Network Working Group
Internet-Draft
Intended status: Informational
Expires: 22 December 2024

J. M. Halpern, Ed.
Ericsson
J. Daley
IETF Administration LLC

20 June 2024

Antitrust Guidelines for IETF Participants
draft-halpern-gendispatch-antitrust-09

Abstract

This document provides education and guidance for IETF participants on compliance with antitrust laws and how to reduce antitrust risks in connection with IETF activities.

Status of This Memo

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

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

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

This Internet-Draft will expire on 22 December 2024.

Copyright Notice

Table of Contents

1. Introduction

Standards development frequently requires collaboration between competitors. Cooperation among competitors can spark concerns about antitrust law or competition law violations. This document is intended to educate IETF participants about how to reduce antitrust risks in connection with IETF activities. Nothing in this document changes existing IETF policies.

2. Background

2.1. A Note About Terminology

"Antitrust law" and "competition law" are used synonymously in this document. â\200\234Antitrustâ\200\235 is the word that is used in the US and in several other jurisdictions; â\200\234competition lawâ\200\235 is the terminology used in Europe and in many other jurisdictions. There can be some nuanced differences between how different jurisdictions address this general area of law, and sometimes people use the terminology differently to highlight these nuances, but here they are being used as synonyms.

2.2. Purpose of Antitrust or Competition law

The U.S. Department of Justice states [DOJ] that â\200\234the goal of the antitrust laws is to protect economic freedom and opportunity by promoting free and fair competition in the marketplace. Competition in a free market benefits consumers through lower prices, better quality and greater choice. Competition provides businesses the opportunity to compete on price and quality, in an open market and on

a level playing field, unhampered by anticompetitive restraints.â\200\235 Similarly, the European Commission [EC] states that the purpose of its competition law rules is "to make EU markets work better, by ensuring that all companies compete equally and fairly on their merits" which "benefits consumers, businesses and the European economy as a whole." Fundamentally, antitrust or competition laws are designed to facilitate open, fair, robust competition, ultimately to benefit consumers.

2.3. Overlapping Areas of Concern

There are two overlapping areas of concern the IETF has in connection with antitrust compliance:

* Most acutely, the IETF cannot have anyone who is officially representing the IETF, in any capacity, engage in anti-competitive behavior and create liability for the IETF.

* Additionally, the IETF cannot be a forum where participants engage in anti-competitive behavior, even if direct liability for that behavior falls on those participants and not the IETF, to avoid reputational harm to the IETF.

3. Existing IETF Antitrust Compliance Strategy

Compliance with the BCPs and other relevant policies that document the established rules and norms of the IETF, facilitates compliance with antitrust law, as the IETF structure and processes are designed to mitigate antitrust risks. As a reminder, participants are required to comply with the following policies:

* The Internet Standards Process as described in BCP 9 [BCP9], which is designed to "provide a fair, open, and objective basis for developing, evaluating, and adopting Internet Standards," and provides robust procedural rules, including an appeals process.

* The Working Group Guidelines and Procedures described in BCP 25 [BCP25], which emphasize requirements for "open and fair participation and for thorough consideration of technical alternatives," and describe IETF's consensus-based decision-making processes.

* The IETF framework that participants engage in their individual capacity, not as company representatives (see [BCP9] and [LLC]), and "use their best engineering judgment to find the best solution for the whole Internet, not just the best solution for any particular network, technology, vendor, or user," as described in BCP 54 [BCP54] .

* The IETF's intellectual property rights policies as set forth in BCP 78 [BCP78] and BCP 79 [BCP79]. These policies are carefully designed to "benefit the Internet community and the public at large, while respecting the legitimate rights of others."

* The established conflict of interest policies, such as the IESG Conflict of Interest Policy, the IAB Conflict of Interest Policy or the IETF LLC Conflict of Interest Policy, if and when applicable.

4. Additional Recommendations

The most important recommendation is for IETF participants to rigorously follow all applicable IETF policies as set out in section 3 above.

This section provides more information about:

* Certain topics that are generally inappropriate for discussion in a standards setting environment.

* The importance of participants obtaining independent legal advice, as appropriate.

* Paths to escalate antitrust-related concerns.

4.1. Topics to Avoid

While IETF participants are expected to participate as individuals, their actions could still be construed as representing their employer, whatever their role. Therefore, participants should be aware that some topics are generally inappropriate for discussion in a standards setting environment where representatives from competitors to their employer are likely to be present. These topics include: discussion about product pricing or profit margins among potential competitors, the details of business relationships between specific vendors and customers, details about the supply chains of specific companies, discussions about market opportunities for specific companies, or employee compensation or benefits among potentially competitive employers. While not all discussions of these topics would necessarily be antitrust violations, and recognizing that analysis of antitrust considerations will be different for differently-positioned participants, prudence suggests that avoiding these specific topics in the context of the collaborative IETF process best mitigates antitrust risks for the IETF and its participants.

Note that antitrust law reaches beyond these topics, however. For example, any behavior that amounts to an agreement to restrain marketplace competition, or that facilitates monopolization of particular markets, raises potential antitrust risks. Participants are responsible for ensuring that their conduct does not violate any antitrust laws or regulations.

4.2. Obtaining Independent Legal Advice

All IETF participants are expected to behave lawfully when engaged in IETF activities, including by following applicable antitrust law. The IETF does not provide legal advice to participants, and instead recommends that participants obtain independent legal advice as needed.

4.3. Escalating Antitrust-Related Concerns

Participants can report potential antitrust issues in the context of IETF activities by contacting IETF legal counsel (legal@ietf.org) or via the IETF LLC whistleblower service [Whistleblower]. Note that reports will only be assessed for their impact upon the IETF; participants directly impacted by an antitrust issue are responsible for obtaining their own legal advice.

5. IANA Considerations

No values are assigned in this document, no registries are created, and there is no action assigned to the IANA by this document.

6. Security Considerations

This document introduces no known security aspects to the IETF or IETF participants.

7. Normative References

[BCP9] Best Current Practice 9, <https://www.rfc-editor.org/info/bcp9>. At the time of writing, this BCP comprises the following:

Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, DOI 10.17487/RFC2026, October 1996, <https://www.rfc-editor.org/info/rfc2026>.

Dusseault, L. and R. Sparks, "Guidance on Interoperation and Implementation Reports for Advancement to Draft Standard", BCP 9, RFC 5657, DOI 10.17487/RFC5657, September 2009, <https://www.rfc-editor.org/info/rfc5657>.

Housley, R., Crocker, D., and E. Burger, "Reducing the Standards Track to Two Maturity Levels", BCP 9, RFC 6410, DOI 10.17487/RFC6410, October 2011, <https://www.rfc-editor.org/info/rfc6410>.

Resnick, P., "Retirement of the "Internet Official Protocol Standards" Summary Document", BCP 9, RFC 7100, DOI 10.17487/RFC7100, December 2013, <https://www.rfc-editor.org/info/rfc7100>.

Kolkman, O., Bradner, S., and S. Turner, "Characterization of Proposed Standards", BCP 9, RFC 7127, DOI 10.17487/RFC7127, January 2014, <https://www.rfc-editor.org/info/rfc7127>.

Dawkins, S., "Increasing the Number of Area Directors in an IETF Area", BCP 9, RFC 7475, DOI 10.17487/RFC7475, March 2015, <https://www.rfc-editor.org/info/rfc7475>.

Halpern, J., Ed. and E. Rescorla, Ed., "IETF Stream Documents Require IETF Rough Consensus", BCP 9, RFC 8789, DOI 10.17487/RFC8789, June 2020, <https://www.rfc-editor.org/info/rfc8789>.

[BCP25] Best Current Practice 25, <https://www.rfc-editor.org/info/bcp25>. At the time of writing, this BCP comprises the following:

Bradner, S., "IETF Working Group Guidelines and Procedures", BCP 25, RFC 2418, DOI 10.17487/RFC2418, September 1998, <https://www.rfc-editor.org/info/rfc2418>.

Wasserman, M., "Updates to RFC 2418 Regarding the Management of IETF Mailing Lists", BCP 25, RFC 3934, DOI 10.17487/RFC3934, October 2004, <https://www.rfc-editor.org/info/rfc3934>.

Resnick, P. and A. Farrel, "IETF Anti-Harassment Procedures", BCP 25, RFC 7776, DOI 10.17487/RFC7776, March 2016, <https://www.rfc-editor.org/info/rfc7776>.

Resnick, P. and A. Farrel, "Update to the IETF Anti-Harassment Procedures for the Replacement of the IETF Administrative Oversight Committee (IAOC) with the IETF Administration LLC", BCP 25, RFC 8716, DOI 10.17487/RFC8716, February 2020, <https://www.rfc-editor.org/info/rfc8716>.

[BCP54] Best Current Practice 54,
<https://www.rfc-editor.org/info/bcp54>.
At the time of writing, this BCP comprises the following:

Moonesamy, S., Ed., "IETF Guidelines for Conduct", BCP 54,
RFC 7154, DOI 10.17487/RFC7154, March 2014,
<https://www.rfc-editor.org/info/rfc7154>.

[BCP78] Best Current Practice 78,
<https://www.rfc-editor.org/info/bcp78>.
At the time of writing, this BCP comprises the following:

Bradner, S., Ed. and J. Contreras, Ed., "Rights
Contributors Provide to the IETF Trust", BCP 78, RFC 5378,
DOI 10.17487/RFC5378, November 2008,
<https://www.rfc-editor.org/info/rfc5378>.

[BCP79] Best Current Practice 79,
<https://www.rfc-editor.org/info/bcp79>.
At the time of writing, this BCP comprises the following:

Bradner, S. and J. Contreras, "Intellectual Property
Rights in IETF Technology", BCP 79, RFC 8179,
DOI 10.17487/RFC8179, May 2017,
<https://www.rfc-editor.org/info/rfc8179>.

8. Informative References

[LLC] "IETF Administration LLC Statement on Competition Law
Issues", <https://www.ietf.org/blog/ietf-llc-statement-
competition-law-issues/>.

[DOJ] "The mission of the Antitrust Division",
<https://www.justice.gov/atr/mission>.

[EC] "Competition", <https://commission.europa.eu/about-
european-commission/departments-and-executive-agencies/
competition_en>.

[Whistleblower]
"IETF Administration LLC Whistleblower Policy",
<https://www.ietf.org/administration/policies-procedures/
whistleblower/>.

Authors' Addresses

Joel M. Halpern (editor)
Ericsson
P. O. Box 6049
Leesburg, VA 20178
United States of America
Email: joel.halpern@ericsson.com

Jay Daley
IETF Administration LLC
1000 N. West Street, Suite 1200
Wilimington, DE 19801
United States of America
Email: jay@staff.ietf.org

BESS Workgroup
Internet-Draft
Updates: 8365, 7432 (if approved)
Intended status: Standards Track
Expires: 9 January 2025

J. Rabadan, Ed.
K. Nagaraj
Nokia
W. Lin
Juniper
A. Sajassi
Cisco

8 July 2024

EVPN Multi-Homing Extensions for Split Horizon Filtering
draft-ietf-bess-evpn-mh-split-horizon-10

Abstract

Ethernet Virtual Private Network (EVPN) is commonly used with Network Virtualization Overlay (NVO) tunnels, as well as MPLS and Segment Routing tunnels. The multi-homing procedures in EVPN may vary based on the type of tunnel used within the EVPN Broadcast Domain. Specifically, there are two multi-homing Split Horizon procedures designed to prevent looped frames on multi-homed Customer Edge (CE) devices: the ESI Label-based procedure and the Local Bias procedure. The ESI Label-based Split Horizon is applied to MPLS-based tunnels, such as MPLSoUDP, while the Local Bias procedure is used for other tunnels, such as VXLAN.

Current specifications do not allow operators to choose which Split Horizon procedure to use for tunnel encapsulations that support both methods. Examples of tunnels that may support both procedures include MPLSoGRE, MPLSoUDP, GENEVE, and SRv6. This document updates the EVPN multi-homing procedures described in RFC 8365 and RFC 7432, enabling operators to select the appropriate Split Horizon procedure for a given tunnel based on their specific requirements.

Status of This Memo

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

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

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

This Internet-Draft will expire on 9 January 2025.

Table of Contents

1. Introduction

Ethernet Virtual Private Networks (EVPN) are commonly used with the following tunnel encapsulations:

* Network Virtualization Overlay (NVO) tunnels, where the EVPN procedures are specified in [RFC8365]. MPLSoGRE [RFC4023], MPLSoUDP [RFC7510], GENEVE [RFC8926] or VXLAN [RFC7348] tunnels are considered NVO tunnels.

* MPLS and Segment Routing with MPLS data plane (SR-MPLS), where the relevant EVPN procedures are specified in [RFC7432]. Segment Routing with MPLS data plane tunneling is specified in [RFC8660].

* Segment Routing with IPv6 data plane (SRv6), where the relevant EVPN procedures are specified in [RFC9252]. SRv6 is specified in [RFC8986].

The EVPN multihoming procedures may vary depending on the type of tunnel utilized within the EVPN Broadcast Domain. Specifically, there are two multihoming Split Horizon procedures employed to prevent looped frames on multihomed Customer Edge (CE) devices: the ESI Label-based procedure and the Local Bias procedure.

The ESI Label-based Split Horizon procedure is used for MPLS or MPLS-over-X (MPLSoX) tunnels, such as MPLS-over-UDP, and its procedures are detailed in [RFC7432]. Conversely, the Local Bias procedure is used for IP-based tunnels, such as VXLAN tunnels, and it is described in [RFC8365].

1.1. Conventions and Terminology

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

* AC: Attachment Circuit.

* A-D per ES route: refers to the EVPN Ethernet Auto-Discovery per ES route defined in [RFC7432].

* Arg.FE2: refers to the ESI filtering argument used for Split Horizon as specified in [RFC9252].

* Broadcast Domain (BD): an emulated ethernet, such that two systems on the same BD will receive each other's link-local broadcasts. In this document, BD also refers to the instantiation of a Broadcast Domain on an EVPN PE. An EVPN PE can be attached to one or multiple BDs of the same tenant.

* BUM: Broadcast, Unknown unicast and Multicast traffic.

* Designated Forwarder (DF): as defined in [RFC7432], an ES may be multihomed (attached to more than one PE). An ES may also contain multiple BDs, of one or more EVIs. For each such EVI, one of the PEs attached to the segment becomes that EVI's DF for that segment. Since a BD may belong to only one EVI, we can speak unambiguously of the BD's DF for a given segment.

* ES and ESI: Ethernet Segment and Ethernet Segment Identifier.

* EVI: EVPN Instance

* EVI-RT: EVI Route Target. A group of NVEs attached to the same EVI will share the same EVI-RT.

* GENEVE: Generic Network Virtualization Encapsulation, [RFC8926].

* MPLS and non-MPLS NVO tunnels: refer to Multi-Protocol Label Switching (or the absence of it) Network Virtualization Overlay tunnels. Network Virtualization Overlay tunnels use an IP encapsulation for overlay frames, where the source IP address identifies the ingress NVE and the destination IP address the egress NVE.

* MPLSoUDP: Multi-Protocol Label Switching over User Datagram Protocol, [RFC7510]

* MPLSoGRE: Multi-Protocol Label Switching over Generic Network Encapsulation, [RFC4023].

* MPLSoX: refers to MPLS over any IP encapsulation. Examples are MPLS-over-UDP or MPLS-over-GRE.

* NVE: Network Virtualization Edge device.

* NVGRE: Network Virtualization Using Generic Routing Encapsulation, [RFC7637].

* VXLAN: Virtual eXtensible Local Area Network, [RFC7348].

* VXLAN-GPE: VXLAN Generic Protocol Extension, [I-D.ietf-nvo3-vxlan-gpe].

* VNI: Virtual Network Identifier. A 24-bit identifier used by Network Virtualization Overlay (NVO) over IP encapsulations. Examples are VXLAN (Virtual Extended Local Area Network) or GENEVE (Generic Network Virtualization Encapsulation).

* SHT: Split Horizon Type, it refers to the Split Horizon method that a PE intends to use and advertises in an A-D per ES route.

* SR-MPLS: Segment Routing with an MPLS data plane, [RFC8660].

* SRv6: Segment routing with an IPv6 data plane, [RFC8986].

This document also assumes familiarity with the terminology of [RFC7432] and [RFC8365].

1.2. Split Horizon Filtering and Tunnel Encapsulations

EVPN supports two Split Horizon Filtering mechanisms:

* ESI Label based Split Horizon filtering [RFC7432]

  When EVPN is employed for MPLS transport tunnels, an MPLS label facilitates Split Horizon filtering to support All-Active multihoming. The ingress Network Virtualization Edge (NVE) device appends a label corresponding to the source Ethernet Segment Identifier (ESI label) during packet encapsulation. The egress NVE verifies the ESI label when attempting to forward a multi-destination frame through a local Ethernet Segment (ES) interface. If the ESI label matches the site identifier (ESI) associated with that ES interface, the packet is not forwarded. This mechanism effectively prevents forwarding loops for Broadcast, Unknown Unicast, and Multicast (BUM) traffic.

  The ESI Label Split Horizon filtering should also be utilized with Single-Active multihoming to prevent transient loops for in-flight packets when the egress NVE assumes the role of Designated Forwarder for an ES.

* Local Bias [RFC8365]

  Since IP tunnels, such as VXLAN or NVGRE, do not support the ESI label or any MPLS label, an alternative Split Horizon filtering procedure must be implemented for All-Active multihoming. This mechanism, known as Local Bias, relies on the source IP address of the tunnel to determine whether to forward Broadcast, Unknown Unicast, and Multicast (BUM) traffic to a local Ethernet Segment (ES) interface at the egress Network Virtualization Edge (NVE).

  In summary, each NVE tracks the IP address(es) of other NVEs with which it shares multihomed ESs. Upon receiving a BUM frame encapsulated in an IP tunnel, the egress NVE inspects the source IP address in the tunnel header, which identifies the ingress NVE. The egress NVE then filters out the frame on all local interfaces connected to ESs that are shared with the ingress NVE.

  Due to this behavior at the egress NVE, the ingress NVE is required to perform local replication to all directly attached ESs, regardless of the Designated Forwarder election state, for all BUM traffic ingressing from the access Attachment Circuits (ACs). This local replication at the ingress NVE is the basis for the term Local Bias.

Local Bias is not suitable for Single-Active multihoming, as the ingress NVE deactivates the ACs for which it is not the Designated Forwarder. Consequently, local replication to non-Designated Forwarder ACs cannot occur, leading to transient in-flight BUM packets to be looped back to the originating site by newly elected Designated Forwarder egress NVEs.

[RFC8365] specifies that Local Bias is exclusively utilized for IP tunnels, while ESI Label-based Split Horizon is employed for IP-based MPLS tunnels. However, IP-based MPLS tunnels, such as MPLS over GRE (MPLSoGRE) or MPLS over UDP (MPLSoUDP), are also categorized as IP tunnels and have the potential to support both procedures. These tunnels are capable of carrying ESI labels and also utilize a tunnel IP header in which the source IP address identifies the ingress Network Virtualization Edge (NVE).

Similarly, certain IP tunnels - that include an identifier for the source Ethernet Segment (ES) in the tunnel header - may also potentially support either procedure. Examples of such tunnels include GENEVE and SRv6.:

* In a GENEVE tunnel, the source IP address identifies the ingress NVE therefore local bias is possible. Also, [I-D.ietf-bess-evpn-geneve] defines an Ethernet option TLV (Type Length Value) to encode an ESI label value.

* In an SRv6 tunnel, the source IP address identifies the ingress NVE. By default, and as outlined in [RFC9252], the ingress PE adds specific information to the SRv6 packet to enable the egress PE to identify the source ES of the BUM packet. This information is the ESI filtering argument (Arg.FE2) [RFC9252][RFC8986] of the service Segment Identifier (SID) received on an A-D per ES route from the egress PE.

Table 1 presents various tunnel encapsulations along with their supported and default Split Horizon methods. For GENEVE, the default Split Horizon Type (SHT) is contingent upon the negotiation of the Ethernet Option with the Source ID TLV. In the case of SRv6, the default SHT is specified as ESI Label filtering in the table, as its behavior is analogous to that of ESI Label filtering. In this document, ESI Label filtering refers to the Split Horizon filtering based on the presence of a source Ethernet Segment (ES) identifier in the tunnel header.

This document classifies the tunnel encapsulations used by EVPN into:

1. IP-based MPLS tunnels

2. (SR-)MPLS tunnels

3. IP tunnels

4. SRv6 tunnels

Any tunnel encapsulation not listed in Table 1) that can be categorized into one of the four encapsulation groups mentioned above will support Split Horizon filtering based on the following rules:

* IP-based MPLS tunnels and SRv6 tunnels are capable of supporting both Split Horizon filtering methods.

* (SR-)MPLS tunnels only support ESI Label-based Split Horizon filtering

* IP tunnels support Local Bias Split Horizon filtering and may also support ESI Label-based Split Horizon filtering, provided they incorporate a mechanism to identify the source ESI in the header.

Tunnel Encapsulation	Default Split Horizon Type (SHT)	Supports Local Bias	Supports ESI Label
MPLSoGRE (IP-based MPLS)	ESI Label filtering	Yes	Yes
MPLSoUDP (IP-based MPLS)	ESI Label filtering	Yes	Yes
(SR-)MPLS	ESI Label filtering	No	Yes
VXLAN (IP tunnels)	Local Bias	Yes	No
NVGRE (IP tunnels)	Local Bias	Yes	No
VXLAN-GPE (IP tunnels)	Local Bias	Yes	No
GENEVE (IP tunnels)	Local Bias (no ESI Lb), ESI Label (if ESI lb)	Yes	Yes
SRv6	ESI Label filtering	Yes	Yes

Table 1: Tunnel Encapsulations and Split Horizon Types

The ESI Label method is applicable for both All-Active and Single-Active configurations, whereas the Local Bias method is suitable only for All-Active configurations. Moreover, the ESI Label method is effective across different network domains, while Local Bias is constrained to networks where there is no change in the next hop between the NVEs attached to the same ES. Nonetheless, some operators favor the Local Bias method due to its simplification of the encapsulation process, reduced resource consumption on NVEs, and the fact that the ingress NVE always forwards traffic locally to other interfaces, thereby decreasing the delay in reaching multihomed hosts.

This document extends the EVPN multihoming procedures to allow operators to select the preferred Split Horizon method for a given NVO tunnel according to their specific requirements. The choice between Local Bias and ESI Label Split Horizon is now allowed for tunnel encapsulations that support both methods, and this selection is advertised along with the EVPN A-D per ES route. IP tunnels that do not support both methods, such as VXLAN or NVGRE, will continue to adhere to the procedures specified in [RFC8365].

2. BGP EVPN Extensions

Extensions to EVPN are required to enable NVEs to advertise their preferred Split Horizon method for a given ES. Figure 1 illustrates the ESI Label extended community ([RFC7432] Section 7.5), which is consistently advertised alongside the EVPN A-D per ES route. All NVEs connected to an ES advertise an A-D per ES route for that ES, including the extended community, which communicates information regarding the multihoming mode (either All-Active or Single-Active) and, if necessary, specifies the ESI Label to be utilized.

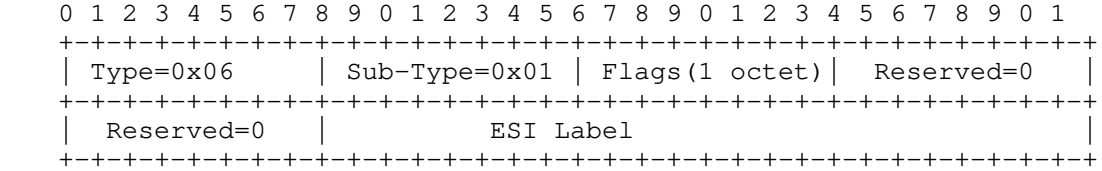


Figure 1: ESI Label extended community

[RFC7432] defines the low-order bit of the Flags octet (bit 0) as the "Single-Active" bit:

* A value of 0 means that the multihomed ES is operating in All-Active multihoming redundancy mode.

* A value of 1 means that the multihomed ES is operating in Single-Active multihoming redundancy mode.

Section 5 establishes a registry for the Flags octet, designating the "Single-Active" bit as the low-order bit of the newly defined multihoming redundancy mode field.

2.1. The Split Horizon Type

[RFC8365] does not include any explicit indication regarding the Split Horizon method in the A-D per Ethernet Segment (ES) route. In this document, the Split Horizon procedure defined in [RFC8365] is considered the default behavior, presuming that Local Bias is employed exclusively for IP tunnels, while ESI Label-based Split Horizon is used for IP-based MPLS tunnels. This document specifies that the two high-order bits in the Flags octet (bits 6 and 7) constitute the "Split Horizon Type" (SHT) field, where:

```
 7 6 5 4 3 2 1 0
+-+-+-+-+-+-+-+-+
|SHT|       |RED|
+-+-+-+-+-+-+-+-+
RED = "Multihoming redundancy mode" field

SHT bit 7 6
-----------
        0 0  --> Default SHT. Backwards compatible with [RFC8365] and [RFC7432]
        0 1  --> Local Bias
        1 0  --> ESI Label based filtering
        1 1  --> reserved for future use
```

* SHT = 00 is backwards compatible with [RFC8365] and [RFC7432], and indicates that the advertising NVE intends to use the default or native SHT. The default SHT is shown in Table 1 for each encapsulation. An egress NVE that follows the [RFC8365] behavior and does not support this specification will ignore the SHT bits (which is equivalent to process them as value of 00).

* SHT = 01 indicates that the advertising NVE intends to use Local Bias procedures in the ES for which the AD per-ES route is advertised.

* SHT = 10 indicates that the advertising NVE intends to use the ESI Label based Split Horizon method procedures in the ES for which the AD per-ES route is advertised.

* SHT = 11 is a reserved value, for future use.

2.2. Use of the Split Horizon Type In A-D Per ES Routes

The following behavior is observed:

* An SHT value of 01 or 10 MUST NOT be used with encapsulations that support only one SHT in Table 1, and MAY be used by encapsulations that support the two SHTs in Table 1.

* An SHT value different from 00 expresses the intent to use a specific Split Horizon method, but does not reflect the actual operational SHT used by the advertising NVE, unless all the NVEs attached to the ES advertise the same SHT.

* In case of inconsistency in the SHT value advertised by the NVEs attached to the same ES for a given EVI, all the NVEs MUST revert to the [RFC8365] behavior, and use the default SHT in Table 1, irrespective of the advertised SHT.

* An SHT different from 00 MUST NOT be set if the Single-Active bit is set. A received A-D per ES route where Single-Active and SHT bits are different from zero MUST follow the treat-as-withdraw behavior [RFC7606].

* The SHT MUST have the same value in each Ethernet A-D per ES route that an NVE advertises for a given ES and a given encapsulation (see Section 3 for NVEs supporting multiple encapsulations).

As an example, egress NVEs that support IP-based MPLS tunnels, such as MPLSoGRE or MPLSoUDP, will advertise A-D per ES routes for the ES along with the BGP Encapsulation extended community, as defined in [RFC9012]. This extended community indicates the encapsulation type (MPLSoGRE or MPLSoUDP) and may use the SHT value of 01 or 10 to signify the intent to use Local Bias or ESI Label, respectively.

An egress NVE MUST NOT use an SHT value other than 00 when advertising an A-D per ES route with encapsulation types such as VXLAN, NVGRE, MPLS, or no BGP tunnel encapsulation extended community, as specified in [RFC9012]. In all these cases, it is presumed that there is no choice for the Split Horizon method; therefore, the SHT value MUST be set to 00. If a route with any of the mentioned encapsulation options is received and has an SHT value different from 00, it SHOULD apply the treat-as-withdraw behavior.

An egress NVE advertising A-D per ES route(s) for an ES with GENEVE encapsulation MAY use an SHT value of 01 or 10. A value of 01 indicates the intent to use Local Bias, regardless of the presence of an Ethernet option TLV with a non-zero Source-ID, as described in [I-D.ietf-bess-evpn-geneve]. A value of 10 indicates the intent to

use ESI Label-based Split Horizon, and it is only valid if an Ethernet option TLV with non-zero Source-ID is present. A value of 00 indicates the default behavior outlined in Table 1, which is to use Local Bias if no ESI-Label is present in the Ethernet option TLV, or if there is no Ethernet option TLV. Otherwise, the ESI Label Split Horizon method is applied.

These procedures assume a single encapsulation supported in the egress NVE. Section 3 describes additional procedures for NVEs supporting multiple encapsulations.

2.3. ESI Label Value In A-D Per ES Routes

This document also updates [RFC8365] regarding the value that is advertised in the ESI Label field of the ESI Label extended community, as follows:

* The A-D per ES route(s) for an ES MAY have an ESI Label value of zero if the SHT value is 01. Section 2.2 specifies the scenarios where the SHT can be 01. An ESI Label value of zero eliminates the need to allocate labels in cases where they are not utilized, such as in the Local Bias method.

* The A-D per ES route(s) for an ES MAY have an ESI Label value of zero for VXLAN or NVGRE encapsulations.

2.4. Backwards Compatibility With RFC8365 NVEs

As discussed in Section 2.2 this specification is backwards compatible with the Split Horizon filtering behavior in [RFC8365] and a non-upgraded NVE can be attached to the same ES as other NVEs supporting this specification.

An NVE maintains an administrative SHT value for an Ethernet Segment (ES), which is advertised alongside the A-D per ES route, and an operational SHT value, which is the one actually used regardless of what the NVE has advertised. The administrative SHT matches the operational SHT if all the NVEs attached to the ES have the same administrative SHT.

This document assumes that an implementation of [RFC7432] or [RFC8365] that does not support the specifications in this document will ignore the values of all the Flags in the ESI Label extended community, except for the Single-Active bit. Based on this assumption, a non-upgraded NVE will disregard any SHT value other than 00. If an upgraded NVE receives at least one A-D per ES route for the ES with an SHT value of 00, it MUST revert its operational SHT to the default Split Horizon method, as described in Table 1, irrespective of its administrative SHT.

For instance, consider an NVE attached to ES N that receives two A-D per ES routes for N from different NVEs, NVE1 and NVE2. If the route from NVE1 has an SHT value of 00 and the one from NVE2 has an SHT value of 01, the NVE MUST use the default Split Horizon method specified in Table 1 as its operational SHT, regardless of its administrative SHT.

All NVEs attached to an ES with an operational SHT value of 10 MUST advertise a valid, non-zero ESI Label. If the operational SHT value is 01, the ESI Label MAY be zero. If the operational SHT value is 00, the ESI Label may be zero only if the default encapsulation supports Local Bias exclusively, and the NVEs do not require the presence of a valid, non-zero ESI Label.

If an NVE changes its operational SHT value from 01 (Local Bias) to 00 (Default SHT) due to the presence of a new non-upgraded NVE in the ES, and it previously advertised a zero ESI Label, it MUST send an update with a valid, non-zero ESI Label, unless all the non-upgraded NVEs in the ES support only Local Bias. For example, consider NVE1 and NVE2 using MPLSoUDP as encapsulation, attached to the same Ethernet Segment ES1, and advertising an SHT value of 01 (Local Bias) with a zero ESI Label value. Suppose NVE3, which does not support this specification, joins ES1 and advertises an SHT value of 00 (default). Upon receiving NVE3's A-D per ES route, NVE1 and NVE2 MUST update their A-D per ES routes for ES1 to include a valid, non-zero ESI Label value. The assumption here is that NVE3 only supports the default ESI Label-based Split Horizon filtering.

