 1/1000	rl=1000 or rl=900,800
tm	Traffic metered (cut-off / limited over threshold)	(none) or traffic threshold in kilobytes	tm or tm=1000000
cp	Captive portal	(none)	cp
nat	IPv4 NAT in place	(none)	nat

#### 3.2.4. Private Extensions

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

### 4. Special-purpose IPv6 prefixes

A service provider might wish to assign certain global unicast address prefixes which can be used to reach a limited set of services only. In the presence of legacy hosts it must be ensured however that these prefixes are not mistakenly used as source addresses for other destinations. This section therefore defines optional extensions to NDP [RFC4861], DHCPv6 [RFC3315] and DHCPv6-PD [RFC3633] to indicate this state.

#### 4.1. Extensions to Stateless Address Autoconfiguration

NDP [RFC4861] defines the Prefix Information option to announce prefixes for stateless address configuration. The "A-bit" is set, whenever hosts may autonomously derive addresses from a given prefix. For special-purpose prefixes this document defines the first bit of the Reserved1-field as the "S-Bit".

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               Prefix Length    L A S Reserved1										Type                       Length               Prefix Length    L A S Reserved1										Type                       Length               Prefix Length    L A S Reserved1										Type                       Length               Prefix Length    L A S Reserved1									
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+									
										...																													

The following additional requirements apply to hosts intending to support global special-purpose IPv6 prefixes:

Upon reception of a Prefix Information Option with the S-Bit set, it should behave as if the A-Bit was set, however (unless the A-Bit was actually set by the sending router) it MUST delay using any addresses derived from the prefix until it has queried, retrieved and honored (see Section 3) at least all mandatory provisioning domain information related to the given prefix.

A host SHOULD NOT interpret the S-Bit being clear as an indicator that no provisioning domain information is available for a given prefix.

The following additional requirements apply to routers:

A router MUST NOT set the A-Bit for global unicast address prefixes which cannot be used to reach the public IPv6 internet.

A router SHOULD use the S-Bit to indicate that PVD-aware hosts can statelessly assign themselves addresses from a given prefix. It MAY use the S-Bit in addition to the A-Bit to indicate that a prefix usable to reach the public IPv6 internet has additional (optional) provisioning domain information.

A router announcing one or more Prefix Information options with the S-Bit set MUST also announce one or more recursive DNS servers using a Recursive DNS Server Option [RFC6106]. If none of the Prefix Information Options it announces have the A-Bit set then at least one of these recursive DNS servers MUST be reachable using a link-local address.

#### 4.2. Extensions to DHCPv6

Using DHCPv6 [RFC3315] and DHCPv6-PD [RFC3633] servers can actively decide which addresses or prefixes are assigned to clients and requesting routers, however a mechanism is needed to distinguish PVD-aware devices and in the same sense PVD-aware devices need to be able to detect which prefixes and addresses are special-purpose. Therefore, a zero-length DHCPv6 option `OPTION_SPECIAL_PURPOSE` is