3. Procedures for NVEs Supporting Multiple Encapsulations

As specified in [RFC8365], an NVE that supports multiple data plane encapsulations (e.g., VXLAN, NVGRE, MPLS, MPLSoUDP, GENEVE) must indicate all supported encapsulations using BGP Encapsulation extended communities as defined in [RFC9012] for all EVPN routes. This section provides clarification on the multihoming Split Horizon behavior for NVEs that advertise and receive multiple BGP Encapsulation extended communities along with the A-D per ES routes. This section uses the notation {x, y} to denote the encapsulations

advertised in BGP Encapsulation extended communities, where x and y represent different encapsulation values.

It is important to note that an NVE MAY advertise multiple A-D per ES routes for the same ES, rather than a single route, with each route conveying a set of Route Targets (RT). The total set of Route Targets associated with a given ES is referred to as the RT-set for that ES. Each of the EVIs represented in the RT-set will have its RT included in one, and only one, A-D per ES route for the ES. When multiple A-D per ES routes are advertised for the same ES, each route must have a distinct Route Distinguisher.

As per [RFC8365], an NVE that advertises multiple encapsulations in the A-D per ES route(s) for an ES MUST advertise encapsulations that use the same Split Horizon filtering method in the same route. For example:

* An A-D per ES route for ES-x may be advertised with {VXLAN, NVGRE} encapsulations.

* An A-D per ES route for ES-y may be advertised with {MPLS, MPLSoUDP, MPLSoGRE} encapsulations (or a subset).

* But an A-D per ES route for ES-z MUST NOT be advertised with {MPLS, VXLAN} encapsulations.

This document extends the described behavior as follows:

a. An A-D per ES route for ES-x may be advertised with multiple encapsulations, some of which support a single Split Horizon method. In this case, the Split Horizon Type (SHT) value MUST be 00. For instance, encapsulations such as {VXLAN, NVGRE}, {VXLAN, GENEVE}, or {MPLS, MPLSoGRE, MPLSoUDP} can be advertised in an A-D per ES route. In all these cases, the SHT value MUST be 00.

b. An A-D per ES route for ES-y may be advertised with multiple encapsulations that all support both Split Horizon methods. In this case, the SHT value MAY be 01 if the preferred method is Local Bias, or 10 if the ESI Label-based method is desired. For example, encapsulations such as {MPLSoGRE, MPLSoUDP, GENEVE} (or a subset) MAY be advertised in an A-D per ES route with an SHT value of 01. The ESI Label value in this case MAY be zero.

c. If ES-z with RT-set composed of (RT1, RT2, RT3.. RTn) supports multiple encapsulations requiring different Split Horizon methods, a distinct A-D per ES route (or group of routes) per Split Horizon method MUST be advertised. For example, consider an ES-z with n Route Targets (RTs) where:

* the EVIs corresponding to (RT1..RTi) support VXLAN,

* the ones for (RTi+1..RTm) (with i<m) support MPLSoUDP with Local Bias,

* and the ones for (RTm+1..RTn) (with m<n) support GENEVE with ESI Label based Split Horizon.

In this scenario, three groups of A-D per ES routes MUST be advertised for ES-z:

* A-D per ES route group 1, including (RT1..RTi), with encapsulation {VXLAN}, and an SHT value of 00. The ESI Label MAY be zero.

* A-D per ES route group 2, including (RTi+1..RTm), with encapsulation {MPLSoUDP}, and an SHT value of 01. The ESI Label MAY be zero.

* A-D per ES route group 3, including (RTm+1..RTn), with encapsulation {GENEVE}, and an SHT value of 10. The ESI Label MUST have a valid, non-zero value, and the Ethernet option as defined in [RFC8926] MUST be advertised.

As per [RFC8365], it is the responsibility of the operator of a given EVI to ensure that all of the NVEs within that EVI support a common encapsulation. Failure to meet this condition may result in service disruption or failure.

4. Security Considerations

All the security considerations described in [RFC7432] are applicable to this document.

Additionally, this document modifies the procedures for Split Horizon filtering as outlined in [RFC8365], offering operators a choice between Local Bias and ESI Label-based filtering for tunnels that support both methods. Misconfiguration of the desired Split Horizon Type (SHT) could lead to forwarding behaviors that differ from the intended configuration. Apart from this risk, this document describes procedures to ensure that all Provider Edge (PE) devices or Network Virtualization Edges (NVEs) connected to the same Ethernet Segment (ES) agree on a common SHT method, with a fallback to a default behavior in case of a mismatch in the SHT bits being advertised by any two PEs or NVEs in the Ethernet Segment. Consequently, unauthorized changes to the SHT configuration by an attacker on a single PE or NVE of the Ethernet Segment should not cause traffic disruption (as long as the SHT value is valid as per this document) but may result in alterations to forwarding behavior.

5. IANA Considerations

This document creates a registry called "EVPN ESI Label Extended Community Flags" for the 1-octet Flags field in the ESI Label Extended Community [RFC7432]. Initial registrations are made for the "Multihoming redundancy mode" field in bits 0 and 1, and the "Split Horizon Type" field in bits 6 and 7, as follows:

Bit Position	Name	Reference
0-1	Multihoming redundancy mode	[RFC7432]
2-5	Unused	
6-7	Split Horizon Type	This Document

Table 2

In addition, the "Multihoming redundancy mode" field is initialized as follows:

Value	Multihoming redundancy mode
00	All-Active mode
01	Single-Active mode
10	Unused
11	Unused

Table 3

And the field "Split Horizon Type" is initialized as follows:

Value	Split Horizon Type value
00	Default SHT
01	Local Bias
10	ESI Label based filtering
11	Unused

Table 4

New registrations in the "EVPN ESI Label Extended Community Flags" registry will be made through the "IETF Review" procedure defined in [RFC8126]. This registry is located in the "Border Gateway Protocol (BGP) Extended Communities" registry.

6. Acknowledgments

The authors would like to thank Anoop Ghanwani, Gyan Mishra and Jeffrey Zhang for their review and useful comments. Thanks to Gunter van de Velde as well, for his thorough review.

7. References

7.1. Normative References

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

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

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <https://www.rfc-editor.org/info/rfc7432>.

[RFC8365] Sajassi, A., Ed., Drake, J., Ed., Bitar, N., Shekhar, R., Uttaro, J., and W. Henderickx, "A Network Virtualization Overlay Solution Using Ethernet VPN (EVPN)", RFC 8365, DOI 10.17487/RFC8365, March 2018, <https://www.rfc-editor.org/info/rfc8365>.

[RFC9252] Dawra, G., Ed., Talaulikar, K., Ed., Raszuk, R., Decraene, B., Zhuang, S., and J. Rabadan, "BGP Overlay Services Based on Segment Routing over IPv6 (SRv6)", RFC 9252, DOI 10.17487/RFC9252, July 2022, <https://www.rfc-editor.org/info/rfc9252>.

7.2. Informative References

[I-D.ietf-bess-evpn-geneve] Boutros, S., Sajassi, A., Drake, J., Rabadan, J., and S. Aldrin, "EVPN control plane for Geneve", Work in Progress, Internet-Draft, draft-ietf-bess-evpn-geneve-08, 5 July 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-bess-evpn-geneve-08>.

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <https://www.rfc-editor.org/info/rfc7348>.

[RFC4023] Worster, T., Rekhter, Y., and E. Rosen, Ed., "Encapsulating MPLS in IP or Generic Routing Encapsulation (GRE)", RFC 4023, DOI 10.17487/RFC4023, March 2005, <https://www.rfc-editor.org/info/rfc4023>.

[RFC7637] Garg, P., Ed. and Y. Wang, Ed., "NVGRE: Network Virtualization Using Generic Routing Encapsulation", RFC 7637, DOI 10.17487/RFC7637, September 2015, <https://www.rfc-editor.org/info/rfc7637>.

[RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", RFC 7510, DOI 10.17487/RFC7510, April 2015, <https://www.rfc-editor.org/info/rfc7510>.

[RFC8926] Gross, J., Ed., Ganga, I., Ed., and T. Sridhar, Ed., "Geneve: Generic Network Virtualization Encapsulation", RFC 8926, DOI 10.17487/RFC8926, November 2020, <https://www.rfc-editor.org/info/rfc8926>.

[RFC9012] Patel, K., Van de Velde, G., Sangli, S., and J. Scudder, "The BGP Tunnel Encapsulation Attribute", RFC 9012, DOI 10.17487/RFC9012, April 2021, <https://www.rfc-editor.org/info/rfc9012>.

[RFC7606] Chen, E., Ed., Scudder, J., Ed., Mohapatra, P., and K. Patel, "Revised Error Handling for BGP UPDATE Messages", RFC 7606, DOI 10.17487/RFC7606, August 2015, <https://www.rfc-editor.org/info/rfc7606>.

[RFC8660] Bashandy, A., Ed., Filsfils, C., Ed., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with the MPLS Data Plane", RFC 8660, DOI 10.17487/RFC8660, December 2019, <https://www.rfc-editor.org/info/rfc8660>.

[RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <https://www.rfc-editor.org/info/rfc8986>.

[I-D.ietf-nvo3-vxlan-gpe] Maino, F., Kreeger, L., and U. Elzur, "Generic Protocol Extension for VXLAN (VXLAN-GPE)", Work in Progress, Internet-Draft, draft-ietf-nvo3-vxlan-gpe-13, 4 November 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-nvo3-vxlan-gpe-13>.

Authors' Addresses

Jorge Rabadan (editor)
Nokia
520 Almanor Avenue
Sunnyvale, CA 94085
United States of America
Email: jorge.rabadan@nokia.com

Kiran Nagaraj
Nokia
520 Almanor Avenue
Sunnyvale, CA 94085
United States of America
Email: kiran.nagaraj@nokia.com

Wen Lin
Juniper Networks
Email: wlin@juniper.net

Ali Sajassi
Cisco Systems, Inc.
Email: sajassi@cisco.com

CCWG
Internet-Draft
Obsoletes: 5033 (if approved)
Intended status: Best Current Practice
Expires: 24 January 2025

M. Duke, Ed.
Google LLC
G. Fairhurst, Ed.
University of Aberdeen

23 July 2024

Specifying New Congestion Control Algorithms
draft-ietf-ccwg-rfc5033bis-07

Abstract

Introducing new or modified congestion control algorithms in the global Internet has possible ramifications to both the flows using the proposed congestion control algorithms and to flows using a standardized congestion control algorithm. Therefore, the IETF must proceed with caution when evaluating proposals for alternate congestion control. The goal of this document is to provide guidance for considering standardization of a proposed congestion control algorithm at the IETF. It obsoletes RFC5033 to reflect changes in the congestion control landscape.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-ccwg-rfc5033bis/.

Discussion of this document takes place on the Congestion Control Working Group (ccwg) Working Group mailing list (mailto:ccwg@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/ccwg/. Subscribe at https://www.ietf.org/mailman/listinfo/ccwg/.

Source for this draft and an issue tracker can be found at https://github.com/ietf-wg-ccwg/rfc5033bis.

Status of This Memo

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

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

Copyright Notice

Table of Contents

1. Introduction

This document provides guidelines for the IETF to use when evaluating a proposed congestion control algorithm that differs from the general congestion control principles outlined in [RFC2914]. The guidance is intended to be useful to authors proposing congestion control algorithms and for the IETF community when evaluating whether a proposal is appropriate for publication in the RFC series and for deployment in the Internet.

This document obsoletes [RFC5033], which was published in 2007 as a Best Current Practice to evaluate proposed congestion control algorithms as Experimental or Proposed Standard RFCs.

The IETF specifies standard Internet congestion control algorithms in the RFC-series. These congestion control algorithms can suffer performance challenges when used in various environments (e.g., high-speed networks, cellular and WiFi wireless technologies, and long distance satellite links), and also when flows carry specific workloads (Voice over IP (VoIP), gaming, and videoconferencing).

When [RFC5033] was published in 2007, TCP [RFC9293] was the dominant consumer of IETF congestion control work, and proposals were typically discussed in the Internet Congestion Control Research Group (ICCRG). Around the same time, the Datagram Congestion Control Protocol (DCCP) [RFC4340] was developed as a method for defining new congestion control algorithms for datagram traffic. The Stream Control Transmission Protocol (SCTP) [RFC9260] re-used TCP congestion control algorithms.

Since then, many conditions have changed. The set of protocols using congestion control algorithms has grown to include QUIC [RFC9000], RTP Media Congestion Avoidance Techniques (RMCAT) (e.g., [RFC8836]) and beyond. Some proponents of alternative congestion control algorithms now have the opportunity to test and deploy at scale without IETF review. There is more interest in specialized use cases, such as data centers (e.g., [RFC8257]), and in support for a variety of upper layer protocols/applications, e.g., real-time protocols. Finally, the community has gained much more experience with indications of congestion beyond packet loss.

Multicast congestion control is a considerably less mature field of study and is not in the scope of this document. However, Section 4 of [RFC8085] provides additional guidelines for multicast and broadcast usage of UDP.

Congestion control algorithms have been developed outside of the IETF, including at least two that saw large scale deployment: CUBIC [HRX08] and Bottleneck Bandwidth and Round-trip propagation time (BBR) [BBR-draft].

CUBIC was documented in a research publication in 2007 [HRX08], and was then adopted as the default congestion control algorithm for the TCP implementation in Linux. It was already used in a significant fraction of TCP connections over the Internet before being documented in an Informational Internet-Draft in 2015, published as an Informational RFC in 2017 as [RFC8312] and then as a Proposed Standard in 2023 [RFC9438].

At the time of writing, BBR is being developed as an internal research project by Google, with the first implementation contributed to Linux kernel 4.19 in 2016. It was described in an IRTF Internet-

Draft in 2018, and that Internet- Draft is regularly updated to document the evolving versions of the algorithm [BBR-draft]. BBR is currently widely used for Google services using either TCP or QUIC, and is also widely deployed outside of Google.

We cannot say now whether the original authors of [RFC5033] expected that developers would be waiting for IETF review before widely deploying a new congestion control algorithm over the Internet, but the examples of CUBIC and BBR teach us that deployment of new algorithms is not, in fact, gated by the publication of the algorithm as an RFC.

Nevertheless, a specification for a congestion control algorithm provides a number of advantages:

* It can help implementers, operators, and other interested parties develop a shared understanding of how the algorithm works and how it is expected to behave in various scenarios and configurations.

* It can help potential contributors understand the algorithm, which can make it easier for them to suggest improvements and/or identify limitations. Furthermore, the specification can help multiple contributors align on a consensus change to the algorithm.

* A specification that is accessible to anyone can circumvent the issue that some implementers may be unable to read open source reference implementations due to the constraints of some open source licenses.

Beyond helping develop specific algorithm proposals, guidelines can also serve as a reminder to potential inventors and developers of the multiple facets of the congestion control problem.

The evaluation guidelines in this document are intended to be consistent with the congestion control principles from [RFC2914] of preventing congestion collapse, considering fairness, and optimizing a flow's own performance in terms of throughput, delay, and loss. [RFC2914] also discusses the goal of avoiding a congestion control "arms race" among competing transport protocols.

This document does not give hard-and-fast requirements for an appropriate congestion control algorithm. Rather, the document provides a set of criteria that should be considered and weighed by the developers of alternative algorithms and by the IETF in the context of each proposal.

The high-order criterion for advancing any proposal within the IETF is a serious scientific study of the pros and cons that occurs when the proposal is considered for publication by the IETF, or before it is deployed at large scale.

After initial studies, authors are encouraged to write a specification of their proposal for publication in the RFC series. This allows others to understand and investigate the wealth of proposals in this space.

This document is intended to reduce the barriers to entry for new congestion control work to the IETF. As such, proponents of new congestion control algorithms ought not to interpret these criteria as a checklist of requirements before approaching the IETF. Instead, proponents are encouraged to think about these issues beforehand, and have the willingness to do the work implied by the remainder of this document.

2. Specification of Requirements

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

3. Guidelines for Authors

3.1. Guidelines for Authors about Evaluation

This document does not specify specific evaluation methods, short of internet-scale deployment and measurement, to test the criteria described below. There are multiple possible approaches to evaluation. Each has a role, and the most appropriate approach depends on the criteria being evaluated and the maturity of the specification.

For many algorithms, an initial evaluation will consider individual protocol mechanisms in a simulator to analyse their stability and safety across a wide range of conditions, including overload. For example, [RFC8869] describes evaluation test cases for interactive real-time media over wireless networks. Such results could also be published or discussed in IRTF research groups, such as ICCRG and MAPRG.

Before a proposed congestion control algorithm is published as an Experimental or Standards Track RFC, the community SHOULD gain practical experience with implementation and experience using the

algorithm. Where there is implementation by independent teams, this can help provide assurance that a specification has avoided assumptions or ambiguity. An independent evaluation by multiple teams helps provide assurance that the design meets the evaluation criteria, and can assess typical interactions with other traffic. This evaluation could use an emulated laboratory environment or a controlled experiment (within a limited domain or at Internet-scale). Evidence of results is normally considered by the working group in deciding if a specification is ready for publication and ought to be documented in any request for the working group to publish the specification.

3.2. Guidelines for Authors about Document Status

This document applies to proposals for congestion control algorithms that seek Experimental or Standards Track status. Evaluation of both cases involves the same questions, but with different expectations for both the answers and the degree of certainty of those answers.

Congestion control algorithms without empirical evidence of Internet-scale deployment SHOULD seek Experimental status.

Congestion control algorithms with a record of measured Internet-scale deployment MAY directly seek the Standards Track if there is solid data that reflects that it is safe, and the design is stable, guided by the considerations in Section 6. However, the existence of this data does not waive the other considerations in this document.

Experimental specifications SHOULD NOT be deployed as a default. They SHOULD only be deployed in situations where they are being actively measured, and where it is possible to deactivate them if there are signs of pathological behavior.

Each published congestion control algorithm is REQUIRED to include a statement in the abstract indicating whether or not there is IETF consensus that the proposed congestion control algorithm is considered safe for use on the Internet. Each published algorithm is also required to include a statement in the abstract describing environments where the protocol is not recommended for deployment. There can be environments where the congestion control algorithm is deemed safe for use, but it is still is not recommended for use because it does not perform well for the user.

As examples of such statements, [RFC3649] specifies HighSpeed TCP and includes a statement in the abstract stating that the proposed congestion control algorithm is Experimental, but may be deployed in the Internet. In contrast, the Quick-Start document [RFC4782] includes a paragraph in the abstract stating that the mechanism is

only being proposed for use in controlled environments. The abstract specifies environments where the Quick- Start request could give false positives (and therefore would be unsafe for incremental deployment where some routers forward, but do not process the option). The abstract also specifies environments where packets containing the Quick-Start request could be dropped in the network; in such an environment, Quick-Start would not be unsafe to deploy, but deployment is not recommended because it could lead to unnecessary delays for the connections attempting to use Quick-Start. The Quick-Start method is discussed as an example in [RFC9049].

Strictly speaking, Informational RFCs in the IETF stream need not meet all of the criteria in this document, as they do not carry a formal recommendation from the community. Instead, the community judges the publication of Informational RFCs based on the value of their addition to the RFC series.

Although out of the scope of this document, proponents of a new algorithm could alternatively seek publication as an Informational or Experimental RFC via the Internet Research Task Force (IRTF). In general, these algorithms are expected to be less mature than ones that follow the procedures in this document. Authors documenting deployed congestion control algorithms that cannot be changed by IETF or IRTF review are invited to publish as an Informational RFC via the Independent Stream Editor (ISE).

4. Specifying Algorithms for Use in Controlled Environments

Algorithms can be designed for general Internet deployment or for use in controlled environments [RFC8799]. Within a controlled environment, an operator can ensure that flows are isolated from other Internet flows, or they might allow these flows to share resources with other Internet flows. A data center is an example of a controlled environment, which often deploys fabrics with rich signalling from switches to endpoints.

Algorithms that rely on specific functions or configurations in the network need to provide a reference or specification for these functions (an RFC or another stable specification). The IETF will need to assess whether the relevant working group is able to review the proposed new algorithm and whether there is sufficient experience to understand any dependent functions.

In evaluating a new proposal for use in a controlled environment, the IETF needs to understand the usage, e.g., how the usage is scoped to the controlled environment, whether the algorithm will share resources with Internet traffic, and consider what could happen if used in a protocol that is bridged across an Internet path.

Algorithms that are designed to be confined to a controlled environment and are not intended for use in the general Internet, might instead seek real-world data for those environments. In such cases, the evaluation criteria in the remainder of this document might not apply.

5. Evaluation Criteria

As noted above, authors are expected to do a well-rounded evaluation of the pros and cons of congestion control algorithms that are brought to the IETF. The following guidelines are designed to help authors and the IETF community. Concerns that fall outside the scope of these guidelines are certainly possible; these guidelines should not be considered an all-encompassing check-list.

When considering a proposed congestion control algorithm, the community MUST consider the following criteria. These criteria will be evaluated in various domains (see Section 6 and Section 7).

5.1. Single Algorithm Behavior

The criteria in this section evaluate the congestion control algorithm when one or more flows using that algorithm share a bottleneck link (i.e., with no flows using a differing congestion control algorithm).

5.1.1. Protection Against Congestion Collapse

A congestion control algorithm should either stop sending when the packet drop rate exceeds some threshold [RFC3714], or should include some notion of "full backoff". For "full backoff", at some point the algorithm would reduce the sending rate to one packet per round-trip time and then exponentially backoff the time between single packet transmissions if the congestion persists. Exactly when either "full backoff" or a pause in sending comes into play will be algorithm-specific. However, as discussed in [RFC2914] and [RFC8961], this requirement is crucial to protect the network in times of extreme (persistent) congestion.

If full backoff is used, this test does not require that the mechanism must be identical to that of TCP ([RFC6298], [RFC8961]). For example, this does not preclude full backoff mechanisms that would give flows with different round- trip times comparable capacity during backoff.

5.1.2. Protection Against Bufferbloat

A congestion control algorithm should try to avoid maintaining excessive queues in the network. Exactly how the algorithm achieves this is algorithm-specific, but see [RFC8961] and [RFC8085] for requirements.

Bufferbloat [Bufferbloat] refers to the building of excessive queues in the network. Many network routers are configured with very large buffers. The standards-track Reno [RFC5681] and CUBIC [RFC9438] congestion control algorithms send at progressively higher rates until a First-In First-Out (FIFO) buffer completely fills, and packet losses then occur. Every connection passing through that bottleneck experiences increased latency due to the high buffer occupancy. This adds unwanted latency that negatively impacts highly interactive applications such as videoconferencing or games, but it also affects routine web browsing and video playing.

This problem has been widely discussed since 2011 [Bufferbloat], but was not discussed in the Congestion Control Principles published in September 2002 [RFC2914]. The Reno and CUBIC congestion control algorithms do not address this problem, but a new congestion control algorithm has the opportunity to improve the state of the art.

5.1.3. Protection Against High Packet Loss

A congestion control algorithm should try to avoid causing excessively high rates of packet loss. To accomplish this, it should avoid excessive increases in sending rate, and reduce its sending rate if experiencing high packet loss.

The first version of the BBR algorithm [BBRv1-draft] failed this requirement. Experimental evaluation [BBRv1-Evaluation] showed that it caused a sustained rate of packet loss when multiple BBRv1 flows shared a bottleneck and the buffer size was less than roughly one and a half times the Bandwidth Delay Product (BDP). This was unsatisfactory, and indeed further versions provided a fix for this aspect of BBR [BBR-draft].

This requirement does not imply that the algorithm should react to packet losses in exactly the same way as current standards-track congestion control algorithms (e.g., [RFC5681]).

5.1.4. Fairness within the Proposed Congestion Control Algorithm

When multiple competing flows all use the same proposed congestion control algorithm, the proposal should explore how the capacity is shared among the competing flows. Capacity fairness can be important when a small number of similar flows compete to fill a bottleneck. However, it can also not be useful, for example, when comparing flows that seek to send at different rates, or if some of the flows do not last sufficiently long to approach asymptotic behavior.

5.1.5. Short Flows

A great deal of congestion control analysis concerns the steady-state behavior of long flows. However, many Internet flows are relatively short-lived. Many short-lived flows today remain in the "slow start" mode of operation [RFC5681] that commonly features exponential congestion window growth because the flow never experiences congestion (e.g., packet loss).

A proposed congestion control algorithm MUST consider how new and short-lived flows affect long-lived flows, and vice versa.

5.2. Mixed Algorithm Behavior

Mixed algorithm behavior criteria evaluate the interaction of the proposed congestion control algorithm with commonly deployed congestion control algorithms.

In contexts where differing congestion control algorithms are used, it is important to understand whether the proposed congestion control algorithm could result in more harm than previous standards-track algorithms (e.g., [RFC5681], [RFC9002], [RFC9438]) to flows sharing a common bottleneck. The measure of harm is not restricted to unequal capacity, but ought also to consider metrics such as the introduced latency, or an increase in packet loss. An evaluation MUST assess the potential to cause starvation, including assurance that a loss of all feedback (e.g., detected by expiry of a retransmission time out) results in backoff.

5.2.1. Existing General-Purpose Congestion Control

A proposed congestion control algorithm SHOULD be evaluated when competing using standard IETF congestion controls, e.g. [RFC5681], [RFC9002], [RFC9438]. A proposed congestion control algorithm that has a significantly negative impact on flows using standard congestion control might be suspect, and this aspect should be part of the community’s decision making with regards to the suitability of the proposed congestion control algorithm. The community should also

consider other non-standard congestion control algorithms that are known to be widely deployed.

Note that this guideline is not a requirement for strict Reno- or CUBIC- friendliness as a prerequisite for a proposed congestion control mechanism to advance to Experimental or Standards Track status. As an example, HighSpeed TCP is a congestion control mechanism specified as Experimental, that is not TCP- friendly in all environments. When a new congestion control algorithm is deployed, the existing major algorithm deployments need to be considered to avoid severe performance degradation. Note that this guideline does not constrain the interaction with non-best-effort flows.

As an example from an Experimental RFC, fairness with standard TCP is discussed in Sections 4 and 6 of [RFC3649] (HighSpeed TCP) and using spare capacity is discussed in Sections 6, 11.1, and 12 of [RFC3649].

5.2.2. Real-Time Congestion Control

General-purpose algorithms need to coexist in the Internet with real-time congestion control algorithms, which, in general, have finite throughput requirements (i.e., do not seek to utilize all available capacity) and more strict latency bounds. See [RFC8836] for a description of the characteristics of this use case and the resulting requirements.

[RFC8868] provides suggestions for real-time congestion control design and [RFC8867] suggests test cases. [RFC9392] describes some considerations for the RTP Control Protocol (RTCP). In particular, real-time flows can use less frequent feedback (acknowledgement) than that provided by reliable transports. This document does not change the informational status of those RFCs.

A proposed congestion control algorithm SHOULD consider coexistence with widely deployed real-time congestion control algorithms. Regrettably, at the time of writing (2024), many algorithms with detailed public specifications are not widely deployed, while many widely deployed real-time congestion control algorithms have incomplete public specifications. It is hoped that this situation will change.

To the extent that behavior of widely deployed algorithms is understood, proponents of a proposed congestion control algorithm can analyze and simulate a proposal’s interaction with those algorithms. To the extent they are not, experiments can be conducted where possible.

Real-time flows can be directed into distinct queues via Differentiated Services Code Points (DSCP) or other mechanisms, which can substantially reduce the interplay with other traffic. However, a proposal targeting general Internet use can not assume this is always the case.

Section 7.2 describes the impact of network transport circuit breaker algorithms. [RFC8083] also defines a minimal set of RTP circuit breakers that operate end-to-end across a path. This identifies conditions under which a sender needs to stop transmitting media data to protect the network from excessive congestion. It is expected that, in the absence of long-lived excessive congestion, RTP applications running on best-effort IP networks will be able to operate without triggering these circuit breakers.

5.2.3. Short and Long Flows

The effect on short-lived and long-lived flows using other common congestion control algorithms MUST be evaluated, as in Section 5.1.5.

5.3. Other Criteria

5.3.1. Differences with Congestion Control Principles

A proposed congestion control algorithm SHOULD include a clear explanation of any deviations from [RFC2914] and [RFC7141].

5.3.2. Incremental Deployment

A congestion control algorithm proposal MUST discuss whether it allows for incremental deployment in the targeted environment. For a mechanism targeted for deployment in the current Internet, the proposal SHOULD discuss what is known (if anything) about the correct operation of the mechanisms with some of the equipment in the current Internet, e.g., routers, transparent proxies, WAN optimizers, intrusion detection systems, home routers, and the like.

Similarly, if the proposed congestion control algorithm is intended only for specific environments (and not the global Internet), the proposal SHOULD consider how this intention is to be realised. The community will have to address the question of whether the scope can be enforced by stating the restrictions, or whether additional protocol mechanisms are required to enforce this scoping. The answer will necessarily depend on the proposed change.

As an example from an Experimental RFC, deployment issues are discussed in Sections 10.3 and 10.4 of [RFC4782] (Quick-Start).

6. General Use

The criteria in Section 5 will be evaluated in the following scenarios. Unless a proposed congestion control specification explicitly forbids use on the public Internet, the community MUST reach consensus that it meets the criteria in these scenarios for the proposed congestion control algorithm to progress.

The evaluation in each scenario SHOULD occur over a representative range of bandwidths, delays, and queue depths. Of course, the set of parameters representative of the public Internet will change over time.

These criteria are intended to capture a statistically dominant set of Internet conditions. In the case that a proposed congestion control algorithm has been tested at Internet scale, the results from that deployment are often useful for answering these questions.

6.1. Paths with Tail-drop Queues

The performance of a congestion control algorithm is affected by the queue discipline applied at the bottleneck link. The drop-tail queue discipline (using a FIFO buffer) MUST be evaluated. See Section 7.1 for evaluation of other queue disciplines.

6.2. Tunnel Behavior

When a proposed congestion control algorithm relies on explicit signals from the path, the proposal MUST consider the effect of traffic passing through a tunnel, where routers may not be aware of the flow.

The design of tunnels and similar encapsulations might need to consider nested congestion control interactions. For example, when ECN is used by both an IP and lower layer technology [ECN-Encaps].

6.3. Wired Paths

Wired networks are usually characterized by extremely low rates of packet loss except for those due to queue drops. They tend to have stable aggregate capacity, usually higher than other types of links, and low non-queueing delay. Because the properties are relatively simple, wired links are typically used as a "baseline" case even if they are not always the bottleneck link in the modern Internet.

6.4. Wireless Paths

While the early Internet was dominated by wired links, the properties of wireless links have become important to Internet performance. In particular, a proposed congestion control algorithm should be evaluated in situations where some packet losses are due to radio effects, rather than router queue drops; the link capacity varies over time due to changing link conditions; and media access delays and link-layer retransmission lead to increased jitter in round-trip times. See [RFC3819] and Section 16 of [Tools] for further discussion of wireless properties.

7. Special Cases

The criteria in Section 5 will be evaluated in the following scenarios, unless the proposed congestion control algorithm specifically excludes its use in a scenario. For these specific use-cases, the community MAY allow a proposal to progress even if the criteria indicate an unsatisfactory result for these scenarios.

In general, measurements from Internet-scale deployments might not expose the properties of operation in each of these scenarios, because they are not as ubiquitous as the General Use scenarios.

7.1. Active Queue Management (AQM)

The proposed congestion control algorithm SHOULD be evaluated under a variety of bottleneck queue disciplines. The effect of an AQM discipline can be hard to detect by Internet evaluation. At a minimum, a proposal should reason about an algorithm's response to various AQM disciplines. Simulation or empirical results are, of course, valuable.

Among the AQM techniques that might have an impact on a proposed congestion control algorithm are Flow Queue CoDel (FQ-CoDel) [RFC8290]; Proportional Integral Controller Enhanced (PIE) [RFC8033]; and Low Latency, Low Loss, and Scalable Throughput (L4S) [RFC9332].