defined to be added as a suboption to `OPTION_IAADDR` and `OPTION_IAPREFIX` options.

```

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
+-----+-----+-----+-----+-----+-----+-----+-----+
| OPTION_SPECIAL_PURPOSE (TBD) | 0 |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

The following additional requirements apply to clients and requesting routers intending to support global special-purpose IPv6 prefixes via DHCPv6:

A client or requesting router **MUST** include the option code `OPTION_SPECIAL_PURPOSE` in an option `OPTION_ORO` in its `SOLICIT` and `REQUEST` messages, whenever it wishes to accept special-purpose prefixes in its identity associations.

Upon reception of an `OPTION_IAADDR` or `OPTION_IAPREFIX` option having an embedded `OPTION_SPECIAL_PURPOSE` option it **MUST** delay using any addresses derived from the prefix as source address for its own communication until it has queried, retrieved and honored (see Section 3) at least all mandatory provisioning domain information related to the given prefix or address. If it is a requesting router, it **MAY** however subdelegate prefixes or assign addresses from special-purpose prefixes to clients without doing so as long as the requirements in the following paragraph are honored.

The following additional requirements apply to routers assigning addresses from or delegating (parts of) special-purpose prefixes using DHCPv6:

A router **MUST** include a zero-length suboption of type `OPTION_SPECIAL_PURPOSE` in every `OPTION_IAADDR` and `OPTION_IAPREFIX` option it assigns or delegates containing a global unicast address or prefix which cannot be used to reach the public IPv6 internet. It **MUST NOT** assign or delegate such an address or prefix to a client or requesting router not including the option code of `OPTION_SPECIAL_PURPOSE` inside an `OPTION_ORO` option.

A router announcing one or more addresses or prefixes with an embedded `OPTION_SPECIAL_PURPOSE` option **MUST** also announce one or more recursive DNS servers using a `OPTION_DNS_SERVERS` option [RFC3646]. If all of the addresses in a DHCPv6 reply carry the embedded `OPTION_SPECIAL_PURPOSE` option then at least one of the announced recursive DNS servers **MUST** be reachable using a link-local address.

## 5. Security Considerations

The security implications of using PVDs in general depend on two factors: what they are allowed to do, and whether or not the authentication and authorization of the PVD information received is sufficient for the particular usecase. As the defined scheme uses DNS for retrieval of the connection parameters, the retrieval of both the PTR and the TXT records should be secured, if they are to be trusted. The PVD information allows for the following types of attacks:

- o Traffic redirection, both by providing custom DNS server, as well as actual potentially different next-hop and/or source address selection.
- o Faking of connection capabilities to e.g. prefer some connection fraudulently over others.

If a host requires DNSSEC authentication and the retrieved information is not sufficiently secured, they **MUST** be ignored as the defined way of using them in Section 3.2 requires honoring the supplied information.

Security properties of NDP and DHCPv6 are inherited for the respective extensions, therefore relevant sections of [RFC4861] and [RFC3315] should be consulted. In any case, signaling addresses and prefixes to be special-purpose does not have a significant impact on the underlying assignment and delegation mechanisms.

## 6. IANA Considerations

IANA is requested to setup a PVD DNS TXT Record Key registry with the initial types: s, z, 6, 4 (Section 3.2.1); r, d (Section 3.2.2); bw, lt, rl, tm, cp, nat (Section 3.2.3) and a prefix x- (Section 3.2.4) for Private Use [RFC5226]. The policy for further additions to the registry is requested to be RFC Required [RFC5226].

This document defines a new bit for the NDP Prefix Information Option (the S-bit). There is currently no registration procedure for such bits, so IANA should not take any action on this matter.

IANA should assign a DHCPv6 option code `OPTION_SPECIAL_PURPOSE` to the DHCPv6 option code space defined in [RFC3315].

## 7. References

### 7.1. Normative references

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<http://www.rfc-editor.org/info/rfc1034>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<http://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, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <<http://www.rfc-editor.org/info/rfc2308>>.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<http://www.rfc-editor.org/info/rfc4861>>.
- [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, <<http://www.rfc-editor.org/info/rfc3315>>.
- [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, DOI 10.17487/RFC3633, December 2003, <<http://www.rfc-editor.org/info/rfc3633>>.