A proposed congestion control algorithm that sets one of the two Explicit Congestion Transport (ECT) codepoints in the IP header can gain the benefits of receiving Explicit Congestion Notification (ECN) Congestion Experienced (CE) signals from an on-path AQM [RFC8087]. Use of ECN [RFC3168], [RFC9332] requires the congestion control algorithm to react when it receives a packet with an ECN-CE marking. This reaction needs to be evaluated to confirm that the algorithm conforms with the requirements of the ECT codepoint that was used.

Note that evaluation of AQM techniques -- as opposed to their impact on a specific proposed congestion control algorithm -- is out of scope of this document. [RFC7567] describes design considerations for AQMs.

7.2. Operation with the Envelope set by Network Circuit Breakers

Some equipment in the network uses an automatic mechanism to continuously monitor the use of resources by a flow or aggregate set of flows [RFC8084]. Such a network transport circuit breaker can automatically detect excessive congestion, and when detected, it can terminate (or significantly reduce the rate of) the flow(s). A well-designed congestion control algorithm ought to react before the flow uses excessive resources, and therefore will operate within the envelope set by network transport circuit breaker algorithms.

7.3. Paths with Varying Delay

An Internet Path can include simple links, where the minimum delay is the propagation delay, and any additional delay can be attributed to link buffering. This cannot be assumed. An Internet Path can also include complex subnetworks where the minimum delay changes over various time scales, resulting in a non- stationary minimum delay.

Varying delay occurs when a subnet changes the forwarding path to optimise capacity, resilience, etc. It could also arise when a subnet uses a capacity management method where the available resource is periodically distributed among the active nodes. A node might then have to buffer data until an assigned transmission opportunity or until the physical path changes (e.g., when the length of a wireless path changes, or the physical layer changes its mode of operation). Variation also arises when traffic with a higher priority DSCP pre-empts transmission of traffic with a lower class. In these cases, the delay varies as a function of external factors, and attempting to infer congestion from an increase in the delay results in reduced throughput. This variation in the delay over short timescales (jitter) might not be distinguishable from jitter that results from other effects.

A proposed congestion control algorithm SHOULD be evaluated to ensure its operation is robust when there is a significant change in the minimum delay.

7.4. Internet of Things

The "Internet of Things" (IoT) is a broad concept, but when evaluating a proposed congestion control algorithm, it is often associated with unique characteristics: IoT nodes might be more constrained in power, CPU, or other parameters than conventional Internet hosts. This might place limits on the complexity of any given algorithm. These power and radio constraints might make the volume of control packets in a given algorithm a key evaluation metric.

Extremely low-power links can lead to very low throughput and a low bandwidth- delay product, well below the standard operating range of most Internet flows.

Furthermore, many IoT applications do not a have a human in the loop, and therefore can have weaker latency constraints because they do not relate to a user experience. Congestion control algorithm can still need to share the path with other flows with different constraints.

7.5. Paths with High Delay

A proposed congestion control algorithm ought not to presume that all general Internet paths have a low delay. Some paths include links that contribute much more delay than for a typical Internet path. Satellite links often have delays longer than typical for wired paths [RFC2488] and high delay bandwidth products [RFC3649].

Paths can also present a variable delay as described in Section 7.3.

7.6. Misbehaving Nodes

A proposed congestion control algorithm SHOULD explore how the algorithm performs with non-compliant senders, receivers, or routers. In addition, the proposal should explore how a proposed congestion control algorithm performs with outside attackers. This can be particularly important for proposed congestion control algorithms that involve explicit feedback from routers along the path.

As an example from an Experimental RFC, performance with misbehaving nodes and outside attackers is discussed in Sections 9.4, 9.5, and 9.6 of [RFC4782]. This includes discussion of misbehaving senders and receivers; collusion between misbehaving routers; misbehaving middleboxes; and the potential use of Quick- Start to attack routers or to tie up available Quick-Start bandwidth.

7.7. Extreme Packet Reordering

A proposed congestion control algorithm ought not to presume that all general Internet paths reliably deliver packets in order. [RFC4653] discusses the effect of extreme packet reordering.

7.8. Transient Events

A proposed congestion control algorithm SHOULD consider how the proposed congestion control algorithm would perform in the presence of transient events such as sudden onset of congestion, a routing change, or a mobility event. Routing changes, link disconnections, intermittent link connectivity, and mobility are discussed in more detail in Section 16 of [Tools].

As an example from an Experimental RFC, response to transient events is discussed in Section 9.2 of [RFC4782].

7.9. Sudden changes in the Path

An IETF transport is not tied to a specific Internet path or type of path. The set of routers that form a path can and do change with time. This will cause the properties of the path to change with respect to time. A proposed congestion control algorithm MUST evaluate the impact of changes in the path, and be robust to changes in path characteristics on the interval of common Internet re-routing intervals.

7.10. Multipath Transport

Multipath transport protocols permit more than one path to be differentiated and used by a single connection at the sender. A multipath sender can schedule which packets travel on which of its active paths. This enables a tradeoff in timeliness and reliability. There are various ways that multipath techniques can be used.

One example use is to provide fail-over from one path to another when the original path is no longer viable, or provides inferior performance. Designs need to independently track the congestion state of each path, and demonstrate independent congestion control for each path being used. Authors of a proposed multipath congestion control algorithm that implements path fail-over MUST evaluate the harm resulting from a change in the path, and show that this does not result in flow starvation. Synchronisation of failover (e.g., where multiple flows change their path on similar timeframes) can also contribute to harm and/or reduce fairness. These effects also ought to be evaluated.

Another example use is concurrent multipath, where the transport protocol simultaneously schedules a flow to aggregate the capacity across multiple paths. The Internet provides no guarantee that different paths (e.g., using different endpoint addresses) are disjoint. This introduces additional implications: A congestion control algorithm proposal MUST evaluate the potential harm to other flows when the multiple paths share a common congested bottleneck or share resources that are coupled between different paths, such as an overall capacity limit). A proposal SHOULD consider the potential for harm to other flows. Synchronisation of congestion control mechanisms (e.g., where multiple flows change their behaviour on similar timeframes) can also contribute to harm and/or reduce fairness. These effects also ought to be evaluated.

At the time of writing (2024), there are currently no Standards Track RFCs for concurrent multipath, but there is an Experimental RFC [RFC6356] that specifies a concurrent multipath congestion control algorithm for MPTCP [RFC8684].

7.11. Data Centers

Data centers are characterized by very low latencies (< 2 ms). Many workloads involve bursty traffic where many nodes complete a task at the same time. As a controlled environment, data centers often deploy fabrics that employ rich signalling from switches to endpoints. Furthermore, the operator can often limit the number of operating congestion control algorithms.

For these reasons, data center congestion controls are often distinct from those running elsewhere on the Internet (see Section 4). A proposed congestion control need not coexist well with all other algorithms if it is intended for data centers, but the proposal SHOULD indicate which are expected to safely coexist with it.

8. Security Considerations

This document does not represent a change to any aspect of the TCP/IP protocol suite and therefore does not directly impact Internet security. The implementation of various facets of the Internet's current congestion control algorithms do have security implications (e.g., as outlined in [RFC5681]).

The IETF process that results in publication needs to ensure that these security implications are considered. Proposed congestion control algorithms therefore ought to be mindful of pitfalls, and SHOULD examine any potential security issues that may arise.

9. IANA Considerations

This document has no IANA actions.

10. References

10.1. Normative References

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

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <https://www.rfc-editor.org/rfc/rfc2914>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <https://www.rfc-editor.org/rfc/rfc5681>.

[RFC7141] Briscoe, B. and J. Manner, "Byte and Packet Congestion Notification", BCP 41, RFC 7141, DOI 10.17487/RFC7141, February 2014, <https://www.rfc-editor.org/rfc/rfc7141>.

[RFC8083] Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", RFC 8083, DOI 10.17487/RFC8083, March 2017, <https://www.rfc-editor.org/rfc/rfc8083>.

[RFC8084] Fairhurst, G., "Network Transport Circuit Breakers", BCP 208, RFC 8084, DOI 10.17487/RFC8084, March 2017, <https://www.rfc-editor.org/rfc/rfc8084>.

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <https://www.rfc-editor.org/rfc/rfc8085>.

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

[RFC8961] Allman, M., "Requirements for Time-Based Loss Detection", BCP 233, RFC 8961, DOI 10.17487/RFC8961, November 2020, <https://www.rfc-editor.org/rfc/rfc8961>.

[RFC9002] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", RFC 9002, DOI 10.17487/RFC9002, May 2021, <https://www.rfc-editor.org/rfc/rfc9002>.

[RFC9438] Xu, L., Ha, S., Rhee, I., Goel, V., and L. Eggert, Ed., "CUBIC for Fast and Long-Distance Networks", RFC 9438, DOI 10.17487/RFC9438, August 2023, <https://www.rfc-editor.org/rfc/rfc9438>.

10.2. Informative References

[BBR-draft]
Cardwell, N., Cheng, Y., Yeganeh, S. H., Swett, I., and V. Jacobson, "BBR Congestion Control", Work in Progress, Internet-Draft, draft-cardwell-iccrg-bbr-congestion-control-02, 7 March 2022, <https://datatracker.ietf.org/doc/html/draft-cardwell-iccrg-bbr-congestion-control-02>.

[BBRv1-draft]
Cardwell, N., Cheng, Y., Yeganeh, S. H., and V. Jacobson, "BBR Congestion Control", Work in Progress, Internet-Draft, draft-cardwell-iccrg-bbr-congestion-control-00, 3 July 2017, <https://datatracker.ietf.org/doc/html/draft-cardwell-iccrg-bbr-congestion-control-00>.

[BBRv1-Evaluation]
Zitterbart, M., "Experimental evaluation of BBR congestion control", 2017 IEEE 25th International Conference on Network Protocols (ICNP) , 2017, <https://ieeexplore.ieee.org/document/8117540>.

[Bufferbloat]
Kathleen Nichols, "Bufferbloat: Dark Buffers in the Internet", ACM Queue Volume 9, Issue 11 , 2011, <https://queue.acm.org/detail.cfm?id=2071893>.

[ECN-Encaps]
Briscoe, B. and J. Kaippallimalil, "Guidelines for Adding Congestion Notification to Protocols that Encapsulate IP", Work in Progress, Internet-Draft, draft-ietf-tsvwg-ecn-encap-guidelines-22, 5 December 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-tsvwg-ecn-encap-guidelines-22>.

[HRX08] Ha, S., Rhee, I., and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant", ACM SIGOPS Operating Systems Review, vol. 42, no. 5, pp. 64-74 , July 2008, <https://doi.org/10.1145/1400097.1400105>.

[RFC2488] Allman, M., Glover, D., and L. Sanchez, "Enhancing TCP Over Satellite Channels using Standard Mechanisms", BCP 28, RFC 2488, DOI 10.17487/RFC2488, January 1999, <https://www.rfc-editor.org/rfc/rfc2488>.

[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, <https://www.rfc-editor.org/rfc/rfc3168>.

[RFC3649] Floyd, S., "HighSpeed TCP for Large Congestion Windows", RFC 3649, DOI 10.17487/RFC3649, December 2003, <https://www.rfc-editor.org/rfc/rfc3649>.

[RFC3714] Floyd, S., Ed. and J. Kempf, Ed., "IAB Concerns Regarding Congestion Control for Voice Traffic in the Internet", RFC 3714, DOI 10.17487/RFC3714, March 2004, <https://www.rfc-editor.org/rfc/rfc3714>.

[RFC3819] Karn, P., Ed., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, DOI 10.17487/RFC3819, July 2004, <https://www.rfc-editor.org/rfc/rfc3819>.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <https://www.rfc-editor.org/rfc/rfc4340>.

[RFC4653] Bhandarkar, S., Reddy, A. L. N., Allman, M., and E. Blanton, "Improving the Robustness of TCP to Non-Congestion Events", RFC 4653, DOI 10.17487/RFC4653, August 2006, <https://www.rfc-editor.org/rfc/rfc4653>.

[RFC4782] Floyd, S., Allman, M., Jain, A., and P. Sarolahti, "Quick-Start for TCP and IP", RFC 4782, DOI 10.17487/RFC4782, January 2007, <https://www.rfc-editor.org/rfc/rfc4782>.

[RFC5033] Floyd, S. and M. Allman, "Specifying New Congestion Control Algorithms", BCP 133, RFC 5033, DOI 10.17487/RFC5033, August 2007, <https://www.rfc-editor.org/rfc/rfc5033>.

[RFC5166] Floyd, S., Ed., "Metrics for the Evaluation of Congestion Control Mechanisms", RFC 5166, DOI 10.17487/RFC5166, March 2008, <https://www.rfc-editor.org/rfc/rfc5166>.

[RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <https://www.rfc-editor.org/rfc/rfc6298>.

[RFC6356] Raiciu, C., Handley, M., and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols", RFC 6356, DOI 10.17487/RFC6356, October 2011, <https://www.rfc-editor.org/rfc/rfc6356>.

[RFC7567] Baker, F., Ed. and G. Fairhurst, Ed., "IETF Recommendations Regarding Active Queue Management", BCP 197, RFC 7567, DOI 10.17487/RFC7567, July 2015, <https://www.rfc-editor.org/rfc/rfc7567>.

[RFC8033] Pan, R., Natarajan, P., Baker, F., and G. White, "Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem", RFC 8033, DOI 10.17487/RFC8033, February 2017, <https://www.rfc-editor.org/rfc/rfc8033>.

[RFC8087] Fairhurst, G. and M. Welzl, "The Benefits of Using Explicit Congestion Notification (ECN)", RFC 8087, DOI 10.17487/RFC8087, March 2017, <https://www.rfc-editor.org/rfc/rfc8087>.

[RFC8257] Bensley, S., Thaler, D., Balasubramanian, P., Eggert, L., and G. Judd, "Data Center TCP (DCTCP): TCP Congestion Control for Data Centers", RFC 8257, DOI 10.17487/RFC8257, October 2017, <https://www.rfc-editor.org/rfc/rfc8257>.

[RFC8290] Hoeiland-Joergensen, T., McKenney, P., Taht, D., Gettys, J., and E. Dumazet, "The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm", RFC 8290, DOI 10.17487/RFC8290, January 2018, <https://www.rfc-editor.org/rfc/rfc8290>.

[RFC8312] Rhee, I., Xu, L., Ha, S., Zimmermann, A., Eggert, L., and R. Scheffenegger, "CUBIC for Fast Long-Distance Networks", RFC 8312, DOI 10.17487/RFC8312, February 2018, <https://www.rfc-editor.org/rfc/rfc8312>.

[RFC8684] Ford, A., Raiciu, C., Handley, M., Bonaventure, O., and C. Paasch, "TCP Extensions for Multipath Operation with Multiple Addresses", RFC 8684, DOI 10.17487/RFC8684, March 2020, <https://www.rfc-editor.org/rfc/rfc8684>.

[RFC8799] Carpenter, B. and B. Liu, "Limited Domains and Internet Protocols", RFC 8799, DOI 10.17487/RFC8799, July 2020, <https://www.rfc-editor.org/rfc/rfc8799>.

[RFC8836] Jesup, R. and Z. Sarker, Ed., "Congestion Control Requirements for Interactive Real-Time Media", RFC 8836, DOI 10.17487/RFC8836, January 2021, <https://www.rfc-editor.org/rfc/rfc8836>.

[RFC8867] Sarker, Z., Singh, V., Zhu, X., and M. Ramalho, "Test Cases for Evaluating Congestion Control for Interactive Real-Time Media", RFC 8867, DOI 10.17487/RFC8867, January 2021, <https://www.rfc-editor.org/rfc/rfc8867>.

[RFC8868] Singh, V., Ott, J., and S. Holmer, "Evaluating Congestion Control for Interactive Real-Time Media", RFC 8868, DOI 10.17487/RFC8868, January 2021, <https://www.rfc-editor.org/rfc/rfc8868>.

[RFC8869] Sarker, Z., Zhu, X., and J. Fu, "Evaluation Test Cases for Interactive Real-Time Media over Wireless Networks", RFC 8869, DOI 10.17487/RFC8869, January 2021, <https://www.rfc-editor.org/rfc/rfc8869>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[RFC9049] Dawkins, S., Ed., "Path Aware Networking: Obstacles to Deployment (A Bestiary of Roads Not Taken)", RFC 9049, DOI 10.17487/RFC9049, June 2021, <https://www.rfc-editor.org/rfc/rfc9049>.

[RFC9260] Stewart, R., TÃ¼xen, M., and K. Nielsen, "Stream Control Transmission Protocol", RFC 9260, DOI 10.17487/RFC9260, June 2022, <https://www.rfc-editor.org/rfc/rfc9260>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/rfc/rfc9293>.

[RFC9332] De Schepper, K., Briscoe, B., Ed., and G. White, "Dual-Queue Coupled Active Queue Management (AQM) for Low Latency, Low Loss, and Scalable Throughput (L4S)", RFC 9332, DOI 10.17487/RFC9332, January 2023, <https://www.rfc-editor.org/rfc/rfc9332>.

[RFC9392] Perkins, C., "Sending RTP Control Protocol (RTCP) Feedback for Congestion Control in Interactive Multimedia Conferences", RFC 9392, DOI 10.17487/RFC9392, April 2023, <https://www.rfc-editor.org/rfc/rfc9392>.

[Tools] Floyd, S. and E. Kohler, "Tools for the Evaluation of Simulation and Testbed Scenarios", Work in Progress , July 2007, <https://datatracker.ietf.org/doc/draft-irtf-tmrg-tools>.

Acknowledgments

Sally Floyd and Mark Allman were the authors of this document's predecessor, [RFC5033], which served the community well for over a decade.

Thanks to Richard Scheffenegger for helping to get this revision process started.

The editors would like to thank Mohamed Boucadair, Neal Cardwell, Reese Enghardt, Jonathan Lennox, Matt Mathis, Zahed Sarker, Juergen Schoenwaelder, Dave Taht, Sean Turner, Michael Welzl, Magnus Westerlund, and Greg White for suggesting improvements to this document.

Discussions with Lars Eggert and Aaron Falk seeded the original RFC5033. Bob Briscoe, Gorry Fairhurst, Doug Leith, Jitendra Padhye, Colin Perkins, Pekka Savola, members of TSVWG, and participants at the TCP Workshop at Microsoft Research all provided feedback and contributions to that document. It also drew from [RFC5166].

Evolution of RFC5033bis

Since draft-ietf-ccwg-rfc5033bis-06

* OPSDIR review

* ARTART review

Since draft-ietf-ccwg-rfc5033bis-05

* AD evaluation comments

Since draft-ietf-ccwg-rfc5033bis-04

* Editorial pass after shepherd review.

Since draft-ietf-ccwg-rfc5033bis-03

* Harmonised the "proposed congestion control algorithm"
* Addressed issues.
* Examined RFC-2119 keywords and consistency with other RFCs.
* Added text on constrained environments/limited domains
* Added text on circuit breakers and aligned with other RFCs.
* Several editorial passes

Since draft-ietf-ccwg-rfc5033bis-02

* Added discussion of real-time protocols
* Added discussion of short flows
* Listed properties of wired networks
* Added IoT section
* Added discussion of AQM response
* Rewrote the "Document Status" section
* Adding improved first sentence of abstract and intro.
* Added section on Multicast, noting this is out of scope
* Editorial changes

Since draft-ietf-ccwg-rfc5033bis-01

* Added discussion of multipath transports
* Totally reorganized central sections of the draft

Since draft-ietf-ccwg-rfc5033bis-00

* Added QUIC, other congestion control standards
* Added wireless environments
* Aligned motivation for this work with the CCWG charter
* Refined discussion of QuickStart

Since draft-scheffenegger-congress-rfc5033bis-00

* Renamed file to reflect WG adpotion
* Updated authorship and acknowledgements.
* Include updated text suggested by Dave Taht
* Added criterion for bufferbloat
* Mentioned CUBIC and BBR as motivation
* Include section to track updates between revisions
* Update references

Since RFC5033

* converted to Markdown and xml2rfc v3
* various formatting changes.

Contributors

Christian Huitema
Private Octopus, Inc.
Email: huitema@huitema.net

Authors' Addresses

Martin Duke (editor)
Google LLC
Email: martin.h.duke@gmail.com

Godred Fairhurst (editor)
University of Aberdeen

Email: gorry@erg.abdn.ac.uk

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 2 January 2025

F. Fieau
E. Stephan
Orange
G. Guillaume
C. Christoph
Broadpeak

1 July 2024

CDNI Metadata for Delegated Credentials
draft-ietf-cdni-https-delegation-subcerts-09

Abstract

The delivery of content over HTTPS involving multiple CDNs raises credential management issues. This document defines metadata in the CDNI Control and Metadata interface to setup HTTPS delegation using delegated credentials from an Upstream CDN (uCDN) to a Downstream CDN (dCDN).

Status of This Memo

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

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

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

This Internet-Draft will expire on 2 January 2025.

Table of Contents

1. Introduction

Content delivery over HTTPS using one or more CDNs along the path requires credential management. This specifically applies when an entity delegates to another trusted entity delivery of content via HTTPS.

This document defines the CDNI Metadata interface to setup HTTPS delegation using delegated credentials (as defined by [RFC9345]) between an upstream CDN (uCDN) and downstream CDN (dCDN).

2. Terminology

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

This document uses terminology from CDNI framework documents: CDNI framework document [RFC7336], CDNI requirements [RFC7337] and CDNI interface specifications documents: CDNI Metadata interface [RFC8006].

3. CDNI Footprint and Capabilities Advertisement interface (FCI) capabilities object for delegated credentials

A dCDN should advertise its supported delegation methods using the Footprint and Capabilities Advertisement interface (FCI) as defined in [RFC8008]. The FCI.Metadata object allows a dCDN to advertise its capabilities and the Metadata interface (MI) objects supported by the dCDN. Accordingly, to announce the support for delegated credentials, the dCDN should announce the support of MI.DelegatedCredentials as shown in the example below.

```
{
  "capabilities": [
    {
      "capability-type": "FCI.Metadata",
      "capability-value": {
        "metadata": [
          "MI.DelegatedCredentials",
          "... other supported MI objects ..."
        ]
      },
      "footprints": [
        "Footprint objects"
      ]
    }
  ]
}
```

This document also defines an object that announces to the delegating entity how many delegated credentials the downstream supports such that the delegating entity can provide the corresponding number of delegated credentials. For that purpose we introduce the FCI object FCI.DelegationCredentials.

3.1. FCI.DelegatedCredentials

The FCI.DelegationCredentials object enables advertising the maximum number of delegated credentials supported by the dCDN. This number typically (but not necessarily) corresponds to the number of servers designated by the dCDN to support delegated credentials.

The property PrivateKeyEncryptionKey contains a public key provided by the dCDN that MUST be used by the uCDN to encrypt private keys whenever such private keys are transmitted to the dCDN using MI.DelegatedCredentials (see Section 4).

Property: number-delegated-certs-supported

Description: Number of delegated credentials supported by the dCDN.

Type: integer

Mandatory-to-Specify: Yes

Property: PrivateKeyEncryptionKey

Description: Base64-encoded (as defined in Section 4 of [RFC4648]) public key of the dCDN to be used by the uCDN to encrypt private keys.

Type: string

Mandatory-to-Specify: No

The following is an example of the FCI.DelegatedCredentials.

```
{
  "capabilities": [
    {
     "capability-type": "FCI.DelegatedCredentials",
     "capability-value": {
        "number-delegated-certs-supported": 10
      }
     "footprints": [
        <Footprint objects>
      ]
    }
  ]
}
```

3.2. Expected usage of the property number of supported delegated credentials

The dCDN uses the FCI.DelegatedCredentials object to announce the number of servers that support delegated credentials

When the uCDN receives the FCI.DelegatedCredentials object it can issue the supported number of delegated credentials to the dCDN. When configuring the dCDN, the uCDN MAY decide to provide less than the maximum supported delegated credentials to the dCDN. Note that, within a dCDN, different deployment possibilities of the delegated credentials on the endpoints exist. The dCDN may use one single delegated credential and deploy it on multiple endpoints. Alternatively, the dCDN may deploy a different delegated credential for each endpoint (provided that the uCDN delivers enough different delegated credentials). This choice is at the discretion of the dCDN and depends on the number of delegated credentials provided by the uCDN.

The FCI.DelegationCredentials object does not address expiry and renewal of delegated credentials. Once the uCDN has provided delegated credentials via the MI, uCDN SHOULD monitor the provided credentials and their expiry times. The uCDN SHOULD timely refresh dCDN credentials via the MI. The uCDN may decide not to monitor the validity period of delegated credentials and not to refresh the credentials, for example in cases of short-term one shot deployments or once it decided to deprovision a dCDN. If the delegated credential is not renewed on time by the uCDN, the servers of the dCDN that only have expired delegated credentials MUST refuse any new TLS connection that requires an up-to-date delegated credential.

4. CDNI Metadata interface (MI) metadata object for delegated credentials

As expressed in [RFC9345], when an uCDN has delegated to a dCDN, the dCDN presents the "delegated_credential" during the TLS handshake [RFC8446] to the User Agent, instead of its own certificate. This implies that the dCDN is also in the possession of the private key corresponding to the public key in DelegatedCredential.cred [RFC9345]. This allows the User Agent to verify the signature in CertificateVerify message ([RFC8446] Section 4.4.3.) sent and signed by the dCDN.

This section defines the MI.DelegatedCredentials object containing an array of delegated credentials and optionally the corresponding private keys. The CDNI MI [RFC8006] describes the CDNI metadata distribution mechanisms according to which a dCDN can retrieve the MI.DelegatedCredentials object from the uCDN.

The properties of the MI.DelegatedCredentials object are as follows:

Property: delegated-credentials

Description: Array of delegated credentials

Type: Array of DelegatedCredentialObject objects

Mandatory-to-Specify: Yes

The DelegatedCredentialObject object is composed of the following properties:

Property: delegated-credential

Description: Base64-encoded (as defined in Section 4 of [RFC4648]) version of a CertificateEntry as defined in [RFC8446] Section 4.4.2. The CertificateEntry MUST contain a DelegatedCredential structure (as defined in [RFC9345]) using the extension in the CertificateEntry of its end-entity certificate (see [RFC9345] section 4.1.1)

Type: string

Mandatory-to-Specify: Yes

Property: private-key

Description: Encrypted and base64-encoded (as defined in Section 4 of [RFC4648]) private key corresponding to the public key contained in the DelegatedCredential

Type: string

Mandatory-to-Specify: No

The private-key property is not mandatory. If not specified, it is assumed that the dCDN generated the public-private key pair for the delegated credential itself and provided the public key information with an out-of-band mechanism to the uCDN. As discussed in Section 7, it is NOT RECOMMENDED to communicate private keys to the dCDN using MI.

If the private-key property is used, the transported private key MUST be encrypted using the PrivateKeyEncryptionKey specified in FCI.DelegatedCredentials. The base64 envelope format for this property MUST use JWE [RFC7516], whereas the private key is included as JWE Ciphertext in the JWE.

Below, please see an example MI.DelegatedCredential Object.

```
{
"generic-metadata-type": "MI.DelegatedCredentials",
"generic-metadata-value": {
    "delegated-credentials": [
            {"delegated-credential":
                "cBBfm8KK6pPz/tdgKyedwA...
                iXCCIAmzMM0R8FLI3Ba0UQ=="},
            {"delegated-credential":
                "4pyIGtjFdys1+9y/4sS/Fg...
                J+h9lnRY/xgmi65RLGKoRw=="},
            {"delegated-credential":
                "6PWFO0g2AXvUaULXLObcVA...
                HXoldT/qaYCCNEyCc8JM2A=="}
        ]
    }
}
```

5. Delegated credentials call flow

An example call-flow using delegated credentials is depicted in Figure 1.

1. It is assumed that the uCDN has been provisioned and configured with a certificate. Note that it is out of scope of CDNI and the present document how and from where (e.g., CSP) the uCDN acquired its certificate.

2. The uCDN generates a set of delegated credentials (here it is assumed that public keys of the dCDN are known). Note that the uCDN may generate this material at different points in time, e.g., in advance to have a pool of delegated credentials or on-demand when the dCDN announces its maximum number of supported delegated credentials.

3. Using the CDNI FCI [RFC8008], the dCDN advertises MI.DelegatedCredentials capabilities to the uCDN. The dCDN further uses FCI.DelegatedCredentials to advertise the maximum number of supported delegated credentials.

4. Using the CDNI MI [RFC8006], the dCDN acquires the MI.DelegatedCredentials, retrieving an array of delegated credentials.

5. The client establishes a TLS connection with an endpoint of the dCDN according to [RFC9345] using the delegated credentials retrieved in step 4.

6. When some delegated credentials are about to expire, the uCDN uses the CDNI MI [RFC8006] to provide new, valid delegated credentials.

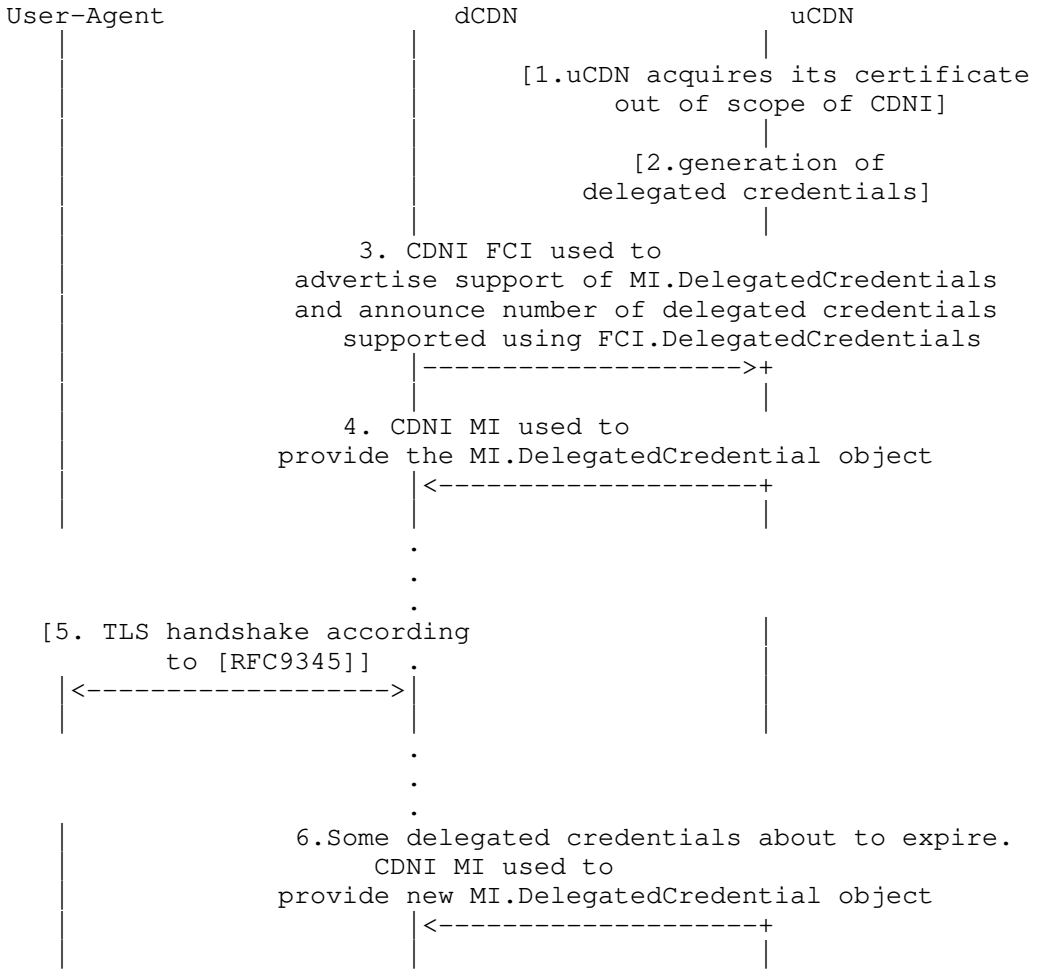


Figure 1: Example call-flow of Delegated credentials in CDNI

6. IANA Considerations

This document requests IANA registration of the following entries under the "CDNI Payload Types" registry hosted by IANA regarding "CDNI delegation":

Payload Type	Specification
MI.DelegatedCredentials	RFCthis
FCI.DelegatedCredentials	RFCthis

Table 1

[RFC Editor: Please replace RFCthis with the published RFC number for this document.]