- [RFC3646] Droms, R., Ed., "DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3646, DOI 10.17487/RFC3646, December 2003, <<http://www.rfc-editor.org/info/rfc3646>>.
- [RFC6106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 6106, DOI 10.17487/RFC6106, November 2010, <<http://www.rfc-editor.org/info/rfc6106>>.

- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <<http://www.rfc-editor.org/info/rfc5226>>.
- [RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <<http://www.rfc-editor.org/info/rfc7556>>.
- [RFC3596] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi, "DNS Extensions to Support IP Version 6", RFC 3596, DOI 10.17487/RFC3596, October 2003, <<http://www.rfc-editor.org/info/rfc3596>>.

## 7.2. Informative references

- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<http://www.rfc-editor.org/info/rfc6763>>.
- [PEN] IANA, "Private Enterprise Numbers", <<https://www.iana.org/assignments/enterprise-numbers>>.

Appendix A. This solution compared to doing this in DHCPv6/NDP [RFC Editor: please remove]

The angle of attack of the MIF work to date (autumn 2015) has been to add container options and their transfer mechanisms to DHCPv6 and NDP. This document details a different approach, and therefore it is sensible to compare it to the existing solutions in terms of (highly subjective) pros and cons. The authors consider pros of this proposal to be:

- o No overhead for hosts that do not care (possibly most; no spurious RA options, ...)
- o No problems with relaying data; if the first-hop device does not care, DNS requests propagate onward.
- o Little/no changes to DHCP, DHCPv6, DHCPv6-PD or RA.
- o Much more scalable; no worries about multicast packet size limits.
- o No duplication of specifications / TLVs for DHCP, DHCPv6 and RA.
- o Solves m:n prefix <-> PVD elegantly: no need to either duplicate applying prefix for each PVD or duplicate each PVD for each applying prefix.



- o Easily extensible (TXT records, no TLV definitions, parsing and generation necessary)
- o Probably not affected by IPR on draft-ietf-mif-mpvd-dhcp-support
- o Reuses the existing reverse DNS infrastructure

The authors consider cons of this proposal to be:

- o This scheme requires DNS servers 'close' on the path to the user, if changed information is to be sent; otherwise centralized solution would work (with some synthesized records).
- o Security using either DNSSEC or in-band hashes is rather painful (but possibly not more than the scheme in the current DHCP/RA drafts), so the default would most likely be insecure. That is not much different from DHCP\*/RA, which are also 99.999...% of the time not secured.

#### Appendix B. Discussion Points [RFC Editor: please remove]

- o Besides special purpose prefixes, it might be desirable to have special purpose routers which only provide access to certain services but not the entire internet. These services could be announced by only using more-specific routes, however if the destination addresses are possibly changing, extension of the RIO mechanism might be needed. One possibility would be to add a new RIO S-flag with semantics like: "When the host receives a Route Information Option with the S-Bit set, it MUST ignore the value in the Prf-field (even if it is (10)) and instead assume the preference to have a value greater than (11). However, it MUST only use the route for packets having a source prefix announced by the same router.". This would allow selective routes (Prf=(10)) only applying to MIF-hosts.
- o DNSSEC delegation of reverse zones might be difficult in some cases. It is debatable, whether we want a complementary in-band signing mechanism as well, e.g., pre-shared public keys associated the domain name of the TXT records and "sig-X=..." keys (where X identifies the specific key) and ... is an EdDSA or ECDSA signature over all records not starting with "sig-". Care would need to be taken wrt. TTL and negative caching though.
- o Should PVD-aware hosts be recommended or even required to always prefer routers that announced the respective source address in a PIO over those that didn't when making routing decisions? This takes on the points made in draft-baker-6man-multi-homed-host.

Appendix C. Changelog [RFC Editor: please remove]

draft-stenberg-mif-mpvd-dns-00:

- o Initial version.

Appendix D. Draft Source [RFC Editor: please remove]

As usual, this draft is available at <https://github.com/fingon/ietf-drafts/> in source format (with nice Makefile too). Feel free to send comments and/or pull requests if and when you have changes to it!

Appendix E. Acknowledgements

Thanks to Eric Vyncke for the original idea of using DNS for transmitting PVD information.

Authors' Addresses

Markus Stenberg  
Independent  
Helsinki 00930  
Finland

Email: markus.stenberg@iki.fi

Steven Barth  
Independent  
Halle 06114  
Germany

Email: cyrus@openwrt.org