6.1. CDNI MI DelegatedCredentials Payload Type

Purpose: The purpose of this Payload Type is to distinguish delegated credentials MI Objects

Interface: MI/FCI

Encoding: see Section 4

6.2. CDNI FCI DelegatedCredentials Payload Type

Purpose: The purpose of this Payload Type is to advertise the number of delegated credentials needed (and any associated capability advertisement)

Interface: FCI

Encoding: see Section 3.1

7. Security Considerations

The extensions defined enable providing delegated credentials to dCDNs. A delegated credential can only be used by a dCDN if it is in possession of the associated private key. Similarly, an attacker requires access to the private key in order to exploit delegated credential and impersonate dCDN nodes. Thus, leakage of only the delegated credential without the private key represents a limited security risk.

Delegated credentials and associated private keys are short-lived (per default the maximum validity period set to 7 days in [RFC9345]) and as such a single leaked delegated credential with its private key represents a limited security risk. Still, it is NOT RECOMMENDED to send private keys through the MI. Omitting the private key further limits the possibility exploits by an attacker to exploit the delegated credential.

If despite this recommendation, the private key is communicated via the MI, the transported private key MUST be encrypted within a JWE envelope using the encryption key (PrivateKeyEncryptionKey) provided within the FCI.DelegatedCredentials by the dCDN. Note that the specified encryption method does not offer forward secrecy. If the dCDN's encryption key becomes compromised in the future, then all encrypted JWEs will become compromised. Due to the short-lived nature of delegated credentials, the impact is limited.

It is also important to ensure that an attacker is not able to systematically retrieve a consecutive or consistent set of delegated credentials and associated private keys. Such an attack would allow the attacker to systematically impersonate dCDN nodes. The MI objects defined in the present document are transferred via the interfaces defined in CDNI [RFC8006]. [RFC8006] describes how to secure these interfaces, protecting the integrity, confidentiality and ensuring the authenticity of the dCDN and uCDN, which should prevent an attacker to systematically retrieve delegated credential and associated private keys.

8. Privacy Considerations

The information, FCI, and MI objects defined in the present document do not contain any personally identifiable information (PII). As such this document does not change or alter the Confidentiality and Privacy Consideration outlined in the CDNI Metadata and Footprint and Capabilities RFCs [RFC8006].

A single or systematic retrieval of delegated credentials and associated private keys would allow the attacker to decrypt any data sent by the end user intended for the end service, which may include PII.

9. Normative References

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

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC7336] Peterson, L., Davie, B., and R. van Brandenburg, Ed., "Framework for Content Distribution Network Interconnection (CDNI)", RFC 7336, DOI 10.17487/RFC7336, August 2014, <https://www.rfc-editor.org/info/rfc7336>.

[RFC7337] Leung, K., Ed. and Y. Lee, Ed., "Content Distribution Network Interconnection (CDNI) Requirements", RFC 7337, DOI 10.17487/RFC7337, August 2014, <https://www.rfc-editor.org/info/rfc7337>.

[RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <https://www.rfc-editor.org/info/rfc7516>.

[RFC8006] Niven-Jenkins, B., Murray, R., Caulfield, M., and K. Ma, "Content Delivery Network Interconnection (CDNI) Metadata", RFC 8006, DOI 10.17487/RFC8006, December 2016, <https://www.rfc-editor.org/info/rfc8006>.

[RFC8008] Seedorf, J., Peterson, J., Previdi, S., van Brandenburg, R., and K. Ma, "Content Delivery Network Interconnection (CDNI) Request Routing: Footprint and Capabilities Semantics", RFC 8008, DOI 10.17487/RFC8008, December 2016, <https://www.rfc-editor.org/info/rfc8008>.

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

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC9345] Barnes, R., Iyengar, S., Sullivan, N., and E. Rescorla, "Delegated Credentials for TLS and DTLS", RFC 9345, DOI 10.17487/RFC9345, July 2023, <https://www.rfc-editor.org/info/rfc9345>.

Authors' Addresses

Frederic Fieau
Orange
40-48, avenue de la Republique
92320 Chatillon
France
Email: frederic.fieau@orange.com

Emile Stephan
Orange
2, avenue Pierre Marzin
22300 Lannion
France
Email: emile.stephan@orange.com

Guillaume Bichot
Broadpeak
15, rue Claude Chappe
35510 Cesson-Sevigne
France
Email: guillaume.bichot@broadpeak.tv

Christoph Neumann
Broadpeak
15, rue Claude Chappe
35510 Cesson-Sevigne
France
Email: christoph.neumann@broadpeak.tv

COSE
Internet-Draft
Intended status: Standards Track
Expires: 9 January 2025

K. Isobe
SECOM CO., LTD.
H. Tschofenig

O. Steele
Transmute

8 July 2024

CBOR Object Signing and Encryption (COSE) Key Thumbprint
draft-ietf-cose-key-thumbprint-05

Abstract

This specification defines a method for computing a hash value over a CBOR Object Signing and Encryption (COSE) Key. It defines which fields in a COSE Key structure are used in the hash computation, the method of creating a canonical form of the fields, and how to hash the byte sequence. The resulting hash value can be used for identifying or selecting a key that is the subject of the thumbprint.

Status of This Memo

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

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

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

This Internet-Draft will expire on 9 January 2025.

Copyright Notice

Table of Contents

1. Introduction

This specification defines a method for applying a cryptographic hash function (a.k.a. thumbprint) to a CBOR Object Signing and Encryption (COSE) Key structure [RFC9052]. It defines which fields in a COSE Key structure are used in the hash computation, the method of creating a canonical form for those fields, and how to hash the byte sequence. The resulting hash value can be used for identifying or selecting the key that is the subject of the thumbprint, for instance, by using the COSE Key Thumbprint value as a "kid" (key ID) value. The use of the thumbprint of the key as a naming scheme is one of the main use cases of this document. Another use case are key derivation functions that utilize the thumbprints of the public keys of the endpoints, as well as other context, to the derived symmetric key.

This specification defines how thumbprints of COSE keys are created, see Section 3 and Section 4. Additionally, a new CBOR Web Token (CWT) confirmation method is added to the IANA "CWT Confirmation Methods" registry created by [RFC8747]. See Section 3.1 of [RFC8747] for details about the use of a confirmation claim in a CWT with a proof-of-possession key.

2. Terminology

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

3. COSE Key Thumbprint

The thumbprint of a COSE Key MUST be computed as follows:

1. Construct a COSE_Key structure (see Section 7 of [RFC9052]) containing only the required parameters representing the key as described in Section 4 of this document.

2. Apply the deterministic encoding described in Section 4.2.1 of [RFC8949] to the representation constructed in step (1).

3. Hash the bytes produced in step (2) with a cryptographic hash function H. For example, SHA-256 [RFC6234] may be used as a hash function.

The resulting value is the COSE Key Thumbprint with H of the COSE Key. The details of this computation are further described in subsequent sections.

The SHA-256 hash algorithm MUST be supported, other algorithms MAY be supported.

4. Required COSE Key Parameters

Only the required parameters of a key's representation are used when computing its COSE Key Thumbprint value. This section summarizes the required parameters.

The "kty" (label: 1) element MUST be present for all key types and the integer value found in the IANA COSE Key Types registry MUST be used. The tstr data type is not used with the kty element.

Many COSE Key parameters depend on the chosen key type. The subsection below list the required parameters for commonly used key types.

4.1. Octet Key Pair (OKP)

The required parameters for elliptic curve public keys that use the OKP key type, such as X25519, are:

* "kty" (label: 1, data type: int, value: 1)

* "crv" (label: -1, value: int)

* "x" (label: -2, value: bstr)

Details can be found in Section 7.1 of [RFC9053].

4.2. Elliptic Curve Keys with X- and Y-Coordinate Pair

The required parameters for elliptic curve public keys that use the EC2 key type, such as NIST P-256, are:

* "kty" (label: 1, data type: int, value: 2)

* "crv" (label: -1, data type: int)

* "x" (label: -2, data type: bstr)

* "y" (label: -3, data type: bstr)

Details can be found in Section 7.1 of [RFC9053].

Note: [RFC9052] offers both compressed as well as uncompressed point representations. For interoperability, implementations following this specification MUST use the uncompressed point representation. Hence, the y-coordinate is expressed as a bstr. An implementation that uses the compressed point representation MUST compute the uncompressed representation for the purpose of the thumbprint calculation.

4.3. RSA Public Keys

The required parameters for an RSA public key are:

* "kty" (label: 1, data type: int, value: 3)

* "n" (label: -1, data type: bstr)

* "e" (label: -2, data type: bstr)

4.4. Symmetric Keys

The required parameters for a symmetric key are:

* "kty" (label: 1, data type: int, value: 4)

* "k" (label: -1, data type: bstr)

4.5. HSS-LMS

The required parameters for HSS-LMS keys are:

* "kty" (label: 1, data type: int, value: 5)

* "pub" (label: -1, data type: bstr)

4.6. Others

As other key type values are defined, the specifications defining them should be similarly consulted to determine which parameters, in addition to the "kty" element, are required.

5. Miscellaneous Considerations

5.1. Why Not Include Optional COSE Key Parameters?

Optional parameters of COSE Keys are intentionally not included in the COSE Key Thumbprint computation so that their absence or presence in the COSE Key does not alter the resulting value. The COSE Key Thumbprint value is a digest of the parameters required to represent the key as a COSE Key -- not of additional data that may also accompany the key.

Optional parameters are not included so that the COSE Key Thumbprint refers to a key -- not a key with an associated set of key attributes. Different application contexts might or might not include different subsets of optional attributes about the key in the COSE Key structure. If these were included in the calculation of the COSE Key Thumbprint, the values would be different for those COSE Keys, even though the keys are the same. The benefit of including only the required parameters is that the COSE Key Thumbprint of any COSE Key representing the key remains the same, regardless of any other attributes that are present.

Different kinds of thumbprints could be defined by other specifications that might include some or all additional COSE Key parameters, if use cases arise where such different kinds of thumbprints would be useful.

5.2. Selection of Hash Function

A specific hash function must be chosen by an application to compute the hash value of the hash input. For example, SHA-256 [RFC6234] might be used as the hash function by the application. While SHA-256 is a good default choice at the time of writing, the hash function of choice can be expected to change over time as the cryptographic landscape evolves.

Note that in many cases, only the party that creates a key will need to know the hash function used. A typical usage is for the producer of the key to use the thumbprint value as a "kid" (key ID) value. In this case, the consumer of the "kid" treats it as an opaque value that it uses to select the key.

However, in some cases, multiple parties will be reproducing the COSE Key Thumbprint calculation and comparing the results. In these cases, the parties will need to know which hash function was used and use the same one.

5.3. Thumbprints of Keys Not in COSE Key Format

Keys that are in other formats can be represented as COSE Keys. Any party in possession of COSE Keys can use the COSE Key Thumbprint.

5.4. Relationship to Digests of X.509 Values

COSE Key Thumbprint values are computed on the COSE Key object required to represent a key, rather than all parameters of a COSE Key that the key is represented in. Thus, they are more analogous to applications that use digests of X.509 Subject Public Key Info (SPKI) values, which are defined in Section 4.1.2.7 of [RFC5280], than to applications that use digests of complete certificate values, as the "x5t" (X.509 certificate SHA-1 thumbprint) [RFC9360] value defined for X.509 certificate objects does. While logically equivalent to a digest of the SPKI representation of the key, a COSE Key Thumbprint is computed over the CBOR representation of that key, rather than over an ASN.1 representation of it.

5.5. Confirmation Method

[RFC8747] introduced confirmation methods for use with CBOR Web Tokens (CWTs) with the addition of the "cnf" claim. CWTs have been defined in [RFC8392]. This specification adds a new confirmation method based on COSE Key Thumbprints.

The proof-of-possession key is identified using the "ckt" claim, the COSE Key Thumbprint claim. This claim contains the value of the COSE Key Thumbprint encoded as a binary string. Instead of communicating the actual COSE Key only the thumbprint is conveyed. This approach assumes that the recipient is able to obtain the identified COSE Key using the thumbprint contained in the "ckt" claim. In this approach, the issuer of a CWT declares that the presenter possesses a particular key and that the recipient can cryptographically confirm the presenter's proof of possession of the key by including a "ckt" claim in the CWT.

The following example demonstrates the use of the "ckt" claim in a CWT as part of the confirmation method (with line-breaks inserted for editorial reasons):

```
{
 /iss/ 1 : "coaps://as.example.com",
 /aud/ 3 : "coaps://resource.example.org",
 /exp/ 4 : 1361398824,
 /cnf/ 8 : {
   /ckt/ [[TBD1]] : h'496bd8afadf307e5b08c64b0421bf9dc
                      01528a344a43bda88fadd1669da253ec'
  }
}
```

Section 8 registers the "ckt" claim and the confirmation method. The "ckt" claim is expected to be used in the "cnf" claim.

5.6. COSE Key Thumbprint URIs

This specification defines Uniform Resource Identifiers (URIs) to represent a COSE Key Thumbprint value. The design follows the work of the JSON Web Key (JWK) Thumbprint URIs, specified in [RFC9278]. This enables COSE Key Thumbprints to be used, for example, as key identifiers in contexts requiring URIs. This specification defines a URI prefix indicating that the portion of the URI following the prefix is a COSE Key Thumbprint.

The following URI prefix is defined to indicate that the portion of the URI following the prefix is a COSE Key Thumbprint:

urn:ietf:params:oauth:ckt

To make the hash algorithm being used explicit in a URI, the prefix is followed by a hash algorithm identifier and a COSE Key Thumbprint value, each separated by a colon character to form a URI representing a COSE Key Thumbprint.

Hash algorithm identifiers used in COSE Key Thumbprint URIs MUST be values from the "Hash Name String" column in the IANA "Named Information Hash Algorithm Registry" [IANA.Hash.Algorithms]. COSE Key Thumbprint URIs with hash algorithm identifiers not found in this registry are not considered valid and applications will need to detect and handle this error, should it occur.

Since the URN is encoded as a string, the output of the COSE Key Thumbprint computation described in Section 3 MUST be base64url encoded without padding.

[RFC7515] specifies Base64url encoding as follows:

"Base64 encoding using the URL- and filename-safe character set defined in Section 5 of RFC 4648 [RFC4648], with all trailing '=' characters omitted and without the inclusion of any line breaks, whitespace, or other additional characters. Note that the base64url encoding of the empty octet sequence is the empty string. (See Appendix C of [RFC7515] for notes on implementing base64url encoding without padding.)"

The base64url encoding of the thumbprint shown in Section 6 is shown below (with a line-break added for readability purposes).

SWvYr63zB-WwjGSwQhv53AFSijRKQ72oj63RZp2iU-w

The full example of a COSE Key Thumbprint URI is shown below, again with a line-break added.

urn:ietf:params:oauth:ckt:sha-256:
SWvYr63zB-WwjGSwQhv53AFSijRKQ72oj63RZp2iU-w

6. Example

This section demonstrates the COSE Key Thumbprint computation for the following example COSE Key containing an ECC public key.

For better readability, the example is first presented in CBOR diagnostic format (with the long line broken for display purposes only).

```
  {
    / kty set to EC2 = Elliptic Curve Keys /
    1:2,
    / crv set to P-256 /
    -1:1,
    / public key: x-coordinate /
    -2:h'65eda5a12577c2bae829437fe338701a10aaa375e1bb5b5de108de439c0
8551d',
    / public key: y-coordinate /
    -3:h'1e52ed75701163f7f9e40ddf9f341b3dc9ba860af7e0ca7ca7e9eecd008
4d19c',
    / kid is bstr, not used in COSE Key Thumbprint /
    2:h'1decade2facade3'
  }
```

The example above corresponds to the following CBOR encoding (with link breaks added for display purposes only):

```
A5010220012158 2065EDA5A12577C2BAE829437FE338701A10AAA375E1BB5B5DE108D
E439C08551D2258201E52ED75701163F7F9E40DDF9F341B3DC9BA860AF7E0CA7CA7E9
EECD0084D19C0258246D65726961646F632E6272616E64796275636B406275636B6C6
16E642E6578616D706C65
```

Not all of the parameters from the example above are used in the COSE Key Thumbprint computation since the required parameters of an elliptic curve public key are (as listed in Section 4.2) "kty", "crv", "x", and "y".

The resulting COSE Key structure, in CBOR diagnostic format with line-breaks added for better readability, with the minimum parameters in the correct order are.

```
{
   1:2,
  -1:1,
  -2:h'65eda5a12577c2bae829437fe338701a
       10aaa375e1bb5b5de108de439c08551d',
  -3:h'1e52ed75701163f7f9e40ddf9f341b3d
       c9ba860af7e0ca7ca7e9eecd0084d19c'
}
```

In CBOR encoding the result is (with line-breaks added for display purposes only):

```
A4010220012158 2065EDA5A12577C2BAE829437FE338701A10AAA375E1BB5B5DE
108DE439C08551D2258201E52ED75701163F7F9E40DDF9F341B3DC9BA860AF7E0
CA7CA7E9EECD0084D19C
```

Using SHA-256, the resulting thumbprint is:

496bd8afadf307e5b08c64b0421bf9dc01528a344a43bda88fadd1669da253ec

7. Security Considerations

A COSE Key Thumbprint will only uniquely identify a particular key if a single unambiguous COSE Key representation for that key is defined and used when computing the COSE Key Thumbprint.

If two asymmetric keys are used by different parties with different key identifiers then the COSE Key Thumbprints will still be equal since the key identifier itself is not included in the thumbprint calculation (similarly to other optional parameters in the COSE_Key structure). When the inclusion of certain optional parameters in the thumbprint calcuation is important for a given application, this specification is not the appropriate choice.

While thumbprint values are useful for identifying legitimate keys, comparing thumbprint values is not a reliable means of excluding the use of particular keys (or transformations thereof). The reason is that an attacker may supply a key that is a transformation of a key in order to have it appear to be a different key. For instance, if a legitimate RSA key uses a modulus value N and an attacker supplies a key with modulus 3*N, the modified key would still work about 1/3 of the time, but would appear to be a different key.

Producing thumbprints of symmetric keys needs to be done with care. Developers MUST ensure that the symmetric key has sufficient entropy to prevent attackers to precompute tables of symmetric keys with their corresponding hash values. This can be prevented if the symmetric key is a randomly selected key of at least 128 bit length. Thumbprints MUST NOT be used with passwords or other low-entropy secrets. If a developer is unable to determine whether all symmetric keys used in an application have sufficient entropy, then thumbprints of symmetric keys MUST NOT be used. In general, using thumbprints of symmetric keys should only be used in special applications. In most other deployment scenarios it is more appropriate to utilize a different naming scheme for key identifiers.

8. IANA Considerations

IANA is requested to add the following entry to the IANA "CWT Confirmation Methods" registry established by [RFC8747]:

* Confirmation Method Name: ckt

* Confirmation Method Description: COSE Key SHA-256 Thumbprint

* JWT Confirmation Method Name: jkt

* Confirmation Key: [[TBD1]]

* Confirmation Value Type(s): binary string

* Change Controller: IETF

* Specification Document(s): [[This document]]

Furthermore, IANA is requested to add a value to the IANA "OAuth URI" registry established with [RFC6755]:

* URN: urn:ietf:params:oauth:ckt

* Common Name: COSE Key Thumbprint URI

* Change controller: IETF

* Specification Document: [[This document]]

9. Acknowledgements

We would like to thank the authors of [RFC7638] for their work on the JSON Web Key (JWK) Thumbprint specification. This document applies JWK Thumbprints to COSE Key structures.

Additionally, we would like to thank Carsten Bormann, Ilari Liusvaara, Laurence Lundblade, Daisuke Ajitomi, Michael Richardson, Michael B. Jones, Mallory Knodel, Joel Jaeggli, and Brendan Moran for their feedback.

10. References

10.1. Normative References

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

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

[RFC6755] Campbell, B. and H. Tschofenig, "An IETF URN Sub-Namespace for OAuth", RFC 6755, DOI 10.17487/RFC6755, October 2012, <https://www.rfc-editor.org/rfc/rfc6755>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/rfc/rfc7515>.

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

[RFC8392] Jones, M., Wahlstroem, E., Erdtman, S., and H. Tschofenig, "CBOR Web Token (CWT)", RFC 8392, DOI 10.17487/RFC8392, May 2018, <https://www.rfc-editor.org/rfc/rfc8392>.

[RFC8747] Jones, M., Seitz, L., Selander, G., Erdtman, S., and H. Tschofenig, "Proof-of-Possession Key Semantics for CBOR Web Tokens (CWTs)", RFC 8747, DOI 10.17487/RFC8747, March 2020, <https://www.rfc-editor.org/rfc/rfc8747>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/rfc/rfc8949>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/rfc/rfc9052>.

[RFC9053] Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", RFC 9053, DOI 10.17487/RFC9053, August 2022, <https://www.rfc-editor.org/rfc/rfc9053>.

10.2. Informative References

[IANA.Hash.Algorithms]
"Named Information Hash Algorithm Registry", <https://www.iana.org/assignments/named-information>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/rfc/rfc5280>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/rfc/rfc6234>.

[RFC7638] Jones, M. and N. Sakimura, "JSON Web Key (JWK) Thumbprint", RFC 7638, DOI 10.17487/RFC7638, September 2015, <https://www.rfc-editor.org/rfc/rfc7638>.

[RFC9278] Jones, M. and K. Yasuda, "JWK Thumbprint URI", RFC 9278, DOI 10.17487/RFC9278, August 2022, <https://www.rfc-editor.org/rfc/rfc9278>.

[RFC9360] Schaad, J., "CBOR Object Signing and Encryption (COSE): Header Parameters for Carrying and Referencing X.509 Certificates", RFC 9360, DOI 10.17487/RFC9360, February 2023, <https://www.rfc-editor.org/rfc/rfc9360>.

Authors' Addresses

Kohei Isobe
SECOM CO., LTD.
Email: isobekohei@gmail.com

Hannes Tschofenig
Email: hannes.tschofenig@gmx.net

Orie Steele
Transmute
United States
Email: orie@transmute.industries

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 2 January 2025

K. Murchison
R. Signes
M. Horsfall
Fastmail

1 July 2024

Sieve Email Filtering: Extension for Processing Calendar Attachments draft-ietf-extra-processimip-08

Abstract

This document describes the "processcalendar" extension to the Sieve email filtering language. The "processcalendar" extension gives Sieve the ability to process machine-readable calendar data that is encapsulated in an email message using Multipurpose Internet Mail Extensions (MIME).

Status of This Memo

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

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

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

This Internet-Draft will expire on 2 January 2025.

Table of Contents

1. Introduction

Users frequently receive invites, replies, and cancellations for events, tasks, etc. via Internet mail messages. It is sometimes desirable to have such messages automatically parsed and the enclosed calendar data added to, updated on, or deleted from the user's calendars.

Typically such messages are based on the iCalendar Message-Based Interoperability Protocol (iMIP) [RFC6047]. However, sometimes the enclosed iCalendar [RFC5545] data does not include an iTIP method property (see [RFC5546], Section 1.4), or the enclosed data may be in some other machine-readable format (E.g. JSCalendar [RFC8984]).

This document defines an extension to the Sieve language [RFC5228] that enables scripts to process machine-readable calendar data that is encapsulated in an email message using MIME [RFC2045]. Specifically, this extension provides the ability to alter items on a user's calendars referenced in the encapsulated calendar data.

2. Conventions Used in This Document

Conventions for notations are as in Section 1.1 of [RFC5228], including use of the "Usage:" label for the definition of action and tagged arguments syntax.

This document uses terminology and concepts from iCalendar [RFC5545] and iTIP [RFC5546] to describe the processing of calendar data, but this extension can be used with any machine-readable calendar data format that can express similar concepts.

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

3. Capability Identifier

Sieve interpreters that implement this extension MUST have an identifier of "processcalendar" for use with the capability mechanism.

4. Process Calendar Action

```
Usage: processcalendar [ :allowpublic ]
                       [ :addresses <string-list> ]
                       [ :updatesonly / :calendarid <string> ]
                       [ :deletecancelled ]
                       [ :organizers <ext-list-name: string> ]
                       [ :outcome <variablename: string> ]
                       [ :reason <variablename: string> ]
```

The "processcalendar" action is used to parse encapsulated calendar data and perform the appropriate action based on the content. If the calendar data is malformed in any way, it MUST be ignored and no action is taken. Otherwise, based on the iTIP method (see Section 1.4 of [RFC5546]) of the message, calendar objects are created, updated, or deleted from a given calendar.

This action can be used with or without the "extlists" [RFC6134] extension. When the "extlists" extension is enabled in a script using <require "extlists">, the script can use the :organizers (Section 4.6) argument to the "processcalendar" action as described below. When the "extlists" extension is not enabled, the :organizers argument MUST NOT be used and MUST cause an error according to [RFC5228].

This action can be used with or without the "variables" [RFC5229] extension. When the "variables" extension is enabled in a script using <require "variables">, the script can use the :outcome (Section 4.7) and :reason (Section 4.8) arguments to the "processcalendar" action as described below. When the "variables" extension is not enabled, the :outcome and :reason arguments MUST NOT be used and MUST cause an error according to [RFC5228].

If a mail messages contains calendar data in multiple MIME [RFC2045] parts, this action MUST verify that the calendar data in each part are semantically equivalent to one another. If the data is found to be semantically different, the action MUST NOT process the message. Otherwise, the action MUST only process one representation of the data.

This action MUST NOT make any changes to the participant status of the recipient when processing the calendar data. The mechanism for a recipient to change their participant status to an event is out of scope for this document.

This action SHOULD remove alarms from calendar data before applying it to a calendar.

4.1. Allow Public Argument

The optional :allowpublic argument is used to tell the implementation that it can process calendar data that does not contain any ATTENDEE properties, such as iTIP messages where the METHOD is PUBLISH, or non-iTIP messages where the calendar data does not contain METHOD and/or ORGANIZER properties.

If used in conjunction with the :organizers (Section 4.6) argument, the implementation MUST NOT process non-iTIP messages.

If :allowpublic is omitted, the implementation MUST NOT process calendar data unless is it is a well-formed iTIP message and one of the recipient user’s email addresses matches the Calendar User Address (see Section 3.3.3 of [RFC5545]) of the intended target of the message, as determined by the iTIP method (see Section 1.4 of [RFC5546]) of the message:

"REPLY": Value of the "Organizer" property (see Section 3.8.4.1 of [RFC5545])

"REQUEST", "CANCEL", "ADD": Value of one of the "Attendee" properties (see Section 3.8.4.3 of [RFC5545])

The recipient user's email address matches the Calendar User Address of the target if the Calendar User Address is in the form of a mailto URI and the email address matches the "addr-spec" of the URI.

An email address is considered to belong to the recipient if it is one of:

1. an email address known by the implementation to be associated with the recipient,

2. the final envelope recipient address if it's available to the implementation, or

3. an address specified by the script writer via the :addresses (Section 4.2) argument.

4.2. Addresses Argument

The optional :addresses argument is used to specify email addresses that belong to the recipient in addition to the addresses known to the implementation.

4.3. Updates Only Argument

The optional :updatesonly argument is used to limit the messages processed to those targeting existing calendar objects only. If the message contains a new calendar object (its UID does not exist on any of the user's calendars), the implementation MUST NOT add the object to a calendar.

If :updatesonly is omitted, new calendar objects may be added to one of the user's calendars.

The :updatesonly and :calendarid (Section 4.4) arguments are incompatible with each other. It is an error if both arguments are used in the same "processcalendar" action.

4.4. Calendar ID Argument

The optional :calendarid argument specifies the identifier of the calendar onto which new calendar objects should be placed.

If :calendarid is omitted, new calendar objects will be placed on the user's "default" calendar as determined by the implementation.

The :updatesonly (Section 4.3) and :calendarid arguments are incompatible with each other. It is an error if both arguments are used in the same "processcalendar" action.

4.5. Delete Cancelled Argument

The optional :deletecancelled argument is used to tell the implementation that if it receives a cancellation message, it should remove the associated calendar object from the calendar.

If :deletecancelled is omitted, the status of the associated calendar object will be set to cancelled and will remain on the calendar.

4.6. Organizers Argument

The optional :organizers argument is used to specify an external list of email addresses from which the recipient is willing to accept public events, invites, updates, and cancellations. Implementations MUST NOT process calendar data unless is it is a well-formed iTIP message and one of the addresses in the external list matches the Calendar User Address of the "Organizer" property. An email address in the external list matches the Calendar User Address of the "Organizer" property if it is in the form of a mailto URI and the email address matches the "addr-spec" of the URI.

If :organizers is omitted, no validation of the "Organizer" property is performed.

4.7. Outcome Argument

The optional :outcome argument specifies the name of a variable into which one of the following strings specifying the outcome of the action will be stored:

* "no_action": No action was performed (E.g., the message didn't contain calendar data, or the set of provided options prevented the message from being processed).

* "added": A new calendar object was added to a calendar

* "updated": A calendar resource was updated, cancelled, or removed from the calendar.

* "error": The message would have been processed but encountered an error in doing so.

4.8. Reason Argument

The optional :reason argument specifies the name of a variable into which a string describing the reason for the outcome will be stored. If no reason for the outcome is available, implementations MUST set the variable to the empty string.

For example, an outcome of "no_action" may have a reason of "only processing updates" or an outcome of "error" may have a reason of "missing UID property".

4.9. Interaction with Other Sieve Actions

The "processcalendar" action does not cancel Sieve's implicit keep action.

The "processcalendar" action can only be executed once per script. A script MUST fail with an appropriate error if it attempts to execute two or more "processcalendar" actions.

The "processcalendar" action is incompatible with the Sieve reject and ereject [RFC5429] actions.

4.10. Examples

The following example specifies email addresses belonging to the user and the identifier of the calendar onto which to place new calendar objects:

```
require [ "processcalendar" ];

processcalendar :addresses [ "me@example.com", "alsome@example.com" ]
                :calendarid "1ea6d86b-6c7f-48a2-bed3-2a4c40ec281a";
```

The following example tells the interpreter to process flight itineraries from a particular airline:

```
require [ "processcalendar" ];

if allof (address ["from", "sender"] "airline@example.com",
          header :contains "subject" "itinerary") {
   processcalendar :allowpublic;
}
```

The following example adds headers to the message if calendar data isn't processed :

```
require [ "processcalendar", "variables", "editheader" ];

set "processcal_outcome" "no_action";
set "processcal_reason" "";

processcalendar :outcome "processcal_outcome"
                :reason "processcal_reason";

if not string :is "${processcal_outcome}" ["added", "updated"] {
   addheader "X-ProcessCal-Outcome" "${processcal_outcome}";
   addheader "X-ProcessCal-Reason" "${processcal_reason}";
}
```

5. Implementation Status

< RFC Editor: before publication please remove this section and the reference to [RFC7942] >

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

5.1. Cyrus Server

The open source Cyrus Server (http://www.cyrusimap.org/) project is a highly scalable enterprise mail system which supports Sieve email filtering at the point of final delivery. This production level Sieve implementation supports all of the mandatory functionality described in this document and is currently being updated to support the optional functionality. This implementation is freely distributable under a BSD style license from Computing Services at Carnegie Mellon University (https://www.cmu.edu/computing/).

6. Security Considerations

This document describes a method for altering an electronic calendar without user interaction. As such, unless proper precautions are undertaken, it can be used as a vector for calendar abuse.

It is critical that implementations correctly implement the behavior and restrictions described throughout this document. Security issues associated with processing unsolicited calendar data, and methods for mitigating them are discussed in [CALSPAM]. Specifically:

* Processcalendar MUST NOT process any calendar data enclosed in a message flagged as spam and/or malicious. The spamtest and virustest [RFC5235] extensions (or the header [RFC5228] test if messages are scanned outside of the Sieve interpreter) can be used to make processcalendar conditional on "safe" content.

* Processcalendar SHOULD NOT process calendar data received from a potentially malicious sender. The address and envelope [RFC5228] tests (optionally along with the extlists [RFC6134] extension) can be used to create a "deny list" and make processcalendar conditional on the sender not being a member of that list.

* Similarly, processcalendar SHOULD only process calendar data received from a known sender. The address and envelope [RFC5228] tests (optionally along with the extlists [RFC6134] extension) can be used to create an "allow list" and make processcalendar conditional on the sender being a member of that list.

* Processcalendar SHOULD NOT process calendar data received from an untrustworthy sender. Trustworthiness may depend on whether the message has a valid signature (see [RFC8551]) and/or on whether one or more of Sender Policy Framework (SPF) [RFC7208], DomainKeys Identified Mail (DKIM) Signatures [RFC6376], Domain-based Message Authentication, Reporting, and Conformance (DMARC) [RFC7489] passes or fails on the message. The mechanism by which a Sieve interpreter accesses the results of such checks is outside the scope of this document, but if the results are available in the message's header fields, the header [RFC5228] test can be used to make processcalendar conditional on the sender being trustworthy.

Additionally, if the calendar data has embedded (a.k.a. inline) attachments, implementations SHOULD:

* Decode the embedded attachment, if necessary.

* Scan the (decoded) attachment for malicious content.

If an attachment is found to be malicious, processcalendar MUST NOT process the calendar data.

7. Privacy Considerations

It is believed that this extension doesn't introduce any privacy considerations beyond those in [RFC5228].

8. IANA Considerations

8.1. Registration of Sieve Extension

This document defines the following new Sieve extension to be added to the registry defined in Section 6.2 of [RFC5228] and located here: https://www.iana.org/assignments/sieve-extensions/sieve-extensions.xhtml#sieve-extensions

IANA are requested to add a capability to the Sieve Extensions registry:

To: iana@iana.org

Subject: Registration of new Sieve extension

Capability name: processcalendar

Description: Adds the "processcalendar" action command to add and update items on a user's calendars.

RFC number: RFC XXXX

Contact address: The Sieve discussion list <sieve@ietf.org>

8.2. Registration of Sieve Action

This document defines the following new Sieve action to be added to the registry defined in Section 2.1 of [RFC9122] and located here: https://www.iana.org/assignments/sieve-extensions/sieve-extensions.xhtml#sieve-actions

IANA are requested to add a capability to the Sieve Actions registry:

To: iana@iana.org

Subject: Registration of new Sieve action

Name: processcalendar

Description: Add and update items on a user's calendars

References: RFC XXXX [RFC5229] [RFC6134]

Capabilities: "processcalendar", "variables", "extlists"

Action Interactions: This action is incompatible with "reject" and "ereject" actions

Cancels Implicit Keep? No

Can Use with IMAP Events? No

9. Acknowledgments

The authors would like to thank the following individuals for contributing their ideas and support for writing this specification: Ned Freed and Alexey Melnikov.

10. References

10.1. Normative References

[CALSPAM] The Calendaring and Scheduling Consortium, "Calendar operator practices - Guidelines to protect against calendar abuse", CC/R 18003, 2019, <https://standards.calconnect.org/csd/cc-18003.html>.

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

[RFC5228] Guenther, P., Ed. and T. Showalter, Ed., "Sieve: An Email Filtering Language", RFC 5228, DOI 10.17487/RFC5228, January 2008, <https://www.rfc-editor.org/info/rfc5228>.

[RFC5229] Homme, K., "Sieve Email Filtering: Variables Extension", RFC 5229, DOI 10.17487/RFC5229, January 2008, <https://www.rfc-editor.org/info/rfc5229>.

[RFC6047] Melnikov, A., Ed., "iCalendar Message-Based Interoperability Protocol (iMIP)", RFC 6047, DOI 10.17487/RFC6047, December 2010, <https://www.rfc-editor.org/info/rfc6047>.

[RFC6134] Melnikov, A. and B. Leiba, "Sieve Extension: Externally Stored Lists", RFC 6134, DOI 10.17487/RFC6134, July 2011, <https://www.rfc-editor.org/info/rfc6134>.

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

[RFC9122] Melnikov, A. and K. Murchison, "IANA Registry for Sieve Actions", RFC 9122, DOI 10.17487/RFC9122, June 2023, <https://www.rfc-editor.org/info/rfc9122>.

10.2. Informative References

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <https://www.rfc-editor.org/info/rfc2045>.

[RFC5235] Daboo, C., "Sieve Email Filtering: Spamtest and Virustest Extensions", RFC 5235, DOI 10.17487/RFC5235, January 2008, <https://www.rfc-editor.org/info/rfc5235>.

[RFC5429] Stone, A., Ed., "Sieve Email Filtering: Reject and Extended Reject Extensions", RFC 5429, DOI 10.17487/RFC5429, March 2009, <https://www.rfc-editor.org/info/rfc5429>.

[RFC5545] Desruisseaux, B., Ed., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, DOI 10.17487/RFC5545, September 2009, <https://www.rfc-editor.org/info/rfc5545>.

[RFC5546] Daboo, C., Ed., "iCalendar Transport-Independent Interoperability Protocol (iTIP)", RFC 5546, DOI 10.17487/RFC5546, December 2009, <https://www.rfc-editor.org/info/rfc5546>.

[RFC6376] Crocker, D., Ed., Hansen, T., Ed., and M. Kucherawy, Ed., "DomainKeys Identified Mail (DKIM) Signatures", STD 76, RFC 6376, DOI 10.17487/RFC6376, September 2011, <https://www.rfc-editor.org/info/rfc6376>.

[RFC7208] Kitterman, S., "Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1", RFC 7208, DOI 10.17487/RFC7208, April 2014, <https://www.rfc-editor.org/info/rfc7208>.

[RFC7489] Kucherawy, M., Ed. and E. Zwicky, Ed., "Domain-based Message Authentication, Reporting, and Conformance (DMARC)", RFC 7489, DOI 10.17487/RFC7489, March 2015, <https://www.rfc-editor.org/info/rfc7489>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[RFC8551] Schaad, J., Ramsdell, B., and S. Turner, "Secure/ Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification", RFC 8551, DOI 10.17487/RFC8551, April 2019, <https://www.rfc-editor.org/info/rfc8551>.

[RFC8984] Jenkins, N. and R. Stepanek, "JSCalendar: A JSON Representation of Calendar Data", RFC 8984, DOI 10.17487/RFC8984, July 2021, <https://www.rfc-editor.org/info/rfc8984>.

Appendix A. Change History (To be removed by RFC Editor before publication)

Changes since draft-ietf-sieve-processimip-07:

1. Fixed Sieve Action registration to include all associated capabilites and their references.

Changes since draft-ietf-sieve-processimip-06:

1. Fixed example that was still using :errstr rather than :reason.

2. Explicitly stated that the :updateonly and :calendarid options are incompatible with each each.

3. Explicitly stated that if :allowpublic is used with :organizers that non-iTIP messages MUST NOT be processed.

4. Updated security considerations to use "deny list" and "allow list", and to add a bullet discussing use of S/MIME, SPF, DKIM, and DMARC.

5. Updated the status of the Cyrus implementation.

6. Miscellaneous editorial changes.

Changes since draft-ietf-sieve-processimip-05:

1. Renamed :errstr to :reason and added examples.

2. Miscellaneous editorial changes.

Changes since draft-ietf-sieve-processimip-04:

1. Miscellaneous editorial changes.

Changes since draft-ietf-sieve-processimip-03:

1. Added text about multiple MIME parts containing calendar data.

2. Added text about embedded attachments to Security Considerations.

3. Added :organizers option if "extlists" is supported.

4. Miscellaneous editorial changes.

Changes since draft-ietf-sieve-processimip-02:

1. Renamed :nonitip to :allowpublic to cover both non-iTIP and METHOD:PUBLIC messages.

2. Renamed :deletecanceled to :deletecancelled to match RFC5545 language.

3. Specified that this action MUST NOT alter a recipient's participation status.

4. :errstr MUST be set to the empty string if no reason for the outcome is available.

5. Added the "Interaction with Other Sieve Actions" subsection.

6. Add Security Considerations.

7. Added action registration.

8. Added three issues for discussion.

9. Miscellaneous editorial changes.

Changes since draft-ietf-sieve-processimip-01:

1. Changed the name of the action from processimip to processcalendar.

2. The action is now independent of iMIP and is calendar data format agnostic.

3. Added examples.

Changes since draft-ietf-sieve-processimip-00:

1. No changes.

Changes since draft-murchison-sieve-processimip-00:

1. Document name change only.

Authors' Addresses

Kenneth Murchison
Fastmail US LLC
1429 Walnut Street - Suite 1201
Philadelphia, PA 19102
United States of America
Email: murch@fastmailteam.com

Ricardo Signes
Fastmail US LLC
1429 Walnut Street - Suite 1201
Philadelphia, PA 19102
United States of America
Email: rjbs@fastmailteam.com

Matthew Horsfall
Fastmail US LLC
1429 Walnut Street - Suite 1201
Philadelphia, PA 19102
United States of America
Email: alh@fastmailteam.com

IDR
Internet-Draft
Updates: 4271 (if approved)
Intended status: Standards Track
Expires: 30 January 2025

J. Snijders
Fastly
B. Cartwright-Cox
Port 179
Y. Qu
Futurewei

29 July 2024

Border Gateway Protocol 4 (BGP-4) Send Hold Timer
draft-ietf-idr-bgp-sendholdtimer-14

Abstract

This document defines the SendHoldtimer, along with the SendHoldTimer_Expires event, for the Border Gateway Protocol (BGP) Finite State Machine (FSM). Implementation of the SendHoldTimer helps overcome situations where a BGP connection is not terminated after the local system detects that the remote system is not processing BGP messages. This document specifies that the local system should close the BGP connection and not solely rely on the remote system for connection closure when the SendHoldTimer expires. This document updates RFC4271.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on 30 January 2025.

Table of Contents

1. Introduction

This document defines the SendHoldtimer, along with the SendHoldTimer_Expires event, for the Border Gateway Protocol (BGP) [RFC4271] Finite State Machine (FSM) defined in section 8.

Failure to terminate a blocked BGP connection can result in network reachability issues, and the subsequent failure to generate and deliver BGP UPDATE messages to another BGP speaker of the local system is detrimental to all participants of the inter-domain routing system. This phenomena is thought to have contributed to IP traffic blackholing events in the global Internet routing system [bgpzombies].

This specification intends to improve this situation by requiring that BGP connections be terminated if the local system has detected that the remote system cannot possibly have processed any BGP messages for the duration of the SendHoldTime. Through standardization of the aforementioned requirement, operators will benefit from consistent behavior across different BGP implementations.

BGP speakers following this specification do not rely exclusively on remote systems closing blocked connections, but will also locally close blocked connections.

2. Example of a problematic scenario

In implementations lacking the concept of a SendHoldTimer, a malfunctioning or overwhelmed remote speaker may cause data on the BGP socket in the local system to accumulate ad infinitum. This could result in forwarding failure and traffic loss, as the overwhelmed speaker continues to utilize stale routes.

An example fault state: as BGP runs over TCP [RFC9293], it is possible for a BGP speaker in the Established state to encounter a BGP speaker that is advertising a TCP Receive Window (RCV.WND) of size zero. This 0 window prevents the local system from sending KEEPALIVE, UPDATE, or any other critical BGP messages across the network socket to the remote speaker.

Generally BGP implementations have no visibility into lower-layer subsystems such as TCP or the speaker's current Receive Window size, and there is no existing BGP mechanism for such a blocked connection to be recognized. Hence BGP implementations are not able to handle this situation in a consistent fashion.

The major issue arising from a BGP speaker being unable to send a BGP message to a given remote speaker is that as a result that speaker subsequently is operating based on stale routing information. Failure of the BGP speaker to send (and thus the remote speaker to receive) BGP messages on a single BGP session can negatively impact the ability of an entire autonomous system (or even a group of autonomous systems) to converge.

This document provides a mechanism for BGP implementations to detect whether the TCP socket to a BGP speaker is progressing (data is being transmitted), or persisting in a stalled state. In case of stalled state, the BGP connection can be terminated.

3. Changes to RFC 4271 - SendHoldTimer

BGP speakers are implemented following a conceptual model "BGP Finite State Machine" (FSM), which is outlined in section 8 of [RFC4271]. This specification adds a BGP timer, SendHoldTimer, and updates the BGP FSM as follows:

3.1. Session Attributes

The following optional session attributes for each connection are added to Section 8, before "The optional session attributes support different features of the BGP functionality that have implications for the BGP FSM state transitions":

NEW

```
|     14) SendHoldTimer
|
|     15) SendHoldTime
```

The SendHoldTime determines how long a BGP speaker will stay in Established state before the TCP connection is dropped because no BGP messages can be transmitted to its peer. A BGP speaker can configure the value of the SendHoldTime for each peer independently.

3.2. Timer Event: SendHoldTimer_Expires

Another timer event is added to Section 8.1.3 of [RFC4271] as following:

NEW

```
|  Event 29: SendHoldTimer_Expires
|     Definition:  An event generated when the SendHoldTimer expires.
|
|     Status:  Optional
```

3.3. Changes to the FSM

The following changes are made to section 8.2.2 in [RFC4271].

In "OpenConfirm State", the handling of Event 26 is revised as follows:

OLD

> If the local system receives a KEEPALIVE message (KeepAliveMsg (Event 26)), the local system:
> - restarts the HoldTimer and
>
> - changes its state to Established.

NEW

> If the local system receives a KEEPALIVE message (KeepAliveMsg (Event 26)), the local system:
> - restarts the HoldTimer,
>
> - starts the SendHoldTimer if the SendHoldTime is non-zero, and
>
> - changes its state to Established.

The following paragraph is added to section 8.2.2 in "Established State", after the paragraph which ends "unless the negotiated HoldTime value is zero.":

NEW

> If the SendHoldTimer_Expires (Event 29) occurs, the local system:
>
> - (optionally) sends a NOTIFICATION message with the BGP Error Code "Send Hold Timer Expired" if the local system can determine that doing so will not delay the following actions in this paragraph,
>
> - logs an error message in the local system with the BGP Error Code "Send Hold Timer Expired",
>
> - releases all BGP resources,
>
> - sets the ConnectRetryTimer to zero,
>
> - drops the TCP connection,
>
> - increments the ConnectRetryCounter by 1,
>
> - (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and
>
> - changes its state to Idle.

> Each time the local system sends a BGP message, it restarts the SendHoldTimer unless the SendHoldTime value is zero or the negotiated HoldTime value is zero, in which cases the SendHoldTimer is stopped.
>
> The SendHoldTimer is stopped following any transition out of the Established state as part of the "release all BGP resources" action.

3.4. Changes to BGP Timers

Section 10 of [RFC4271] summarizes BGP Timers. This document adds another optional BGP timer: SendHoldTimer.

NEW

> SendHoldTime is an FSM attribute that stores the initial value for the SendHoldTimer. If SendHoldTime is non-zero then it MUST be greater than the value of HoldTime, see Section 5 for suggested default values.

4. Send Hold Timer Expired Error Handling

If the local system does not send any BGP messages within the period specified in SendHoldTime, then a NOTIFICATION message with the "Send Hold Timer Expired" Error Code MAY be sent and the BGP connection MUST be closed. Additionally, an error MUST be logged in the local system, indicating the Send Hold Timer Expired Error Code.

5. Implementation Considerations

Due to the relative rarity of the failure mode that this specification is designed to address, and also the fact that network operators may be unfamiliar with the formal specification of BGP fault detection mechanisms such as HoldTimer, it is likely that a large number of operators are unaware of the necessity of an additional mechanism such as SendHoldtimer.

Accordingly, it is RECOMMENDED that implementations of this specification enable SendHoldtimer by default, without requiring additional configuration of the BGP speaking device.

The default value of SendHoldTime for a BGP connection SHOULD be the greater of:

* 8 minutes; or

* 2 times the negotiated HoldTime

Implementations MAY make the value of SendHoldTime configurable, either globally or on a per-peer basis, within the constraints set out in Section 3.4 above.

The subcode for NOTIFICATION message "Send Hold Timer Expired" is set to 0 and is not used, no additional data is to be appended to the end of a "Send Hold Timer Expired" NOTIFICATION message.

6. Operational Considerations

When the local system recognizes a remote speaker is not processing any BGP messages for the duration of the SendHoldTime, it is likely that the local system will not be able to inform the remote peer through a NOTIFICATION message as to why the connection is being closed. This documents suggests that an attempt to send a NOTIFICATION message with the "Send Hold Timer Expired" error code is still made, if doing so will not delay closing the BGP connection. Meanwhile an error message is logged into the local system.

Other mechanisms can be used as well, for example BGP speakers SHOULD provide this reason as part of their operational state; e.g. bgpPeerLastError in the BGP MIB [RFC4273].

7. Security Considerations

This specification does not change BGP’s security characteristics. Implementing the BGP SendHoldTimer as specified in this document will enhance network resilience by terminating connections with malfunctioning or overwhelmed remote peers.

8. IANA Considerations

IANA has registered code 8 for "Send Hold Timer Expired" in the "BGP Error (Notification) Codes" sub-registry under the "Border Gateway Protocol (BGP) Parameters" registry.

9. Acknowledgements

The authors would like to thank William McCall, Theo de Raadt, John Heasley, Nick Hilliard, Jeffrey Haas, Tom Petch, Susan Hares, Keyur Patel, Ben Maddison, Claudio Jeker, and John Scudder for their helpful review of this document.

10. References

10.1. Normative References

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

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

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

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/info/rfc9293>.

10.2. Informative References

[bgpzombies] Fontugne, R., "BGP Zombies", April 2019, <https://labs.ripe.net/author/romain_fontugne/bgp-zombies/>.

[BIRD] Kubecova, K., "BIRD Internet Routing Daemon", October 2023, <https://gitlab.nic.cz/labs/bird/-/commit/bcf2327425d4dd96f381b87501cccf943bed606e>.

[frr] Lamparter, D., "bgpd: implement SendHoldTimer", May 2022, <https://github.com/FRRouting/frr/pull/11225>.

[neo-bgp] Cartwright-Cox, B., "What does bgp.tools support", August 2022, <https://bgp.tools/kb/bgp-support>.

[openbgpd] Jeker, C., "bgpd send side hold timer", December 2020, <https://marc.info/?l=openbsd-tech&m=160820754925261&w=2>.

[RFC4273] Haas, J., Ed. and S. Hares, Ed., "Definitions of Managed Objects for BGP-4", RFC 4273, DOI 10.17487/RFC4273, January 2006, <https://www.rfc-editor.org/info/rfc4273>.

Appendix A. Implementation status - RFC EDITOR: REMOVE BEFORE PUBLICATION

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in RFC 7942. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

* OpenBGPD [openbgpd]

* FRRouting [frr]

* neo-bgp (bgp.tools) [neo-bgp]

* BIRD [BIRD]

Patches to recognize error code 8 were merged into OpenBSD's and the-tcpdump-group's tcpdump implementations.

Authors' Addresses

Job Snijders
Fastly
Amsterdam
Netherlands
Email: job@fastly.com

Ben Cartwright-Cox
Port 179 Ltd
London
United Kingdom
Email: ben@benjojo.co.uk

Yingzhen Qu
Futurewei Technologies
Santa Clara,
United States
Email: yingzhen.ietf@gmail.com

Network Working Group
Internet-Draft
Updates: 9012 (if approved)
Intended status: Standards Track
Expires: 27 January 2025

S. Previdi
Huawei Technologies
C. Filsfils
K. Talaulikar, Ed.
Cisco Systems
P. Mattes
Microsoft
D. Jain
Google

26 July 2024

Advertising Segment Routing Policies in BGP
draft-ietf-idr-sr-policy-safi-06

Abstract

A Segment Routing (SR) Policy is an ordered list of segments (i.e., instructions) that represent a source-routed policy. An SR Policy consists of one or more candidate paths, each consisting of one or more segment lists. A headend may be provisioned with candidate paths for an SR Policy via several different mechanisms, e.g., CLI, NETCONF, PCEP, or BGP.

This document specifies how BGP may be used to distribute SR Policy candidate paths. It introduces a BGP SAFI to advertise a candidate path of a Segment Routing (SR) Policy and defines sub-TLVs for the Tunnel Encapsulation Attribute for signaling information about these candidate paths.

This documents updates RFC9012 with extensions to the Color Extended Community to support additional steering modes over SR Policy.

This Internet-Draft will expire on 27 January 2025.

Table of Contents

1. Introduction

Segment Routing (SR) [RFC8402] allows a headend node to steer a packet flow along a specific path. Intermediate per-path states are eliminated thanks to source routing.

The headend node is said to steer a flow into an SR Policy [RFC8402].

The packets steered into an SR Policy carry an ordered list of segments associated with that SR Policy.

[RFC9256] further details the concepts of SR Policy and steering into an SR Policy. These apply equally to the SR-MPLS and Segment Routing for IPv6 (SRv6) data-plane instantiations of Segment Routing using SR-MPLS and SRv6 Segment Identifiers (SIDs) as described in [RFC8402]. [RFC8660] describes the representation and processing of this ordered list of segments as an MPLS label stack for SR-MPLS. While [RFC8754] and [RFC8986] describe the same for SRv6 with the use of the Segment Routing Header (SRH).

The SR Policy related functionality described in [RFC9256] can be conceptually viewed as being incorporated in an SR Policy Module (SRPM). Following is a reminder of the high-level functionality of SRPM:

* Learning multiple candidate paths (CP) for an SR Policy via various mechanisms (CLI, NETCONF, PCEP, or BGP).

* Selection of the best candidate path for an SR Policy.

* Associating a Binding SID (BSID) to the selected candidate path of an SR Policy.

* Installation of the selected candidate path and its BSID in the forwarding plane.

This document specifies the use of BGP to distribute one or more of the candidate paths of an SR Policy to the headend of that policy. The document describes the functionality provided by BGP and, as appropriate, provides references for the functionality which is outside the scope of BGP (i.e. resides within SRPM on the headend node).

This document specifies a way of representing SR Policy candidate paths in BGP UPDATE messages. BGP can then be used to propagate the SR Policy candidate paths to the headend nodes in a network. The usual BGP rules for BGP propagation and best-path selection are used. At the headend of a specific policy, this will result in one or more candidate paths being installed into the "BGP table". These paths are then passed to the SRPM. The SRPM may compare them to candidate paths learned via other mechanisms and will choose one or more paths to be installed in the data plane. BGP itself does not install SR Policy candidate paths into the data plane.

This document introduces a BGP subsequent address family (SAFI) for IPv4 and IPv6 address families. In UPDATE messages of those AFI/SAFIs, the NLRI identifies an SR Policy Candidate Path while the attributes encode the segment lists and other details of that SR Policy Candidate Path.

While for simplicity we may write that BGP advertises an SR Policy, it has to be understood that BGP advertises a candidate path of an SR policy and that this SR Policy might have several other candidate paths provided via BGP (via an NLRI with a different distinguisher as defined in Section 2.1), PCEP, NETCONF, or local policy configuration.

Typically, a SR Policy Controller [RFC9256] defines the set of policies and advertises them to policy headend routers (typically ingress routers). These policy advertisements use the BGP extensions defined in this document. The policy advertisement is, in most but not all cases, tailored for a specific policy headend; such an advertisement may be sent on a BGP session to that headend and not propagated any further.

Alternatively, a router (i.e., a BGP egress router) advertises SR Policies representing paths to itself. In this case, it is possible to send the policy to each headend over a BGP session to that headend, without requiring any further propagation of the policy.

An SR Policy intended only for the receiver will, in most cases, not traverse any Route Reflector (RR, [RFC4456]) (see Section 4.2.3).

In some situations, it is undesirable for a controller or BGP egress router to have a BGP session to each policy headend. In these situations, BGP Route Reflectors may be used to propagate the advertisements. In certain other deployments, it may be necessary for the advertisement to propagate through a sequence of one or more ASes within an SR Domain (refer to Section 7 for the associated security considerations). To make this possible, an attribute needs to be attached to the advertisement that enables a BGP speaker to determine whether it is intended to be a headend for the advertised policy. This is done by attaching one or more Route Target Extended Communities to the advertisement [RFC4360].

The BGP extensions for the advertisement of SR Policies include following components:

* A Subsequent Address Family Identifier (SAFI) whose NLRIs identifies an SR Policy candidate path.

* A Tunnel Type identifier for SR Policy, and a set of sub-TLVs to be inserted into the Tunnel Encapsulation Attribute (as defined in [RFC9012]) specifying segment lists of the SR Policy candidate path, as well as other information about the SR Policy.

* One or more IPv4 address-specific format route target extended community ([RFC4360]) attached to the SR Policy Candidate Path advertisement and that indicates the intended headend of such an SR Policy Candidate Path advertisement.

The SR Policy SAFI route updates use the Tunnel Encapsulation Attribute to signal an SR Policy - which is a tunnel itself. Its usage of this attribute is hence very different from [RFC9012] where this attribute is associated with a BGP route update (e.g., for Internet or VPN routes) to specify the tunnel which is used for forwarding traffic for that route. This document does not update or change the usage of the Tunnel Encapsulation Attribute as specified in [RFC9012] for existing AFI/SAFIs as specified in that document. The details of processing of the Tunnel Encapsulation Attribute for the SR Policy SAFI are specified in Section 2.2 and Section 2.3.

The northbound advertisement of the operational state of the SR Policy Candidate Paths as part of BGP-LS [RFC9552] topology information is specified in [I-D.ietf-idr-bgp-ls-sr-policy].

The signaling of Dynamic and Composite Candidate Paths (sections 5.2 and 5.3 respectively of [RFC9256]) is outside the scope of this document.

The Color Extended Community (as defined in [RFC9012]) is used to steer traffic into an SR Policy, as described in section 8.8 of [RFC9256]. The Section 3 of this document updates [RFC9012] with modifications to the format of the Flags field of the Color Extended Community by using the two leftmost bits of that field.

1.1. Requirements Language

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

2. SR Policy Encoding

2.1. SR Policy SAFI and NLRI

A SAFI is introduced in this document: the SR Policy SAFI with codepoint 73. The AFI used MUST be IPv4(1) or IPv6(2).

The SR Policy SAFI uses the NLRI format defined as follows:

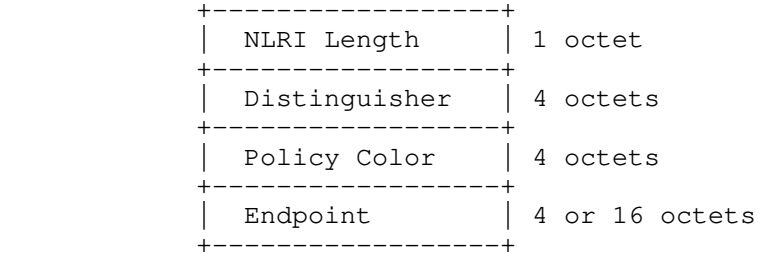


Figure 1: SR Policy SAFI Format

where:

* NLRI Length: 1 octet indicating the length expressed in bits as defined in [RFC4760]. When AFI = 1 the value MUST be 96 and when AFI = 2 the value MUST be 192.

* Distinguisher: 4-octet value uniquely identifying the policy in the context of <color, endpoint> tuple. The distinguisher has no semantic value and is solely used by the SR Policy originator to

make unique (from an NLRI perspective) both for multiple candidate paths of the same SR Policy as well as candidate paths of different SR Policies (i.e. with different segment lists) with the same Color and Endpoint but meant for different headends.

* Policy Color: 4-octet value identifying (with the endpoint) the policy. The color is used to match the color of the destination prefixes to steer traffic into the SR Policy as specified in section 8 of [RFC9256].

* Endpoint: value identifies the endpoint of a policy. The Endpoint may represent a single node or a set of nodes (e.g., an anycast address). The Endpoint is an IPv4 (4-octet) address or an IPv6 (16-octet) address according to the AFI of the NLRI. The address can be either a unicast or an unspecified address (0.0.0.0 for IPv4, :: for IPv6), known as null endpoint, as specified in section 2.1 of [RFC9256].

The color and endpoint are used to automate the steering of BGP service routes on SR Policy as described in section 8 of [RFC9256].

The NLRI containing an SR Policy candidate path is carried in a BGP UPDATE message [RFC4271] using BGP multi-protocol extensions [RFC4760] with an AFI of 1 or 2 (IPv4 or IPv6) and with a SAFI of 73. The fault management and error handling in the encoding of the NLRI is specified in Section 5.

An update message that carries the MP_REACH_NLRI or MP_UNREACH_NLRI attribute with the SR Policy SAFI MUST also carry the BGP mandatory attributes. In addition, the BGP update message MAY also contain any of the BGP optional attributes.

The next-hop network address field in SR Policy SAFI (73) updates may be either a 4-octet IPv4 address or a 16-octet IPv6 address, independent of the SR Policy AFI. The length field of the next-hop address specifies the next-hop address family. If the next-hop length is 4, then the next-hop is an IPv4 address; if the next-hop length is 16, then it is a global IPv6 address; if the next-hop length is 32, then it has a global IPv6 address followed by a link-local IPv6 address. The setting of the next-hop field and its attendant processing is governed by standard BGP procedures as described in section 3 of [RFC4760] and section 3 of [RFC2545].

It is important to note that any BGP speaker receiving a BGP message with an SR Policy NLRI, the SRPM will process it only if the NLRI is among the best paths as per the BGP best-path selection algorithm. In other words, this document leverages the existing BGP propagation and best-path selection rules. Details of the procedures are described in Section 4.

It has to be noted that if several candidate paths of the same SR Policy (endpoint, color) are signaled via BGP to a headend, then it is RECOMMENDED that each NLRI uses a different distinguisher. If BGP has installed into the BGP table two advertisements whose respective NLRIs have the same color and endpoint, but different distinguishers, both advertisements are passed to the SRPM as different candidate paths along with their respective originator information (i.e., ASN and BGP Router-ID) as described in section 2.4 of [RFC9256]. The ASN would be the ASN of the origin and the BGP Router-ID is determined in the following order:

* From the Route Origin Community [RFC4360] if present and carrying an IP Address, or

* As the BGP Originator ID [RFC4456] if present, or

* As the BGP Router-ID of the peer from which the update was received as a last resort.

The Section 2.9 of [RFC9256] specifies the selection of the active candidate path of the SR Policy by the SRPM based on the information provided to it by BGP.

2.2. SR Policy and Tunnel Encapsulation Attribute

The content of the SR Policy Candidate Path is encoded in the Tunnel Encapsulation Attribute defined in [RFC9012] using a Tunnel-Type called SR Policy Type with codepoint 15. The use of SR Policy Tunnel-type is applicable only for the AFI/SAFI pairs of (1/73, 2/73). This document specifies the use of the Tunnel Encapsulation Attribute with the SR Policy Tunnel-Type and the use of any other Tunnel-Type with the SR Policy SAFI MUST be considered malformed and handled by the "Treat-as-Withdraw" strategy [RFC7606].

The SR Policy Encoding structure is as follows:

```
SR Policy SAFI NLRI: <Distinguisher, Policy-Color, Endpoint>
Attributes:
   Tunnel Encapsulation Attribute (23)
      Tunnel Type: SR Policy (15)
          Binding SID
          SRv6 Binding SID
          Preference
          Priority
          Policy Name
          Policy Candidate Path Name
          Explicit NULL Label Policy (ENLP)
          Segment List
              Weight
              Segment
              Segment
              ...
          ...
```

Figure 2: SR Policy Encoding

where:

* SR Policy SAFI NLRI is defined in Section 2.1.

* Tunnel Encapsulation Attribute is defined in [RFC9012].

* Tunnel-Type is set to 15.

* Preference, Binding SID, SRv6 Binding SID, Priority, Policy Name, Policy Candidate Path Name, ENLP, Segment-List, Weight, and Segment sub-TLVs are defined in Section 2.4.

* Additional sub-TLVs may be defined in the future.

A Tunnel Encapsulation Attribute MUST NOT contain more than one TLV of type "SR Policy"; such updates MUST be considered malformed and handled by the "Treat-as-Withdraw" strategy [RFC7606].

BGP does not need to perform the validation of the tunnel (i.e., SR Policy) itself as indicated in section 6 of [RFC9012]. The validation of the SR Policy information that is advertised using the sub-TLVs specified in Section 2.4 is performed by the SRPM.

2.3. Applicability of Tunnel Encapsulation Attribute Sub-TLVs

The Tunnel Egress Endpoint and Color Sub-TLVs of the Tunnel Encapsulation Attribute [RFC9012] are not used for SR Policy encodings and therefore their value is irrelevant in the context of the SR Policy SAFI NLRI. If present, the Tunnel Egress Endpoint sub-TLV and the Color sub-TLV MUST be ignored by the BGP speaker and MAY be removed from the Tunnel Encapsulation Attribute during propagation.

Similarly, any other sub-TLVs (including those defined in [RFC9012]) whose applicability is not specifically defined for the SR Policy SAFI MUST be ignored by the BGP speaker and MAY be removed from the Tunnel Encapsulation Attribute during propagation.

2.4. SR Policy Sub-TLVs

This section specifies the sub-TLVs defined for encoding the information about the SR Policy Candidate Path.

Preference, Binding SID, SRv6 Binding SID, Segment-List, Priority, Policy Name, Policy Candidate Path Name, and Explicit NULL Label Policy are all optional sub-TLVs introduced for the BGP Tunnel Encapsulation Attribute [RFC9012] being defined in this section.

Weight and Segment are sub-TLVs of the Segment-List sub-TLV mentioned above.

An early version of this document included only the Binding SID sub-TLV that could be used for both SR-MPLS and SRv6 Binding SIDs. The SRv6 Binding SID TLV was introduced in later versions to support the advertisement of additional SRv6 capabilities without affecting backward compatibility for early implementations.

The fault management and error handling in the encoding of the sub-TLVs defined in this section are specified in Section 5. For the TLVs/sub-TLVs that are specified as single instance, only the first instance of that TLV/sub-TLV is used and the other instances MUST be ignored and MUST NOT considered to be malformed.

None of the sub-TLVs defined in the following sub-sections have any effect on the BGP best-path selection or propagation procedures. These sub-TLVs are not used by the BGP path selection process and are instead passed on to SRPM as SR Policy Candidate Path information for further processing described in section 2 of [RFC9256].

The use of SR Policy Sub-TLVs is applicable only for the AFI/SAFI pairs of (1/73, 2/73). Future documents may extend their applicability to other AFI/SAFI.

2.4.1. Preference Sub-TLV

The Preference sub-TLV is used to carry the Preference of an SR Policy candidate path. The contents of this sub-TLV are used by the SRPM as described in section 2.7 of [RFC9256].

The Preference sub-TLV is OPTIONAL and it MUST NOT appear more than once in the SR Policy encoding.

The Preference sub-TLV has following format:

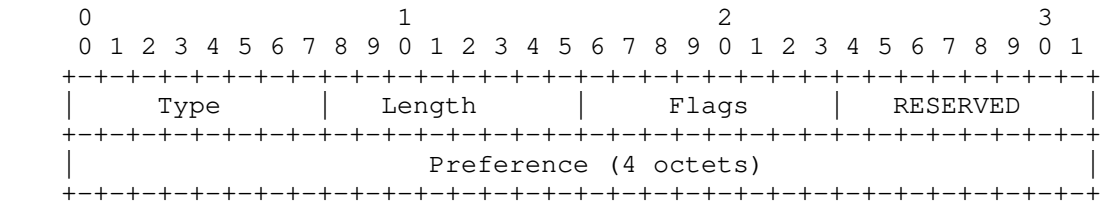


Figure 3: Preference sub-TLV

where:

* Type: 12
* Length: Specifies the length of the value field (i.e., not including Type and Length fields) in terms of octets. The value MUST be 6.
* Flags: 1 octet of flags. No flags are defined in this document. The Flags field MUST be set to zero on transmission and MUST be ignored on receipt.
* RESERVED: 1 octet of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.
* Preference: a 4-octet value indicating the Preference of the SR Policy Candidate Path as described in section 2.7 of [RFC9256].

2.4.2. Binding SID Sub-TLV

The Binding SID sub-TLV is used to signal the binding SID related information of the SR Policy candidate path. The contents of this sub-TLV are used by the SRPM as described in section 6 in [RFC9256].

The Binding SID sub-TLV is OPTIONAL and it MUST NOT appear more than once in the SR Policy encoding.

When the Binding SID sub-TLV is used to signal an SRv6 SID, the choice of its SRv6 Endpoint Behavior [RFC8986] to be instantiated is left to the headend node. It is RECOMMENDED that the SRv6 Binding SID sub-TLV defined in Section 2.4.3, that enables the specification of the SRv6 Endpoint Behavior, be used for signaling of an SRv6 Binding SID for an SR Policy candidate path.

The Binding SID sub-TLV has the following format:

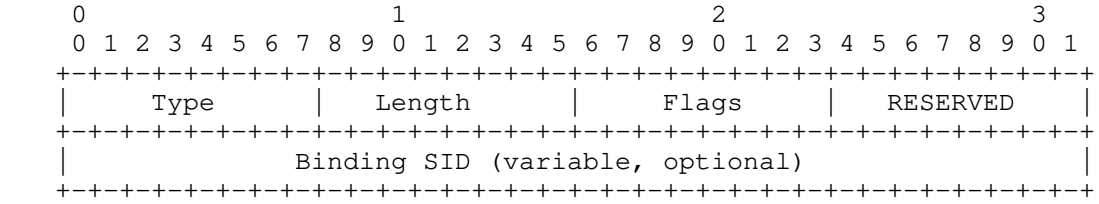


Figure 4: Binding SID sub-TLV

where:

* Type: 13

* Length: Specifies the length of the value field (i.e., not including Type and Length fields) in terms of octets. The value MUST be one of: 18 when a SRv6 BSID is present, 6 when a SR-MPLS BSID is present, or 2 when no BSID is present.

* Flags: 1 octet of flags. The following flags are defined in the registry "SR Policy Binding SID Flags" as described in Section 6.6:

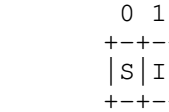


Figure 5: Binding SID Flags

where:

- S-Flag: This flag encodes the "Specified-BSID-only" behavior. It is used by SRPM as described in section 6.2.3 in [RFC9256].

- I-Flag: This flag encodes the "Drop Upon Invalid" behavior. It is used by SRPM as described in section 8.2 in [RFC9256] to define a specific SR Policy forwarding behavior. The flag indicates that the SR Policy is to perform the "drop upon invalid" behavior when no valid candidate path (CP) is available for this SR Policy. In this situation, the CP with the highest preference amongst those with the "drop upon invalid" config is made active to drop traffic steered over the SR Policy.

- The unassigned bits in the Flag octet MUST be set to zero upon transmission and MUST be ignored upon receipt.

* RESERVED: 1 octet of reserved bits. MUST be set to zero on transmission and MUST be ignored on receipt.

* Binding SID: If the length is 2, then no Binding SID is present. If the length is 6 then the Binding SID is encoded in 4 octets using the format below. Traffic Class (TC), S, and TTL (Total of 12 bits) are RESERVED and MUST be set to zero and MUST be ignored.

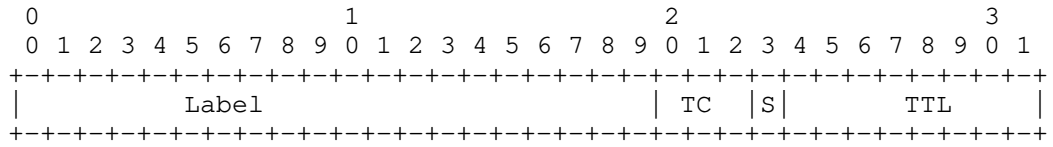


Figure 6: Binding SID Label Encoding

The Label field is validated by the SRPM, but MUST NOT contain the reserved MPLS label values (0-15). If the length is 18 then the Binding SID contains a 16-octet SRv6 SID.

2.4.3. SRv6 Binding SID Sub-TLV

The SRv6 Binding SID sub-TLV is used to signal the SRv6 Binding SID related information of an SR Policy candidate path. It enables the specification of the SRv6 Endpoint Behavior [RFC8986] to be instantiated on the headend node. The contents of this sub-TLV are used by the SRPM as described in section 6 in [RFC9256].

The SRv6 Binding SID sub-TLV is OPTIONAL. More than one SRv6 Binding SID sub-TLVs MAY be signaled in the same SR Policy encoding to indicate one or more SRv6 SIDs, each with potentially different SRv6 Endpoint Behaviors to be instantiated.

The SRv6 Binding SID sub-TLV has the following format:

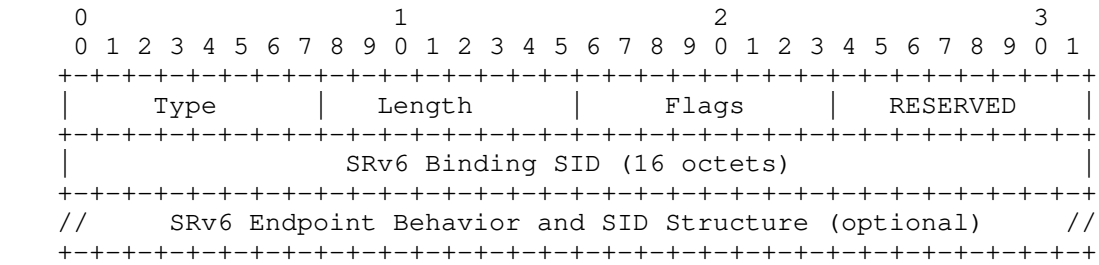


Figure 7: SRv6 Binding SID sub-TLV

where:

* Type: 20

* Length: Specifies the length of the value field (i.e., not including Type and Length fields) in terms of octets. The value MUST be 26 when the SRv6 Endpoint Behavior and SID Structure is present else it MUST be 18.

* Flags: 1 octet of flags. The following flags are defined in the registry "SR Policy Binding SID Flags" as described in Section 6.7:

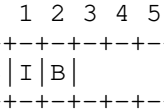


Figure 8: SRv6 Binding SID Flags

where:

- S-Flag: This flag encodes the "Specified-BSID-only" behavior. It is used by SRPM as described in section 6.2.3 in [RFC9256].

- I-Flag: This flag encodes the "Drop Upon Invalid" behavior. It is used by SRPM as described in section 8.2 in [RFC9256].

- B-Flag: This flag, when set, indicates the presence of the SRv6 Endpoint Behavior and SID Structure encoding specified in Section 2.4.4.2.4.

- The unassigned bits in the Flag octet MUST be set to zero upon transmission and MUST be ignored upon receipt.

* RESERVED: 1 octet of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.

* SRv6 Binding SID: Contains a 16-octet SRv6 SID.

* SRv6 Endpoint Behavior and SID Structure: Optional, as defined in Section 2.4.4.2.4.

2.4.4. Segment List Sub-TLV

The Segment List sub-TLV encodes a single explicit path towards the endpoint as described in section 5.1 of [RFC9256]. The Segment List sub-TLV includes the elements of the paths (i.e., segments) as well as an optional Weight sub-TLV.

The Segment List sub-TLV may exceed 255 bytes in length due to a large number of segments. A 2-octet length is thus required. According to section 2 of [RFC9012], the sub-TLV type defines the size of the length field. Therefore, for the Segment List sub-TLV, a code point of 128 or higher is used.

The Segment List sub-TLV is OPTIONAL and MAY appear multiple times in the SR Policy encoding. The ordering of Segment List sub-TLVs does not matter since each sub-TLV encodes a Segment List.

The Segment List sub-TLV contains zero or more Segment sub-TLVs and MAY contain a Weight sub-TLV.

The Segment List sub-TLV has the following format:

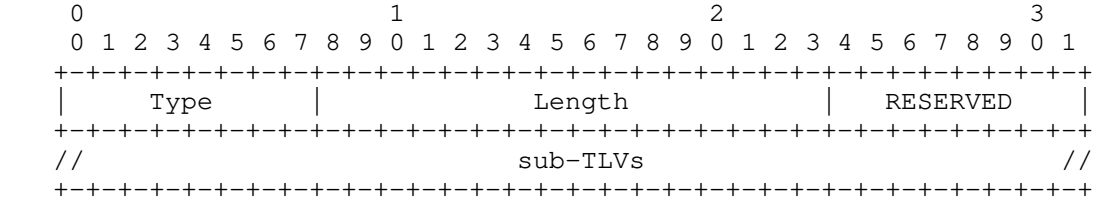


Figure 9: Segment List sub-TLV

where:

* Type: 128.

* Length: the total length (not including the Type and Length fields) of the sub-TLVs encoded within the Segment List sub-TLV in terms of octets.

* RESERVED: 1 octet of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.

* sub-TLVs currently defined:

  - An optional single Weight sub-TLV.

  - Zero or more Segment sub-TLVs.

Validation of an explicit path encoded by the Segment List sub-TLV is beyond the scope of BGP and performed by the SRPM as described in section 5 of [RFC9256].

2.4.4.1. Weight Sub-TLV

The Weight sub-TLV specifies the weight associated with a given segment list. The contents of this sub-TLV are used only by the SRPM as described in section 2.11 of [RFC9256].

The Weight sub-TLV is OPTIONAL and it MUST NOT appear more than once inside the Segment List sub-TLV.

The Weight sub-TLV has the following format:

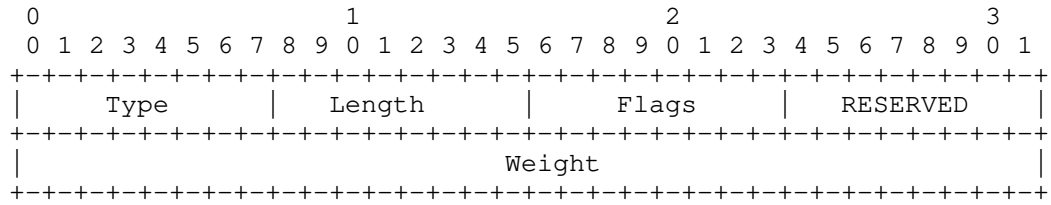


Figure 10: Weight sub-TLV

where:

* Type: 9.

* Length: Specifies the length of the value field (i.e., not including Type and Length fields) in terms of octets. The value MUST be 6.

* Flags: 1 octet of flags. No flags are defined in this document. The Flags field MUST be set to zero on transmission and MUST be ignored on receipt.

* RESERVED: 1 octet of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.

* Weight: 4 octets value indicating the weight associated with a segment list as described in section 2.11 of [RFC9256]. A weight value of zero is invalid.

2.4.4.2. Segment Sub-TLVs

A Segment sub-TLV describes a single segment in a segment list (i.e., a single element of the explicit path). One or more Segment sub-TLVs constitute an explicit path of the SR Policy candidate path. The contents of these sub-TLVs are used only by the SRPM as described in section 4 in [RFC9256].

The Segment sub-TLVs are OPTIONAL and MAY appear multiple times in the Segment List sub-TLV.

Section 4 of [RFC9256] defines several Segment Types:

Type A: SR-MPLS Label
Type B: SRv6 SID
Type C: IPv4 Prefix with optional SR Algorithm
Type D: IPv6 Global Prefix with optional SR Algorithm for SR-MPLS
Type E: IPv4 Prefix with Local Interface ID
Type F: IPv4 Addresses for link endpoints as Local, Remote pair
Type G: IPv6 Prefix and Interface ID for link endpoints as Local, Remote pair for SR-MPLS
Type H: IPv6 Addresses for link endpoints as Local, Remote pair for SR-MPLS
Type I: IPv6 Global Prefix with optional SR Algorithm for SRv6
Type J: IPv6 Prefix and Interface ID for link endpoints as Local, Remote pair for SRv6
Type K: IPv6 Addresses for link endpoints as Local, Remote pair for SRv6

The following sub-sections specify the sub-TLVs used for Segment Types A and B. The other segment types are specified in [I-D.ietf-idr-bgp-sr-segtypes-ext]. As specified in section 5.1 of [RFC9256], a mix of SR-MPLS and SRv6 segments make the segment-list invalid.

2.4.4.2.1. Segment Type A

The Type A Segment Sub-TLV encodes a single SR-MPLS SID. The format is as follows and is used to encode MPLS Label fields as specified in [RFC3032] [RFC5462].:

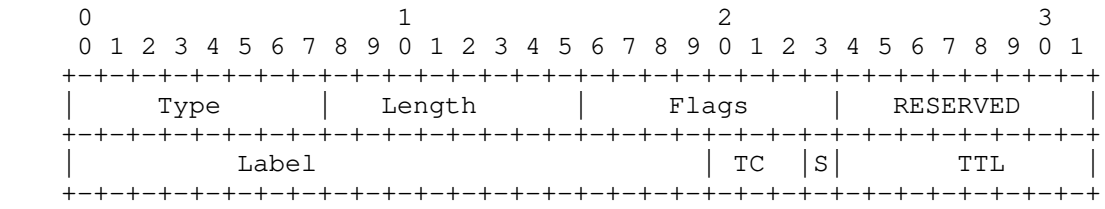


Figure 11: Type A Segment sub-TLV

where:

* Type: 1.

* Length: Specifies the length of the value field (i.e., not including Type and Length fields) in terms of octets. The value MUST be 6.

* Flags: 1 octet of flags as defined in Section 2.4.4.2.3.

* RESERVED: 1 octet of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.

* Label: 20 bits of label value.

* TC: 3 bits of traffic class.

* S: 1 bit of bottom-of-stack.

* TTL: 1 octet of TTL.

The following applies to the Type-1 Segment sub-TLV:

* The S bit MUST be zero upon transmission and MUST be ignored upon reception.

* If the originator wants the receiver to choose the TC value, it sets the TC field to zero.

* If the originator wants the receiver to choose the TTL value, it sets the TTL field to 255.

* If the originator wants to recommend a value for these fields, it puts those values in the TC and/or TTL fields.

* The receiver MAY override the originator's values for these fields. This would be determined by local policy at the receiver. One possible policy would be to override the fields only if the fields have the default values specified above.

2.4.4.2.2. Segment Type B

The Type B Segment Sub-TLV encodes a single SRv6 SID. The format is as follows:

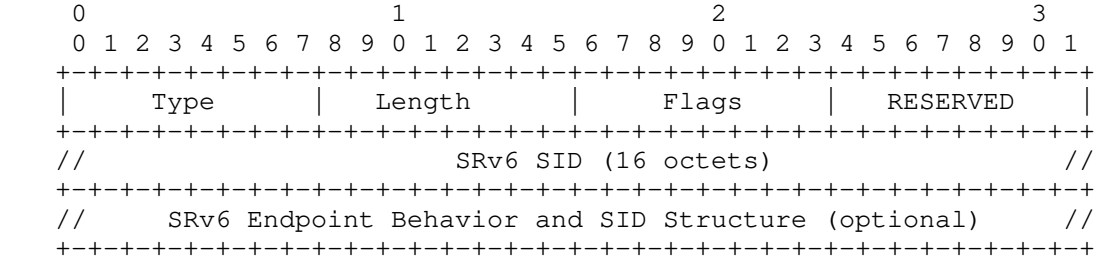


Figure 12: Type B Segment sub-TLV

where:

* Type: 13.

* Length: Specifies the length of the value field (i.e., not including Type and Length fields) in terms of octets. The value MUST be 26 when the SRv6 Endpoint Behavior and SID Structure is present else it MUST be 18.

* Flags: 1 octet of flags as defined in Section 2.4.4.2.3.

* RESERVED: 1 octet of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.

* SRv6 SID: 16 octets of IPv6 address.

* SRv6 Endpoint Behavior and SID Structure: Optional, as defined in Section 2.4.4.2.4.

The Sub-TLV code point 2 defined for the advertisement of Segment Type B in the earlier versions of this document has been deprecated to avoid backward compatibility issues.

2.4.4.2.3. Segment Flags

The Segment Types sub-TLVs described above may contain the following flags in the "Flags" field defined in Section 6.8:

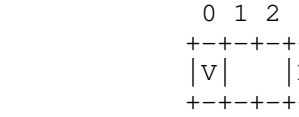


Figure 22: Segment Flags

where:

V-Flag: This flag, when set, is used by SRPM for "SID verification" as described in Section 5.1 of [RFC9256].

B-Flag: This flag, when set, indicates the presence of the SRv6 Endpoint Behavior and SID Structure encoding specified in Section 2.4.4.2.4.

The unassigned bits in the Flag octet MUST be set to zero upon transmission and MUST be ignored upon receipt.

The following applies to the Segment Flags:

* V-Flag applies to all Segment Types.

* B-Flag applies to Segment Type B. If B-Flag appears with Segment Type A it MUST be ignored.

2.4.4.2.4. SRv6 SID Endpoint Behavior and Structure

The Segment Types sub-TLVs described above MAY contain the SRv6 Endpoint Behavior and SID Structure [RFC8986] encoding as described below:

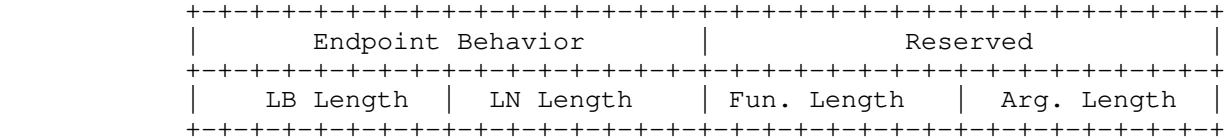


Figure 23: SRv6 SID Endpoint Behavior and Structure

where:

Endpoint Behavior: 2 octets. It carries the SRv6 Endpoint Behavior code point for this SRv6 SID as defined in section 9.2 of [RFC8986]. When set with the value 0xFFFF (i.e., Opaque), the choice of SRv6 Endpoint Behavior is left to the headend.

Reserved: 2 octets of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.

Locator Block Length: 1 octet. SRv6 SID Locator Block length in bits.

Locator Node Length: 1 octet. SRv6 SID Locator Node length in bits.

Function Length: 1 octet. SRv6 SID Function length in bits.

Argument Length: 1 octet. SRv6 SID Arguments length in bits.

The total of the locator block, locator node, function, and argument lengths MUST be less than or equal to 128.

2.4.5. Explicit NULL Label Policy Sub-TLV

To steer an unlabeled IP packet into an SR policy, it is necessary to push a label stack of one or more labels on that packet.

The Explicit NULL Label Policy (ENLP) sub-TLV is used to indicate whether an Explicit NULL Label [RFC3032] must be pushed on an unlabeled IP packet before any other labels.

If an ENLP Sub-TLV is not present, the decision of whether to push an Explicit NULL label on a given packet is a matter of local configuration.

The ENLP sub-TLV is OPTIONAL and it MUST NOT appear more than once in the SR Policy encoding.

The contents of this sub-TLV are used by the SRPM as described in section 4.1 of [RFC9256].

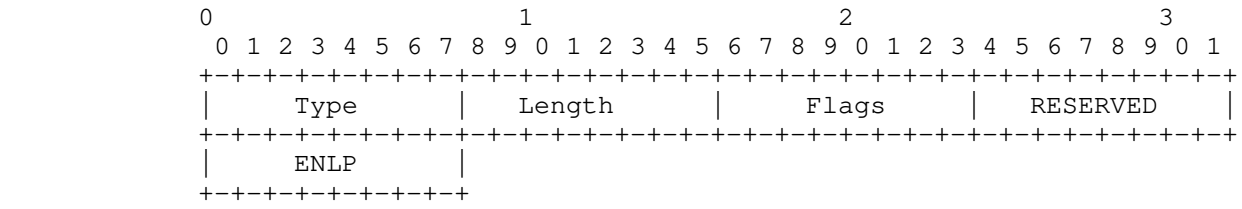


Figure 24: ELNP sub-TLV

Where:

Type: 14.

Length: Specifies the length of the value field (i.e., not including Type and Length fields) in terms of octets. The value MUST be 3.

Flags: 1 octet of flags. No flags are defined in this document. The Flags field MUST be set to zero on transmission and MUST be ignored on receipt.

RESERVED: 1 octet of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.

ENLP (Explicit NULL Label Policy): Indicates whether Explicit NULL labels are to be pushed on unlabeled IP packets that are being steered into a given SR policy. The following values have been currently defined for this field:

- 1: Push an IPv4 Explicit NULL label on an unlabeled IPv4 packet, but do not push an IPv6 Explicit NULL label on an unlabeled IPv6 packet.

- 2: Push an IPv6 Explicit NULL label on an unlabeled IPv6 packet, but do not push an IPv4 Explicit NULL label on an unlabeled IPv4 packet.

- 3: Push an IPv4 Explicit NULL label on an unlabeled IPv4 packet, and push an IPv6 Explicit NULL label on an unlabeled IPv6 packet.

- 4: Do not push an Explicit NULL label.

This field can have one of the values as specified in Section 6.10. The ENLP unassigned values may be used for future extensions and implementations SHOULD ignore the ENLP Sub-TLV with these values. The behavior signaled in this Sub-TLV MAY be

overridden by local configuration. The section 4.1 of [RFC9256] describes the behavior on the headend for the handling of the explicit null label.

2.4.6. Policy Priority Sub-TLV

An operator MAY set the Policy Priority sub-TLV to indicate the order in which the SR policies are re-computed upon topological change. The contents of this sub-TLV are used by the SRPM as described in section 2.12 of [RFC9256].

The Priority sub-TLV is OPTIONAL and it MUST NOT appear more than once in the SR Policy encoding.

The Priority sub-TLV has following format:

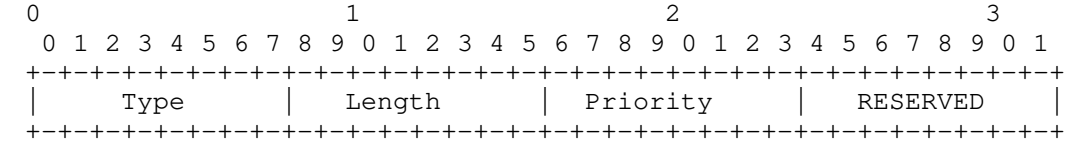


Figure 25: Priority sub-TLV

Where:

Type: 15

Length: Specifies the length of the value field (i.e., not including Type and Length fields) in terms of octets.The value MUST be 2.

Priority: a 1-octet value.

RESERVED: 1 octet of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.

2.4.7. Policy Candidate Path Name Sub-TLV

An operator MAY set the Policy Candidate Path Name sub-TLV to attach a symbolic name to the SR Policy candidate path.

Usage of Policy Candidate Path Name sub-TLV is described in section 2.6 of [RFC9256].

The Policy Candidate Path Name sub-TLV may exceed 255 bytes in length due to a long name. A 2-octet length is thus required. According to section 2 of [RFC9012], the sub-TLV type defines the size of the length field. Therefore, for the Policy Candidate Path Name sub-TLV a code point of 128 or higher is used.

It is RECOMMENDED that the size of the symbolic name for the candidate path is limited to 255 bytes. Implementations MAY choose to truncate long names to 255 bytes when signaling via BGP.

The Policy Candidate Path Name sub-TLV is OPTIONAL and it MUST NOT appear more than once in the SR Policy encoding.

The Policy Candidate Path Name sub-TLV has following format:

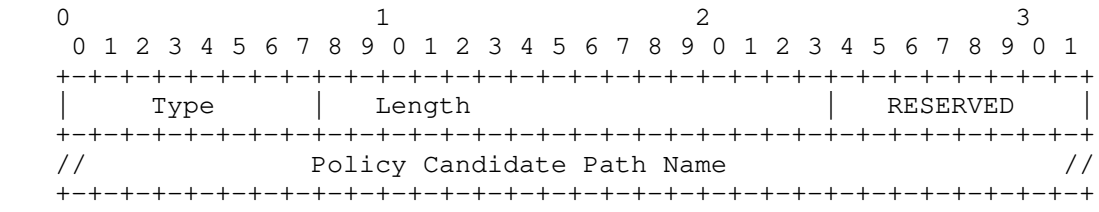


Figure 26: Policy Candidate Path Name sub-TLV

Where:

Type: 129.

Length: Specifies the length of the value field (i.e., not including Type and Length fields) in terms of octets. The value is variable.

RESERVED: 1 octet of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.

Policy Candidate Path Name: Symbolic name for the SR Policy candidate path without a NULL terminator as specified in section 2.6 of [RFC9256].

2.4.8. Policy Name Sub-TLV

An operator MAY set the Policy Name sub-TLV to associate a symbolic name with the SR Policy for which the candidate path is being advertised via the SR Policy NLRI.

Usage of Policy Name sub-TLV is described in section 2.1 of [RFC9256].

The Policy Name sub-TLV may exceed 255 bytes in length due to a long policy name. A 2-octet length is thus required. According to section 2 of [RFC9012], the sub-TLV type defines the size of the length field. Therefore, for the Policy Name sub-TLV a code point of 128 or higher is used.

It is RECOMMENDED that the size of the symbolic name for the SR Policy is limited to 255 bytes. Implementations MAY choose to truncate long names to 255 bytes when signaling via BGP.

The Policy Name sub-TLV is OPTIONAL and it MUST NOT appear more than once in the SR Policy encoding.

The Policy Name sub-TLV has following format:

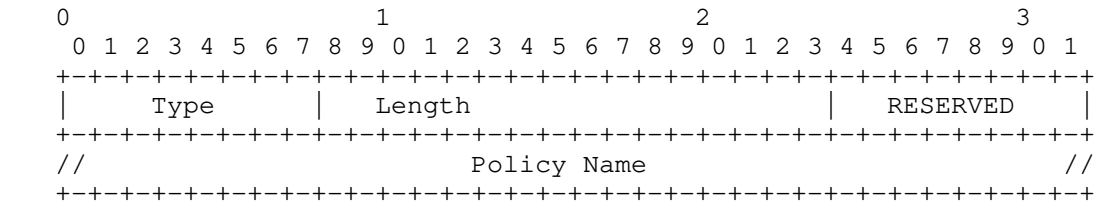


Figure 27: Policy Name sub-TLV

Where:

Type: 130

Length: Specifies the length of the value field (i.e., not including Type and Length fields) in terms of octets. The value is variable.

RESERVED: 1 octet of reserved bits. This field MUST be set to zero on transmission and MUST be ignored on receipt.

Policy Name: Symbolic name for the policy. It SHOULD be a string of printable ASCII characters, without a NULL terminator.

3. Color Extended Community

The Color Extended Community [RFC9012] is used to steer traffic corresponding to BGP routes into an SR Policy with matching color value. The Color Extended Community MAY be carried in any BGP UPDATE message whose AFI/SAFI is 1/1 (IPv4 Unicast), 2/1 (IPv6 Unicast), 1/4 (IPv4 Labeled Unicast), 2/4 (IPv6 Labeled Unicast), 1/128 (VPN-IPv4 Labeled Unicast), 2/128 (VPN-IPv6 Labeled Unicast), or 25/70 (Ethernet VPN, usually known as EVPN). Use of the Color Extended

Community in BGP UPDATE messages of other AFI/SAFIs is outside the scope of this document.

Two bits from the Flags field of the Color Extended Community are used as follows to support the requirements of Color-Only steering as specified in Section 8.8 of [RFC9256]:

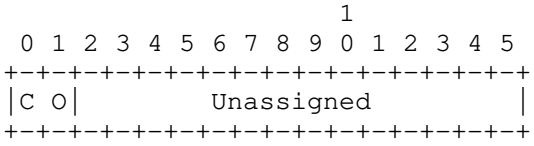


Figure 28: Color Extended Community Flags

The CO bits together form the Color-Only Type field which indicates the various matching criteria between BGP NH and SR Policy endpoint in addition to the matching of the color value. Following types are defined:

* Type 0: Specific Endpoint Match: Request match for the endpoint that is the BGP NH

* Type 1: Specific or Null Endpoint Match: Request match for either the endpoint that is the BGP NH or a null endpoint (e.g., like a default gateway)

* Type 2: Specific, Null, or Any Endpoint Match: Request match for either the endpoint that is the BGP NH or with a null or any endpoint

* Type 3: reserved for future use and SHOULD NOT be used. Upon reception, an implementation MUST treat it like Type 0.

The details of the SR Policy steering mechanisms based on these Color-Only types are specified in section 8.8 of [RFC9256].

One or more Color Extended Communities MAY be associated with a BGP route update. Sections 8.4.1, 8.5.1, and 8.8.2 of [RFC9256] specify the steering behaviors over SR Policies when multiple Color Extended Communities are associated with a BGP route.

4. SR Policy Operations

As mentioned in Section 1, BGP is not the actual consumer of an SR Policy NLRI. BGP is in charge of the origination and propagation of the SR Policy NLRI but its installation and use are outside the scope of BGP. The details of SR Policy installation and use are specified in [RFC9256].

4.1. Advertisement of SR Policies

Typically, but not limited to, an SR Policy is computed by a controller or a path computation engine (PCE) and originated by a BGP speaker on its behalf.

Multiple SR Policy NLRIs may be present with the same <color, endpoint> tuple but with different distinguishers when these SR policies are intended for different headends.

The distinguisher of each SR Policy NLRI prevents undesired BGP route selection among these SR Policy NLRIs and allows their propagation across route reflectors [RFC4456].

Moreover, one or more route targets SHOULD be attached to the advertisement, where each route target identifies one or more intended headends for the advertised SR Policy update.

If no route target is attached to the SR Policy NLRI, then it is assumed that the originator sends the SR Policy update directly (e.g., through a BGP session) to the intended receiver. In such a case, the NO_ADVERTISE community [RFC1997] MUST be attached to the SR Policy update (see further details in Section 4.2.3).

4.2. Reception of an SR Policy NLRI

On reception of an SR Policy NLRI, a BGP speaker first determines if it is valid as described in Section 4.2.1 and then performs the decision process for selection of the best route (Section 9.1 of [RFC4271]). The key difference from the base BGP decision process is that BGP does not download the selected best routes of SR Policy SAFI into the forwarding and instead considers them "usable" for passing on to the SRPM for further processing as described in Section 4.2.2. The selected best route is "propagated" (Section 9.1.3 of [RFC4271]) as described in Section 4.2.3 irrespective of its "usability" by the local router.

4.2.1. Validation of an SR Policy NLRI

When a BGP speaker receives an SR Policy NLRI from a neighbor it MUST first perform validation based on the following rules in addition to the validation described in Section 5:

* The SR Policy NLRI MUST include a distinguisher, color, and endpoint field which implies that the length of the NLRI MUST be either 12 or 24 octets (depending on the address family of the endpoint).

* The SR Policy update MUST have either the NO_ADVERTISE community or at least one route target extended community in IPv4-address format or both. If a router supporting this specification receives an SR Policy update with no route target extended communities and no NO_ADVERTISE community, the update MUST be considered as malformed.

* The Tunnel Encapsulation Attribute MUST be attached to the BGP Update and MUST have a Tunnel Type TLV set to SR Policy (codepoint is 15).

A router that receives an SR Policy update that is not valid according to these criteria MUST treat the update as malformed and the SR Policy candidate path MUST NOT be passed to the SRPM.

4.2.2. Eligibility for Local Use of an SR Policy NLRI

An SR Policy NLRI update without any route target extended community but having the NO_ADVERTISE community is considered usable.

If one or more route targets are present, then at least one route target MUST match the BGP Identifier of the receiver for the update to be considered usable. The BGP Identifier is defined in [RFC4271] as a 4-octet IPv4 address. Therefore, the route target extended community MUST be of the same format.

If one or more route targets are present and none matches the local BGP Identifier, then, while the SR Policy NLRI is valid, it is not usable on the receiver node.

When the SR Policy tunnel type includes any sub-TLV that is unrecognized or unsupported, the update SHOULD NOT be considered usable. An implementation MAY provide an option for ignoring unsupported sub-TLVs.

Once BGP on the receiving node has determined that the SR Policy NLRI is usable, it passes the SR Policy candidate path to the SRPM. Note that, along with the candidate path details, BGP also passes the originator information for breaking ties in the candidate path selection process as described in section 2.4 of [RFC9256].

When an update for an SR Policy NLRI results in its becoming unusable, BGP MUST delete its corresponding SR Policy candidate path from the SRPM.

The SRPM applies the rules defined in section 2 of [RFC9256] to determine whether the SR Policy candidate path is valid and to select the active candidate path for a given SR Policy.

4.2.3. Propagation of an SR Policy

SR Policy NLRIs that have the NO_ADVERTISE community attached to them MUST NOT be propagated.

By default, a BGP node receiving an SR Policy NLRI MUST NOT propagate it to any EBGP neighbor. An implementation MAY provide an explicit configuration to override this and enable the propagation of valid SR Policy NLRIs to specific EBGP neighbors where the SR domain comprises multiple-ASes within a single service provider domain (see Section 7 for details).

A BGP node advertises a received SR Policy NLRI to its IBGP neighbors according to normal IBGP propagation rules.

By default, a BGP node receiving an SR Policy NLRI SHOULD NOT remove route target extended community before propagation. An implementation MAY provide support for configuration to filter and/or remove route target extended community before propagation.

A BGP node MUST NOT alter the SR Policy information carried in the Tunnel Encapsulation Attribute during propagation.

5. Error Handling and Fault Management

This section describes the error handling actions, as described in [RFC7606], that are to be performed for the handling of the BGP update messages for BGP SR Policy SAFI.

A BGP Speaker MUST perform the following syntactic validation of the SR Policy NLRI to determine if it is malformed. This includes the validation of the length of each NLRI and the total length of the MP_REACH_NLRI and MP_UNREACH_NLRI attributes. It also includes the validation of the consistency of the NLRI length with the AFI and the endpoint address as specified in Section 2.1.

When the error determined allows for the router to skip the malformed NLRI(s) and continue the processing of the rest of the update message, then it MUST handle such malformed NLRIs as 'Treat-as-withdraw'. In other cases, where the error in the NLRI encoding results in the inability to process the BGP update message (e.g. length related encoding errors), then the router SHOULD handle such malformed NLRIs as 'AFI/SAFI disable' when other AFI/SAFI besides SR Policy are being advertised over the same session. Alternately, the router MUST perform 'session reset' when the session is only being used for SR Policy or when it 'AFI/SAFI disable' action is not possible.

The validation of the TLVs/sub-TLVs introduced in this document and defined in their respective sub-sections of Section 2.4 MUST be performed to determine if they are malformed or invalid. The validation of the Tunnel Encapsulation Attribute itself and the other TLVs/sub-TLVs specified in Section 13 of [RFC9012] MUST be done as described in that document. In case of any error detected, either at the attribute or its TLV/sub-TLV level, the "treat-as-withdraw" strategy MUST be applied. This is because an SR Policy update without a valid Tunnel Encapsulation Attribute (comprising of all valid TLVs/sub-TLVs) is not usable.

An SR Policy update that is determined to be not valid, and therefore malformed, based on rules described in Section 4.2.1 MUST be handled by the "treat-as-withdraw" strategy.

The validation of the individual fields of the TLVs/sub-TLVs defined in Section 2.4 are beyond the scope of BGP as they are handled by the SRPM as described in the individual TLV/sub-TLV sub-sections. A BGP implementation MUST NOT perform semantic verification of such fields nor consider the SR Policy update to be invalid or not usable based on such validation.

An implementation SHOULD log any errors found during the above validation for further analysis.

6. IANA Considerations

This document uses code point allocations from the following existing registries:

* Subsequent Address Family Identifiers (SAFI) Parameters registry

* BGP Tunnel Encapsulation Attribute Tunnel Types registry under the BGP Tunnel Encapsulation registry

* BGP Tunnel Encapsulation Attribute sub-TLVs registry under the BGP Tunnel Encapsulation registry

* Color Extended Community Flags registry under the BGP Tunnel Encapsulation registry

This document also requests the creation of the following new registries:

* SR Policy Segment List Sub-TLVs under the BGP Tunnel Encapsulation registry

* SR Policy Binding SID Flags under the BGP Tunnel Encapsulation registry

* SR Policy SRv6 Binding SID Flags under the BGP Tunnel Encapsulation registry

* SR Policy Segment Flags under the BGP Tunnel Encapsulation registry

* Color Extended Community Color-Only Types registry under the BGP Tunnel Encapsulation registry

* SR Policy ENLP Values under the Segment Routing registry

6.1. Subsequent Address Family Identifiers (SAFI) Parameters

This document introduces a SAFI in the registry "Subsequent Address Family Identifiers (SAFI) Parameters" that has been assigned a code point by IANA. The entry needs to be updated as follows:

Code Point	Description	Reference
73	SR Policy SAFI	This document

Table 1: BGP SAFI Code Point

6.2. BGP Tunnel Encapsulation Attribute Tunnel Types

This document introduces a Tunnel-Type in the registry "BGP Tunnel Encapsulation Attribute Tunnel Types" that has been assigned a codepoint by IANA. The entry needs to be updated as follows:

Code Point	Description	Reference
15	SR Policy	This document

Table 2: Tunnel Type Code Point

6.3. BGP Tunnel Encapsulation Attribute sub-TLVs

This document defines sub-TLVs in the registry "BGP Tunnel Encapsulation Attribute sub-TLVs" that have been assigned code points by IANA as follows via the early allocation process which needs to be made permanent:

Code Point	Description	Reference
12	Preference sub-TLV	This document
13	Binding SID sub-TLV	This document
14	ENLP sub-TLV	This document
15	Priority sub-TLV	This document
20	SRv6 Binding SID sub-TLV	This document
128	Segment List sub-TLV	This document
129	Policy Candidate Path Name sub-TLV	This document
130	Policy Name sub-TLV	This document

Table 3: BGP Tunnel Encapsulation Attribute Code Points

6.4. Color Extended Community Flags

This document defines the use of 2 bits in the registry called "Color Extended Community Flags" under the "BGP Tunnel Encapsulation" registry that have been assigned by IANA via the early allocation process to form the Color-Only Types field which needs to be made permanent:

Bit Position	Description	Reference
0-1	Color-only Types Field	This document

Table 4: Color Extended Community Flag Bits

6.5. SR Policy Segment List Sub-TLVs

This document requests the creation of a new registry called "SR Policy Segment List Sub-TLVs" under the "BGP Tunnel Encapsulation" registry. The allocation policy of this registry is "Standards Action" according to [RFC8126].

Following initial Sub-TLV codepoints are assigned by this document:

Value	Description	Reference
0	Reserved	This document
1	Segment Type A sub-TLV	This document
2	Deprecated	This document
3-8	Unassigned	
9	Weight sub-TLV	This document
10	Deprecated	This document
11	Deprecated	This document
12	Deprecated	This document
13	Segment Type B sub-TLV	This document
14-255	Unassigned	

Table 5: SR Policy Segment List Code Points

6.6. SR Policy Binding SID Flags

This document requests the creation of a new registry called "SR Policy Binding SID Flags" under the "BGP Tunnel Encapsulation" registry. The allocation policy of this registry is "Standards Action" according to [RFC8126].

The following flags are defined:

Bit	Description	Reference
0	Specified-BSID-Only Flag (S-Flag)	This document
1	Drop Upon Invalid Flag (I-Flag)	This document
2-7	Unassigned	

Table 6: SR Policy Binding SID Flags

6.7. SR Policy SRv6 Binding SID Flags

This document requests the creation of a new registry called "SR Policy SRv6 Binding SID Flags" under the "BGP Tunnel Encapsulation" registry. The allocation policy of this registry is "Standards Action" according to [RFC8126].

The following flags are defined:

Bit	Description	Reference
0	Specified-BSID-Only Flag (S-Flag)	This document
1	Drop Upon Invalid Flag (I-Flag)	This document
2	SRv6 Endpoint Behavior & SID Structure Flag (B-Flag)	This document
3-7	Unassigned	

Table 7: SR Policy SRv6 Binding SID Flags

6.8. SR Policy Segment Flags

This document requests the creation of a new registry called "SR Policy Segment Flags" under the "BGP Tunnel Encapsulation" registry. The allocation policy of this registry is "Standards Action" according to [RFC8126].

The following flags are defined:

Bit	Description	Reference
0	Segment Verification Flag (V-Flag)	This document
1-2	Unassigned	
3	SRv6 Endpoint Behavior & SID Structure Flag (B-Flag)	This document
4-7	Unassigned	

Table 8: SR Policy Segment Flags

6.9. Color Extended Community Color-Only Types

This document requests the creation of a new registry called "Color Extended Community Color-Only Types" under the "BGP Tunnel Encapsulation" registry for assignment of codepoints (values 0 through 3) in the Color-Only Type field of the Color Extended Community Flags field. The allocation policy of this registry is "Standards Action" according to [RFC8126].

The following types are defined:

Type	Description	Reference
0	Specific Endpoint Match	This document
1	Specific or Null Endpoint Match	This document
2	Specific, Null, or Any Endpoint Match	This document
3	Unassigned	This document

Table 9: Color Extended Community Color-Only Types

6.10. SR Policy ENLP Values

Note to IANA (RFC editor to remove this before publication): The new registry creation request below is also present in the draft-ietf-pce-segment-routing-policy-cp. IANA is requested to process the registry creation via the first of these two documents to reach publication stage and the authors of the other document would update the IANA considerations suitably.

This document requests IANA to maintain a new registry under "Segment Routing Parameters" registry group with the allocation policy of "Standards Action" [RFC8126]. The new registry is called "SR Policy ENLP Values" and contains the codepoints allocated to the "ENLP" field defined in Section 2.4.5. The registry contains the following codepoints, with initial values, to be assigned by IANA with the reference set to this document:

Code Point	Description
0	Reserved (not to be used)
1	Push an IPv4 Explicit NULL label on an unlabeled IPv4 packet, but do not push an IPv6 Explicit NULL label on an unlabeled IPv6 packet
2	Push an IPv6 Explicit NULL label on an unlabeled IPv6 packet, but do not push an IPv4 Explicit NULL label on an unlabeled IPv4 packet
3	Push an IPv6 Explicit NULL label on an unlabeled IPv6 packet, and push an IPv4 Explicit NULL label on an unlabeled IPv4 packet
4	Do not push an Explicit NULL label
5-255	Unassigned

7. Security Considerations

The security mechanisms of the base BGP security model apply to the extensions described in this document as well. See the Security Considerations section of [RFC4271] for a discussion of BGP security. Also, refer to [RFC4272] and [RFC6952] for analysis of security issues for BGP.

The BGP SR Policy extensions specified in this document enable traffic engineering and service programming use-cases within an SR domain as described in [RFC9256]. SR operates within a trusted SR domain [RFC8402] and its security considerations also apply to BGP sessions when carrying SR Policy information. The SR Policies

distributed by BGP are expected to be used entirely within this trusted SR domain which comprises a single AS or multiple ASes/ domains within a single provider network. Therefore, precaution is necessary to ensure that the SR Policy information advertised via BGP sessions is limited to nodes in a secure manner within this trusted SR domain. BGP peering sessions for address-families other than SR Policy SAFI may be set up to routers outside the SR domain. The isolation of BGP SR Policy SAFI peering sessions may be used to ensure that the SR Policy information is not advertised by accident or error to an EBGP peering session outside the SR domain.

Additionally, it may be considered that the export of SR Policy information, as described in this document, constitutes a risk to confidentiality of mission-critical or commercially sensitive information about the network (more specifically endpoint/node addresses, SR SIDs, and the SR Policies deployed). BGP peerings are not automatic and require configuration; thus, it is the responsibility of the network operator to ensure that only trusted nodes (that include both routers and controller applications) within the SR domain are configured to receive such information.

8. Manageability Considerations

The specification of BGP models is an ongoing work based on [I-D.ietf-idr-bgp-model] and its future extensions are expected to cover the SR Policy SAFI. Existing BGP operational procedures also apply to the SAFI specified in this document. The management, operations, and monitoring of BGP speakers and the SR Policy SAFI sessions between them are not very different from other BGP sessions and can be managed using the same data models.

The YANG model for the operation and management of SR Policies [I-D.ietf-spring-sr-policy-yang] reports the SR Policies provisioned via BGP SR Policy SAFI along with their operational states.

9. Acknowledgments

The authors of this document would like to thank Shyam Sethuram, John Scudder, Przemyslaw Krol, Alex Bogdanov, Nandan Saha, Bruno Decraene, Gurusiddesh Nidasesi, Kausik Majumdar, Zafar Ali, Swadesh Agarwal, Jakob Heitz, Viral Patel, Peng Shaofu, Cheng Li, Martin Vigoureux, John Scudder, Vincent Roca, Brian Haberman, Mohamed Boucadair, Shunwan Zhuang, Andrew Alston, Jeffrey (Zhaohui) Zhang, Nagendra Nainar, Rajesh Melarcode Venkateswaran, Nat Kao, Boris Hassanov, and Vincent Roca for their comments and review of this document. The authors would like to thank Susan Hares for her detailed shepherd review that helped in improving the document.

10. Contributors

Eric Rosen
Juniper Networks
US

Email: erosen@juniper.net

Arjun Sreekantiah
Cisco Systems
US

Email: asreekan@cisco.com

Acee Lindem
Cisco Systems
US

Email: acee@cisco.com

Siva Sivabalan
Cisco Systems
US

Email: msiva@cisco.com

Imtiyaz Mohammad
Arista Networks
India

Email: imtiyaz@arista.com

Gaurav Dawra
Cisco Systems
US

Email: gdawra.ietf@gmail.com

Peng Shaofu
ZTE Corporation
China

Email: peng.shaofu@zte.com.cn

Steven Lin
Calix
USA

Email: steven.lin@calix.com

11. References

11.1. Normative References

[RFC1997] Chandra, R., Traina, P., and T. Li, "BGP Communities Attribute", RFC 1997, DOI 10.17487/RFC1997, August 1996, <https://www.rfc-editor.org/info/rfc1997>.

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

[RFC2545] Marques, P. and F. Dupont, "Use of BGP-4 Multiprotocol Extensions for IPv6 Inter-Domain Routing", RFC 2545, DOI 10.17487/RFC2545, March 1999, <https://www.rfc-editor.org/info/rfc2545>.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, DOI 10.17487/RFC3032, January 2001, <https://www.rfc-editor.org/info/rfc3032>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

[RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, DOI 10.17487/RFC4360, February 2006, <https://www.rfc-editor.org/info/rfc4360>.

[RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <https://www.rfc-editor.org/info/rfc4760>.

[RFC5462] Andersson, L. and R. Asati, "Multiprotocol Label Switching (MPLS) Label Stack Entry: "EXP" Field Renamed to "Traffic Class" Field", RFC 5462, DOI 10.17487/RFC5462, February 2009, <https://www.rfc-editor.org/info/rfc5462>.

[RFC7606] Chen, E., Ed., Scudder, J., Ed., Mohapatra, P., and K. Patel, "Revised Error Handling for BGP UPDATE Messages", RFC 7606, DOI 10.17487/RFC7606, August 2015, <https://www.rfc-editor.org/info/rfc7606>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

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

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

[RFC8660] Bashandy, A., Ed., Filsfils, C., Ed., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with the MPLS Data Plane", RFC 8660, DOI 10.17487/RFC8660, December 2019, <https://www.rfc-editor.org/info/rfc8660>.

[RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <https://www.rfc-editor.org/info/rfc8754>.

[RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <https://www.rfc-editor.org/info/rfc8986>.

[RFC9012] Patel, K., Van de Velde, G., Sangli, S., and J. Scudder, "The BGP Tunnel Encapsulation Attribute", RFC 9012, DOI 10.17487/RFC9012, April 2021, <https://www.rfc-editor.org/info/rfc9012>.

[RFC9256] Filsfils, C., Talaulikar, K., Ed., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", RFC 9256, DOI 10.17487/RFC9256, July 2022, <https://www.rfc-editor.org/info/rfc9256>.

11.2. Informational References

[I-D.ietf-idr-bgp-ls-sr-policy]
Previdi, S., Talaulikar, K., Dong, J., Gredler, H., and J. Tantsura, "Advertisement of Segment Routing Policies using BGP Link-State", Work in Progress, Internet-Draft, draft-ietf-idr-bgp-ls-sr-policy-04, 20 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-bgp-ls-sr-policy-04>.

[I-D.ietf-idr-bgp-model]
Jethanandani, M., Patel, K., Hares, S., and J. Haas, "YANG Model for Border Gateway Protocol (BGP-4)", Work in Progress, Internet-Draft, draft-ietf-idr-bgp-model-17, 5 July 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-bgp-model-17>.

[I-D.ietf-idr-bgp-sr-segtypes-ext]
Talaulikar, K., Filsfils, C., Previdi, S., Mattes, P., and D. Jain, "Segment Routing Segment Types Extensions for BGP SR Policy", Work in Progress, Internet-Draft, draft-ietf-idr-bgp-sr-segtypes-ext-03, 4 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-bgp-sr-segtypes-ext-03>.

[I-D.ietf-spring-sr-policy-yang]
Raza, S. K., Saleh, T., Shunwan, Z., Voyer, D., Durrani, M., Matsushima, S., and V. P. Beeram, "YANG Data Model for Segment Routing Policy", Work in Progress, Internet-Draft, draft-ietf-spring-sr-policy-yang-03, 4 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-spring-sr-policy-yang-03>.

[RFC4272] Murphy, S., "BGP Security Vulnerabilities Analysis", RFC 4272, DOI 10.17487/RFC4272, January 2006, <https://www.rfc-editor.org/info/rfc4272>.

[RFC4456] Bates, T., Chen, E., and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)", RFC 4456, DOI 10.17487/RFC4456, April 2006, <https://www.rfc-editor.org/info/rfc4456>.

[RFC6952] Jethanandani, M., Patel, K., and L. Zheng, "Analysis of BGP, LDP, PCEP, and MSDP Issues According to the Keying and Authentication for Routing Protocols (KARP) Design Guide", RFC 6952, DOI 10.17487/RFC6952, May 2013, <https://www.rfc-editor.org/info/rfc6952>.

[RFC9552] Talaulikar, K., Ed., "Distribution of Link-State and Traffic Engineering Information Using BGP", RFC 9552, DOI 10.17487/RFC9552, December 2023, <https://www.rfc-editor.org/info/rfc9552>.

Authors' Addresses

Stefano Previdi
Huawei Technologies
Italy
Email: stefano@previdi.net

Clarence Filsfils
Cisco Systems
Brussels
Belgium
Email: cfilsfil@cisco.com

Ketan Talaulikar (editor)
Cisco Systems
India
Email: ketant.ietf@gmail.com

Paul Mattes
Microsoft
One Microsoft Way
Redmond, WA 98052
United States of America
Email: pamattes@microsoft.com

Dhanendra Jain
Google
Email: dhanendra.ietf@gmail.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 2 February 2025

D. Farinacci
lispers.net

1 August 2024

LISP Distinguished Name Encoding
draft-ietf-lisp-name-encoding-10

Abstract

This draft defines how to use the AFI=17 Distinguished Names in LISP.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on 2 February 2025.

Copyright Notice

Table of Contents

1. Introduction

The LISP architecture and protocols [RFC9300] introduces two new numbering spaces, Endpoint Identifiers (EIDs) and Routing Locators (RLOCs) which are intended to replace most use of IP addresses on the Internet. To provide flexibility for current and future applications, these values can be encoded in LISP control messages using a general syntax that includes Address Family Identifier (AFI).

The length of the value field, which represents the address encoding, is implicit in the type of address that follows. For AFI 17, a Distinguished Name can be encoded. A name can be a variable length field so the length cannot be determined solely from the AFI value 17. This draft defines a termination character, an 8-bit value of 0 to be used as a string terminator so the length can be determined.

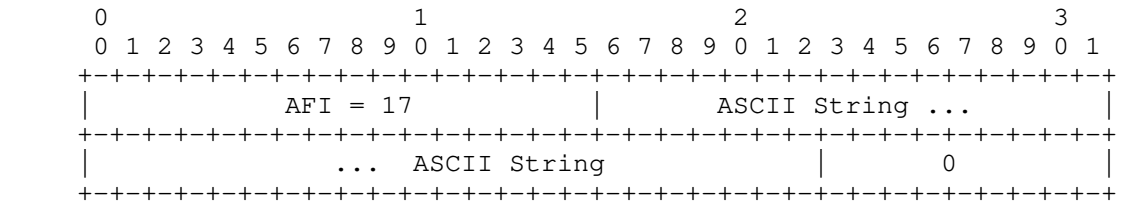
LISP Distinguished Names are useful when encoded either in EID-Records or RLOC-records in LISP control messages. As EIDs, they can be registered in the mapping system to find resources, services, or simply used as a self-documenting feature that accompany other address specific EIDs. As RLOCs, Distinguished Names, along with RLOC specific addresses and parameters, can be used as labels to identify equipment type, location, or any self-documenting string a registering device desires to convey.

2. Definition of Terms

Address Family Identifier (AFI): a term used to describe an address encoding in a packet. An address is family currently defined for IPv4 or IPv6 addresses. See [IANA-ADDRESS-FAMILY-REGISTRY] for details on other types of information that can be AFI encoded.

3. Distinguished Name Format

An AFI=17 Distinguished Name is encoded as:



The string of characters are encoded in the ASCII character-set definition [RFC0020].

When Distinguished Names are encoded for EIDs, the EID Mask-Len length of the EIDs as they appear in EID-Records for all LISP control messages [RFC9301] is the length of the string in bits (including the null 0 octet). Where Distinguished Names are encoded anywhere else (i.e., nested in LCAF encodings [RFC8060]), then any length field is the length of the ASCII string including the null 0 octet in units of octets.

4. Mapping System Lookups for Distinguished Name EIDs

Distinguished Name EID lookups MUST carry as an EID Mask-Len length equal to the length of the name string. This instructs the mapping system to do either an exact match or longest match lookup.

If the Distinguished Name EID is registered with the same length as the length in a Map-Request, the Map-Server (when configured for proxy Map-Replying) returns an exact match lookup with the same EID Mask-Len length. If a less specific name is registered, then the Map-Server returns the registered name with the registered EID Mask-Len length.

For example, if the registered EID name is "ietf" with EID Mask-Len of 40 bits (the length of string "ietf" plus the null octet is 5 octets), and a Map-Request is received for EID name "ietf.lisp" with an EID Mask-Len of 80 bits, the Map-Server will return EID "ietf" with length of 40 bits.

5. Example Use-Cases

This section identifies three specific use-cases examples for the Distinguished Name format. Two are used for an EID encoding and one for an RLOC-record encoding. When storing public keys in the mapping system, as in [I-D.ietf-lisp-ecdsa-auth], a well-known format for a public-key hash can be encoded as a Distinguished Name. When street location to GPS coordinate mappings exist in the mapping system, as

in [I-D.ietf-lisp-geo], the street location can be a free form ASCII representation (with whitespace characters) encoded as a Distinguished Name. An RLOC that describes an Ingress or Egress Tunnel Router (xTR) behind a NAT device can be identified by its router name, as in [I-D.farinacci-lisp-lispers-net-nat]. In this case, Distinguished Name encoding is used in NAT Info-Request messages after the EID-prefix field of the message.

6. Name Collision Considerations

When a Distinguished Name encoding is used to format an EID, the uniqueness and allocation concerns are no different than registering IPv4 or IPv6 EIDs to the mapping system. See [RFC9301] for more details. Also, the use-case documents specified in Section 5 of this specification provide allocation recommendations for their specific uses.

It is RECOMMENDED that each use-case register their Distinguished Names with a unique Instance-ID. For any use-cases which require different uses for Distinguish Names within an Instance-ID MUST define their own Instance-ID and structure syntax for the name registered to the Mapping System. See the encoding procedures in [I-D.ietf-lisp-vpn] for an example.

7. Security Considerations

There are no security considerations.

8. IANA Considerations

The code-point values in this specification are already allocated in [IANA-ADDRESS-FAMILY-REGISTRY].

9. Sample LISP Distinguished Name (DN) Deployment Experience

Practical implementations of the LISP Distinguished Name specification have been running in production networks for some time. The following sections provide some examples of its usage and lessons gathered out of this experience.

9.1. DNs to Advertise Specific Device Roles or Functions

In a practical implementation of [I-D.ietf-lisp-site-external-connectivity] on LISP deployments, routers running as Proxy Egress Tunnel Routers (Proxy-ETRs) register their role with the Mapping System in order to attract traffic destined for external networks. Practical implementations of this functionality make use of a Distinguished Name as an EID to identify

the Proxy-ETR role in a Map-Registration.

In this case all Proxy-ETRs supporting this function register a common Distinguished Name together with their own offered locator. The Mapping-System aggregates the locators received from all Proxy-ETRs as a common locator-set that is associated with this DN EID. The Distinguished Name in this case serves as a common reference EID that can be requested (or subscribed as per [RFC9437]) to dynamically gather this Proxy-ETR list as specified in the LISP Site External Connectivity document.

The use of a Distinguished Name in this case provides descriptive information about the role being registered and allows the Mapping System to form locator-sets associated to specific role. These locator-sets can be distributed on-demand based on using the shared DN as EID. It also allows the network admin and the Mapping System to selectively choose what roles and functions can be registered and distributed to the rest of the participants in the network.

9.2. DNs to Drive xTR On-Boarding Procedures

Following the LISP reliable transport [I-D.ietf-lisp-map-server-reliable-transport], ETRs that plan to switch to using a reliable transport to hold registrations first need to start with traditional UDP registrations. The UDP registration allows the Map-Server to perform basic authentication of the ETR and create the necessary state to permit the reliable transport session to be established (e.g., establish a passive open on TCP port 4342 and add the ETR RLOC to the list allowed to establish a session).

In the basic implementation of this process, the ETRs need to wait until local mappings are available and ready to be registered with the Mapping System. Furthermore, when the mapping system is distributed, the ETR requires to have one specific mapping ready to be registered with each one of the relevant Map-Servers. This process may delay the onboarding of ETRs with the Mapping System so that they can switch to using a reliable transport. This can also lead to generating unnecessary signaling as a reaction to certain triggers like local port flaps and device failures.

The use of dedicated name registrations allows driving this initial ETR on-boarding on the Mapping System as a deterministic process that does not depend on the availability of other mappings. It also provides more stability to the reliable transport session to survive through transient events.

In practice, LISP deployments use dedicated Distinguished Names that are registered as soon as xTRs come online with all the necessary Map-Servers in the Mapping System. The mapping with the dedicated DN together with the RLOCs of each Egress Tunnel Router (ETR) in the locator-set is used to drive the initial UDP registration and also to keep the reliable transport state stable through network condition changes. On the Map-Server, these DN registrations facilitate setting up the necessary state to onboard new ETRs rapidly and in a more deterministic manner.

9.3. DNs for NAT-Traversal

The open source lispers.net NAT-Traversal implementation [I-D.farinacci-lisp-lispers-net-nat] has had 10 years of deployment experience using Distinguished Names for documenting xTRs versus Re-encapsulating Tunnel Router (RTRs) as they appear in an locator-set.

9.4. DNs for Self-Documenting RLOC Names

The open source lispers.net implementation has had 10 years of self-documenting RLOC names in production and pilot environments. The RLOC name is encoded with the RLOC address in Distinguished Name format.

9.5. DNs used as EID Names

The open source lispers.net implementation has had 10 years of deployment experience allowing xTRs to register EIDs as Distinguished Names. The LISP Mapping System can be used as a DNS proxy for Name-to-EID-address or Name-to-RLOC-address mappings. The implementation also supports Name-to-Public-Key mappings to provide key management features in [I-D.ietf-lisp-ecdsa-auth].

10. References

10.1. Normative References

[RFC0020] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <https://www.rfc-editor.org/info/rfc20>.

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

[RFC8060] Farinacci, D., Meyer, D., and J. Snijders, "LISP Canonical Address Format (LCAF)", RFC 8060, DOI 10.17487/RFC8060, February 2017, <https://www.rfc-editor.org/info/rfc8060>.

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

[RFC9300] Farinacci, D., Fuller, V., Meyer, D., Lewis, D., and A. Cabellos, Ed., "The Locator/ID Separation Protocol (LISP)", RFC 9300, DOI 10.17487/RFC9300, October 2022, <https://www.rfc-editor.org/info/rfc9300>.

[RFC9301] Farinacci, D., Maino, F., Fuller, V., and A. Cabellos, Ed., "Locator/ID Separation Protocol (LISP) Control Plane", RFC 9301, DOI 10.17487/RFC9301, October 2022, <https://www.rfc-editor.org/info/rfc9301>.

[RFC9437] Rodriguez-Natal, A., Ermagan, V., Cabellos, A., Barkai, S., and M. Boucadair, "Publish/Subscribe Functionality for the Locator/ID Separation Protocol (LISP)", RFC 9437, DOI 10.17487/RFC9437, August 2023, <https://www.rfc-editor.org/info/rfc9437>.

10.2. Informative References

[I-D.farinacci-lisp-lispers-net-nat]
Farinacci, D., "lispers.net LISP NAT-Traversal Implementation Report", Work in Progress, Internet-Draft, draft-farinacci-lisp-lispers-net-nat-08, 17 June 2024, <https://datatracker.ietf.org/doc/html/draft-farinacci-lisp-lispers-net-nat-08>.

[I-D.ietf-lisp-ecdsa-auth]
Farinacci, D. and E. Nordmark, "LISP Control-Plane ECDSA Authentication and Authorization", Work in Progress, Internet-Draft, draft-ietf-lisp-ecdsa-auth-12, 19 February 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-lisp-ecdsa-auth-12>.

[I-D.ietf-lisp-geo]
Farinacci, D., "LISP Geo-Coordinate Use-Cases", Work in Progress, Internet-Draft, draft-ietf-lisp-geo-08, 21 July 2024, <https://datatracker.ietf.org/api/v1/doc/document/draft-ietf-lisp-geo/>.

[I-D.ietf-lisp-map-server-reliable-transport]
Venkatachalapathy, B., Portoles-Comeras, M., Lewis, D., Kouvelas, I., and C. Cassar, "LISP Map Server Reliable Transport", Work in Progress, Internet-Draft, draft-ietf-lisp-map-server-reliable-transport-04, 21 April 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-lisp-map-server-reliable-transport-04>.

[I-D.ietf-lisp-site-external-connectivity]
Jain, P., Moreno, V., and S. Hooda, "LISP Site External Connectivity", Work in Progress, Internet-Draft, draft-ietf-lisp-site-external-connectivity-00, 27 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-lisp-site-external-connectivity-00>.

[I-D.ietf-lisp-vpn]
Moreno, V. and D. Farinacci, "LISP Virtual Private Networks (VPNs)", Work in Progress, Internet-Draft, draft-ietf-lisp-vpn-12, 19 September 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-lisp-vpn-12>.

[IANA-ADDRESS-FAMILY-REGISTRY]
IANA, "IANA Address Family Numbers Registry", https://www.iana.org/assignments/address-family-numbers/, December 2023.

Appendix A. Acknowledgments

The author would like to thank the LISP WG for their review and acceptance of this draft. And a special thank you goes to Marc Portoles for moving this document through the process and providing deployment experience samples.

Appendix B. Document Change Log

B.1. Changes to draft-ietf-lisp-name-encoding-10

* Submitted August 2024.

* Change to "EID mask-len" per Roman Danyliw's comments.

B.2. Changes to draft-ietf-lisp-name-encoding-09

* Submitted July 2024.

* Added editorial suggestions from Acee Lindem.

B.3. Changes to draft-ietf-lisp-name-encoding-08

* Submitted June 2024.

* Made changes to reflect AD Jim Guichard's comments.

B.4. Changes to draft-ietf-lisp-name-encoding-07

* Submitted May 2024.

* Changed document status to "Proposed Standard" and some rewording per Alberto for the pETR use-case section.

B.5. Changes to draft-ietf-lisp-name-encoding-06

* Submitted April 2024.

* Add Deployment Experience section for standards track requirements.

* Update references.

B.6. Changes to draft-ietf-lisp-name-encoding-05

* Submitted December 2023.

* Update IANA AFI reference.

B.7. Changes to draft-ietf-lisp-name-encoding-04

* Submitted December 2023.

* More comments from Alberto. Change to standard spellings throughout.

* Add RFC 2119 boilerplate.

* Update reference RFC1700 to RFC3232.

B.8. Changes to draft-ietf-lisp-name-encoding-03

* Submitted December 2023.

* Address comments from Alberto, document shepherd.

* Update references.

B.9. Changes to draft-ietf-lisp-name-encoding-02

* Submitted August 2023.
* Update references and document expiry timer.

B.10. Changes to draft-ietf-lisp-name-encoding-01

* Submitted February 2023.
* Update references and document expiry timer.
* Change 68**.bis references to proposed RFC references.

B.11. Changes to draft-ietf-lisp-name-encoding-00

* Submitted August 2022.
* Move individual submission to LISP WG document.

B.12. Changes to draft-farinacci-lisp-name-encoding-15

* Submitted July 2022.
* Added more clarity text about how using VPNs (instance-ID encoding) addresses name collisions from multiple use-cases.
* Update references and document expiry timer.

B.13. Changes to draft-farinacci-lisp-name-encoding-14

* Submitted May 2022.
* Update references and document expiry timer.

B.14. Changes to draft-farinacci-lisp-name-encoding-13

* Submitted November 2021.
* Update references and document expiry timer.

B.15. Changes to draft-farinacci-lisp-name-encoding-12

* Submitted May 2021.
* Update references and document expiry timer.

B.16. Changes to draft-farinacci-lisp-name-encoding-11

* Submitted November 2020.
* Made changes to reflect working group comments.
* Update references and document expiry timer.

B.17. Changes to draft-farinacci-lisp-name-encoding-10

* Submitted August 2020.
* Update references and document expiry timer.

B.18. Changes to draft-farinacci-lisp-name-encoding-09

* Submitted March 2020.
* Update references and document expiry timer.

B.19. Changes to draft-farinacci-lisp-name-encoding-08

* Submitted September 2019.
* Update references and document expiry timer.

B.20. Changes to draft-farinacci-lisp-name-encoding-07

* Submitted March 2019.
* Update referenes and document expiry timer.

B.21. Changes to draft-farinacci-lisp-name-encoding-06

* Submitted September 2018.
* Update document expiry timer.

B.22. Changes to draft-farinacci-lisp-name-encoding-05

* Submitted March 2018.
* Update document expiry timer.

B.23. Changes to draft-farinacci-lisp-name-encoding-04

* Submitted September 2017.

* Update document expiry timer.

B.24. Changes to draft-farinacci-lisp-name-encoding-03

* Submitted March 2017.

* Update document expiry timer.

B.25. Changes to draft-farinacci-lisp-name-encoding-02

* Submitted October 2016.

* Add a comment that the distinguished-name encoding is restricted to ASCII character encodings only.

B.26. Changes to draft-farinacci-lisp-name-encoding-01

* Submitted October 2016.

* Update document timer.

B.27. Changes to draft-farinacci-lisp-name-encoding-00

* Initial draft submitted April 2016.

Author's Address

Dino Farinacci
lispers.net
San Jose, CA
United States of America
Email: farinacci@gmail.com

LSR Working Group
Internet-Draft
Updates: 5029 (if approved)
Intended status: Standards Track
Expires: 30 January 2025

T. Li
Juniper Networks

29 July 2024

Revision to Registration Procedures for IS-IS Neighbor Link-Attribute Bit Values
draft-ietf-lsr-labv-registration-03

Abstract

RFC 5029, "Definition of an IS-IS Link Attribute Sub-TLV", defines a registry for "IS-IS Neighbor Link-Attribute Bit Values". This document changes the registration procedure for that registry from "Standards Action" to "Expert Review". This document updates RFC5029.

Table of Contents

1. Introduction

[RFC5029] defines the "IS-IS Neighbor Link-Attribute Bit Values" registry and specifies a registration procedure of "Standards Action". In practice, this registration procedure is unnecessarily restrictive, as it prevents allocation of bits to experimental protocols, which in turn increases the risk of conflicts introduced by use of unregistered code points (so-called "code point squatting").

Accordingly, this document changes the registration procedure for the registry, as described in Section 2.

2. IANA Considerations

The registration procedure for the "IS-IS Neighbor Link-Attribute Bit Values" registry is changed to be "Expert Review". General guidance for the designated experts is defined in [RFC7370] and more specific guidance can be found in [RFC5029].

3. Security Considerations

This document does not affect the security issues discussed in RFC 5029.

4. Acknowledgements

The author would like to thank John Scudder for his contributions.

5. References

5.1. Normative References

[RFC5029] Vasseur, JP. and S. Previdi, "Definition of an IS-IS Link Attribute Sub-TLV", RFC 5029, DOI 10.17487/RFC5029, September 2007, <https://www.rfc-editor.org/info/rfc5029>.

5.2. Informative References

[RFC7370] Ginsberg, L., "Updates to the IS-IS TLV Codepoints Registry", RFC 7370, DOI 10.17487/RFC7370, September 2014, <https://www.rfc-editor.org/info/rfc7370>.

Author's Address

Tony Li
Juniper Networks
Email: tony.li@tony.li

MOPS
Internet-Draft
Intended status: Informational
Expires: 12 January 2025

L. Giuliano
Juniper Networks
C. Lenart
Verizon
R. Adam
GEANT

11 July 2024

TreeDN- Tree-based CDNs for Live Streaming to Mass Audiences
draft-ietf-mops-treedn-05

Abstract

As Internet audience sizes for high-interest live events reach unprecedented levels and bitrates climb to support 4K/8K/Augmented Reality (AR), live streaming can place a unique type of stress upon network resources. TreeDN is a tree-based CDN architecture designed to address the distinctive scaling challenges of live streaming to mass audiences. TreeDN enables operators to offer Replication-as-a-Service (RaaS) at a fraction the cost of traditional, unicast-based CDNs- in some cases, at no additional cost to the infrastructure. In addition to efficiently utilizing network resources to deliver existing multi-destination traffic, this architecture also enables new types of content and use cases that previously were not possible or economically viable using traditional CDN approaches. Finally, TreeDN is a decentralized architecture and a democratizing technology in the way that it makes content distribution more accessible to more people by dramatically reducing the costs of replication.

Status of This Memo

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

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

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

This Internet-Draft will expire on 12 January 2025.

Table of Contents

1. Problem Statement

Live streaming to mass audiences can impose unique demands on network resources. For example, live sporting events broadcast over the Internet to end users has much lower tolerance for long playout buffers than typical on-demand video streaming. Viewers of live sporting events have long been conditioned by broadcast television to expect to see the content in real time, with only very short buffers for broadcast delays to prevent profanity and other objectionable

content from making on the air (the "seven-second delay" [BROADCAST-DELAY]). With micro-betting, even this 5-10 second delay can be too long. By comparison, when watching on-demand movies, an extra one- or two-minute playout buffer tends to be perfectly acceptable for viewers. If playout buffers for live sports are that long, viewers run the risk of being alerted to the game winning score from text messages from friends or cheers from the bar across the street, minutes before they view it themselves.

Another unique characteristic of live streaming is join rate. While on-demand video streaming can consume massive amounts of network resources, the viewing rates tend to be smooth and predictable. Service Providers observe gradual levels of traffic increases over the evening hours corresponding to prime-time viewing habits. By comparison, viewing rates of live video streams can more closely resemble a step function with much less predictability as mass audiences of viewers tune in to watch the game at the same time.

Previous efforts at more efficient network replication of multi-destination traffic have experienced mixed success in terms of adoption. IP multicast is widely deployed on financial networks, video distribution networks, L3VPN networks and certain enterprises. But most of these deployments are restricted to "walled-garden" networks. Multicast over the global Internet has failed to gain traction, as only a very small portion of the Internet is multicast-enabled at this time.

TreeDN is the result of the evolution of network-based replication mechanisms based on lessons learned from what has and has not worked well in the past. TreeDN addresses the fundamental issues of what has hindered multicast from adoption on the global Internet and enables service providers the opportunity to deliver new Replication-as-a-Service (RaaS) offerings to content providers, while more efficiently utilizing network resources by eliminating duplicated traffic, and thus, improving the experience of end users. Further, by more efficiently supporting multi-destination traffic, TreeDN is an architecture that can enable new types of content, such as Augmented Reality (AR) live streaming to mass audiences, that previously weren’t possible or economically viable on the Internet due to the inefficiencies of unicast.

1.1. Requirements Language

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

2. Applicability

While the primary use case mentioned throughout this document is live streaming of multimedia content (audio, video, AR, real-time telemetry data), the TreeDN architecture can provide efficient delivery for any content that needs to be replicated and delivered to multiple destinations. For example, large software file updates (eg, OS upgrades) that need to be delivered to many end users in a very short window of time can cause significant strain on network resources. Using TreeDN, this use case be handled much more efficiently by the network.

3. Multicast Challenges in the Past

The following issues have been the primary challenges for deployment of IP multicast over the global Internet:

* The "All or Nothing" Problem: IP multicast requires every layer-3 hop between source and receivers to be multicast-enabled. To achieve ubiquitous availability on the global Internet, this essentially means nearly every interface on every router and firewall between all end hosts must support a multicast routing protocol like Protocol Independent Multicast - Sparse Mode (PIM-SM) [RFC7761] or Multipoint Label Distribution Protocol (mLDP) [RFC6388]. This requirement creates a bar to deployment that is practically impossible to overcome.

* The "It’s Too Complex" Problem: operators have long complained that multicast routing protocols like PIM-SM are simply too complex, making it costly to design, configure, manage and troubleshoot IP multicast in the network.

* The "Chicken and Egg" Problem: there’s not much multicast content because there’s not much of a multicast-enabled audience, but there’s not much of a multicast-enabled audience because there’s not much multicast content.

TreeDN is the evolution of network-based replication based on lessons learned over decades and is designed to address the problems listed above.

4. TreeDN Architecture

TreeDN leverages a simplified model for multicast deployment combined with network overlays to deliver traffic to receiving hosts on unicast-only networks. With network overlays, a service can be achieved and delivered to end users while recognizing and tolerating the practical realities of what is possible over a network as diverse as the global Internet. That is, the replication service is available to users and applications across the global Internet regardless of what protocols may exist in the underlying networks that constitute the underlay.

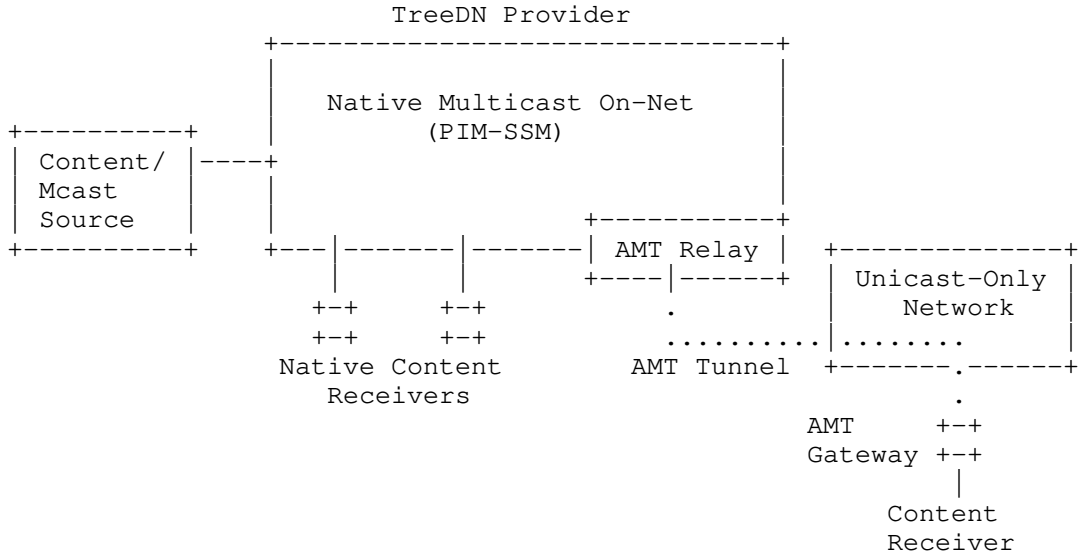


Figure 1: TreeDN Provider Example

4.1. TreeDN Overlays

One overlay technology that TreeDN leverages is Automatic Multicast Tunneling (AMT) [RFC7450]. With AMT, end hosts on unicast-only networks (AMT Gateways) can dynamically build tunnels to routers on the multicast-enabled part of the network (AMT Relays) and receive multicast streams. The AMT Gateway is a thin software client which typically sits on the receiving end host and initiates the tunnel at an AMT Relay, which is a tunnel server that typically sits at the border of the multicast network. AMT allows any end host on the Internet to receive multicast content regardless of whether their local provider supports multicast (aka, "off-net receivers"), which addresses the "All or Nothing" Problem. Links and devices that do not support multicast are simply tunneled over- they no longer

present a barrier to the overall replication service for end users. Those networks that do deploy and support multicast, as well as the content providers that serve up multicast content, are able to enjoy the benefits of efficient replication and delivery. Further, these benefits can serve as incentives for operators who do not yet support multicast to enable it on their networks, a key benefit of incremental deployment described in section 4.3 of [RFC9049]. Once the cost of carrying duplicated unicast tunnels is perceived by those operators to exceed the cost of deploying multicast, they are more likely to enable multicast on their networks. In this way, TreeDN effectively supports incremental deployment in a way that was not previously possible with traditional (non-overlay) multicast networking. Finally, AMT also addresses the "Chicken and Egg" Problem, as all end hosts on the global Internet that have access to an AMT Relay are capable of becoming audience members.

To support receiving on both native and non-native networks, receiving hosts can first attempt to join the traffic natively and, if no multicast traffic is received, fallback to AMT. This fallback mechanism can be handled by the application layer.

In addition to AMT, other overlay technologies like Locator/ID Separation Protocol (LISP) [RFC9300] can be utilized to deliver content from multicast-enabled networks to end hosts that are separated by portions of the network (at the last/middle/first mile) that do not support multicast.

4.2. TreeDN Native On-Net

Networks that support multicast provide the native on-net component of TreeDN. The primary requirement of the native on-net is to support Source-Specific Multicast (SSM) [RFC4607]. PIM-SSM, which is merely a subset of PIM-SM, is the multicast routing protocol typically used in SSM. However, any multicast routing protocol capable of supporting SSM can be used as a TreeDN native on-net, such as mLDP, Global Table Multicast (GTM) [RFC7716] and BGP-based Multicast [I-D.ietf-bess-bgp-multicast], or even BGP-MVPN [RFC6513] for those operators who carry the global routing table in a VRF. Likewise, any data plane technology that supports SSM, including BIER [RFC8279] and SR-P2MP [I-D.ietf-spring-sr-replication-segment] can be used.

The key benefit of SSM as the native on-net component of TreeDN is that it radically simplifies the control plane needed to support replication in the network. This benefit addresses the "It's Too Complex" Problem. Most of the complexity of multicast is eliminated in SSM, which reduces the cost of deploying and operating a multicast network. Further rationale for this SSM-only approach can be found in Any-Source Multicast (ASM) Deprecation [RFC8815].

5. Replication-as-a-Service (RaaS)

Content providers have traditionally used CDNs to distribute content that needs to be delivered to large audiences, essentially outsourcing the task of replication to CDN providers. Most CDNs utilize unicast delivery, as multicast is not an option due to its lack of general availability on the global Internet. TreeDN is a CDN architecture that leverages tree-based replication to more efficiently utilize network resources to deliver simultaneous multi-destination traffic. By leveraging overlay networking to address the "All or Nothing" and "Chicken and Egg" Problems and SSM to address the "It's Too Complex" Problem, TreeDN avoids the practical issues that previously prevented multicast from being a viable option for CDN providers.

TreeDN has several advantages over traditional unicast-based CDN approaches. First, the TreeDN functionality can be delivered entirely by the existing network infrastructure. Specifically, for operators with routers that support AMT natively, multicast traffic can be delivered directly to end users without the need for specialized CDN devices, which typically are servers that need to be racked, powered, cooled and connected to ports on routers that could otherwise have been consumed by paying customers. In this way, SPs can offer new RaaS functionality to content providers at potentially zero additional cost in new equipment.

Additionally, TreeDN is an open architecture that leverages mature, IETF-specified and widely implemented network protocols. TreeDN also requires far less coordination between the content provider and the CDN operator. That is, there are no storage requirements for the data, nor group-key management issues since a TreeDN provider merely forwards packets. A TreeDN provider simply needs to have enough accounting data (eg, traffic data, number of AMT tunnels, etc) to properly bill customers for the service. By contrast, traditional unicast-based CDNs often incorporate proprietary, non-interoperable technologies and require significant coordination between the content provider and the CDN to handle such things as file storage, data protection and key-management.

TreeDN introduces a deployment model that requires new considerations for transport layer mechanisms that are frequently relied upon by traditional unicast-based CDNs. A discussion on these considerations and differences can be found in section 7.

6. Decentralization/Democratization of Content Sourcing

TreeDN is an inherently decentralized architecture. This reduces the cost for content sourcing, as any host connected to a multicast-enabled network, or on a source-capable overlay, can send out a single data stream that can be reached by an arbitrarily large audience. By effectively reducing to zero the marginal cost of reaching each additional audience member, from the perspective of the source, TreeDN democratizes content sourcing on the Internet.

7. Transport Layer-Related Differences between TreeDN and Traditional CDNs

The focus of this document is on the network layer components that comprise the TreeDN architecture. This section introduces some of the key transport layer-related differences between TreeDN and traditional unicast-based CDNs that should be taken into consideration when deploying TreeDN-based services. In many cases, these issues are more related to TCP-UDP differences than unicast-multicast differences, thus UDP-based solutions can be leveraged to address most gaps. The aim of this section is to point to some of the existing work to address these gaps, as well as suggest further work that could be undertaken within the IETF. Further details of these transport layer mechanisms are beyond the scope of this document.

7.1. Integration with Unicast

Since SSM inherently implies unidirectional traffic flows from one to many, mechanisms that rely on bidirectional communication between receivers and the content provider, such as bespoke advertising, telemetry data from receivers detailing end user experience, distribution of decryption keys, switching to higher/lower bandwidth streams, etc, are not well suited to SSM delivery. As such, separate unicast streams between receivers and content providers may be used for this type of "out-of-band" functions while SSM is used to deliver the actual content of interest. These "out-of-band" unicast streams SHOULD use the same congestion control and authentication mechanisms that are used today for mass audience unicast delivery. Generally speaking, this hybrid unicast-multicast approach is best handled by the application layer and further detail is beyond the scope of this document.

7.2. Reliability, Adaptive Bitrate and Congestion Control

Traditional unicast-based CDNs frequently rely on HTTPS over TCP transport and are thus able to leverage the granularity of TCP-based mechanisms for reliability, congestion control and adaptive bitrate streaming. But this granularity comes at a cost of sending a separate datastream to each viewer. Multicast transmissions usually employ UDP, which inherently lacks many of the aforementioned benefits of TCP, but can scale much better for mass audiences of simultaneous viewers. Forward Error Correction (FEC) is a mechanism that has demonstrated full recovery for up to 5% packet loss and interruptions up to 400ms for multicast datastreams in [EUMETSAT-TERRESTRIAL]. NACK-Oriented Reliable Multicast (NORM) [RFC5740] leverages FEC-based repair and other Reliable Multicast Transport building blocks to provide end-to-end reliable transport over multicast networks.

Section 4.1 of [RFC8085] describes how a sender can distribute data across multiple multicast source-group channels so that each receiver can join the most appropriate channels for its own reception rate capability, thus providing adaptive bitrate capabilities for multicast streams. DVB MABR [DVB-MABR] and MAUD [MAUD] extensively describe an architecture that enables reliability and dynamic bitrate adaptation.

TreeDN deployments MUST follow the congestion control guidelines described in Section 4.1.4.2 of [RFC7450]. Multicast applications being distributed over TreeDN deployments SHOULD implement congestion control for its data transmission as described in Section 4.1 in [RFC8085]. The AMT gateway SHOULD use the topologically closes AMT relay. Section 3.1 of [RFC8777] describes a set of procedures for optimal relay selection.

7.3. Authorization and Encryption

A multicast sender typically has little to no control or visibility about which end hosts may receive the datastream. Encryption can be used to ensure that only authorized receivers are able to access meaningful data. That is, even if unauthorized end hosts (eg, non-paying) receive the datastream, without decryption keys, the data is useless. [I-D.ietf-ipsecme-g-ikev2] describes an extension to IKEv2 for the purpose of group key management. DVB MABR [DVB-MABR] and MAUD [MAUD] extensively describe an architecture that includes encryption of multicast streams. Multicast extensions to QUIC have been proposed in [I-D.jholland-quic-multicast].

8. TreeDN Deployments

EUMETCast Terrestrial is a service from EUMETSAT that delivers meteorological satellite data to end users for purposes such as operational monitoring of climate and the detection of global climate changes. EUMETCast Terrestrial connects to the GEANT network, which provides TreeDN services to deliver this data natively to end users on multicast-enabled networks as well as to end users on unicast-only networks via a global deployment of AMT relays. Details of the EUMETCast Terrestrial service over the GEANT TreeDN network are described in [EUMETCast-TERRESTRIAL-over-AMT]. Additional details on how this deployment uses encryption, authorization, reliability and unicast feedback channels for end-to-end file delivery monitoring can be found in [EUMETSAT-TERRESTRIAL].

The Multicast Menu is a web-based portal that lists and can launch active multicast streams that are available on a global TreeDN network of various research and educations networks. Details of the this TreeDN network, as well as the Multicast Menu, are described in [Multicast-Menu].

The RARE network is a global testbed interconnecting several national research and education networks (NRENs) via routers running BIER. AMT relays are deployed to deliver multicast traffic from sources on the RARE network to receivers on unicast-only networks across the Internet. Details of the RARE network are described in [BIER-AMT-Deployment].

9. Security Consideration

TreeDN is essentially the synthesis of SSM plus overlay networking technologies like AMT. As such, the TreeDN architecture introduces no new security threats that are not already documented in SSM and the overlay technologies that comprise it. In particular, Section 6 of [RFC7450] candidly notes that AMT, like UDP, IGMP and MLD provide no mechanisms for ensuring message delivery or integrity, nor does it provide confidentiality, since sources/groups joined through IGMP/MLD could be associated with the particular content being requested.

[RFC4609] and [RFC8815] describes the additional security benefits of using SSM instead of ASM.

10. IANA Considerations

This document has no IANA actions.

11. Acknowledgements

Many thanks to those who have contributed to building and operating the first TreeDN network on the Internet, including Pete Morasca, William Zhang, Lauren Delwiche, Natalie Landsberg, Wayne Brassem, Jake Holland, Andrew Gallo, Casey Russell, Janus Varmarken, Csaba Mate, Frederic Loui, Max Franke, Todor Moskov, Erik Herz, Bradley Cao, Katie Merrill, Karel Hendrych, Haruna Oseni and Isabelle Xiong. The writing of this document to describe the TreeDN architecture was inspired by a conversation with Dino Farinacci and Mike McBride. Thanks also to Jeff Haas and Vinod Kumar for their thoughtful reviews and suggestions, Chris Lemmons for his detailed shepherd review and Stephen Farrell, Magnus Westerlund, Reese Enghardt, and Jurgen Schonwalder for their last call reviews.

12. References

12.1. Normative References

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

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <https://www.rfc-editor.org/info/rfc4607>.

[RFC6388] Wijnands, IJ., Ed., Minei, I., Ed., Kompella, K., and B. Thomas, "Label Distribution Protocol Extensions for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", RFC 6388, DOI 10.17487/RFC6388, November 2011, <https://www.rfc-editor.org/info/rfc6388>.

[RFC7450] Bumgardner, G., "Automatic Multicast Tunneling", RFC 7450, DOI 10.17487/RFC7450, February 2015, <https://www.rfc-editor.org/info/rfc7450>.

[RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <https://www.rfc-editor.org/info/rfc7761>.

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

12.2. Informative References

[BIER-AMT-Deployment]
"BIER + AMT Deployment in GEANT/RARE Network", IETF112 Proceedings , n.d., <https://datatracker.ietf.org/meeting/112/materials/slides-112-mboned-bier-amt-depolyment-in-geantrare-network-00>.

[BROADCAST-DELAY]
"Broadcast Delay", Wikipedia , n.d., <https://en.wikipedia.org/wiki/Broadcast_delay>.

[DVB-MABR] "Adaptive media streaming over IP multicast", DVB Document A176 Rev.3 (Fourth edition) , n.d., <https://dvb.org/wp-content/uploads/2022/01/A176r3_Adaptive-Media-Streaming-over-IP-Multicast_Interim-Draft-TS-103-769-v121_March_2023.pdf>.

[EUMETCast-TERRESTRIAL-over-AMT]
"EUMETCast Terrestrial over AMT", IETF115 Proceedings , n.d., <https://datatracker.ietf.org/meeting/115/materials/slides-115-mboned-eumetcast-over-amt>.

[EUMETSAT-TERRESTRIAL]
"EUMETSAT Terrestrial Service", IETF110 Proceedings , n.d., <https://datatracker.ietf.org/meeting/110/materials/slides-110-mboned-eumetsat-multicast-over-the-mbone-00>.

[I-D.ietf-bess-bgp-multicast]
Zhang, Z. J., Giuliano, L., Patel, K., Wijnands, I., Mishra, M. P., and A. Gulko, "BGP Based Multicast", Work in Progress, Internet-Draft, draft-ietf-bess-bgp-multicast-08, 3 June 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-bess-bgp-multicast-08>.

[I-D.ietf-ipsecme-g-ikev2]
Smyslov, V. and B. Weis, "Group Key Management using IKEv2", Work in Progress, Internet-Draft, draft-ietf-ipsecme-g-ikev2-11, 26 February 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-ipsecme-g-ikev2-11>.

[I-D.ietf-spring-sr-replication-segment]
Voyer, D., Filsfils, C., Parekh, R., Bidgoli, H., and Z. J. Zhang, "SR Replication segment for Multi-point Service Delivery", Work in Progress, Internet-Draft, draft-ietf-

spring-sr-replication-segment-19, 28 August 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-spring-sr-replication-segment-19>.

[I-D.jholland-quic-multicast]
Holland, J., Pardue, L., and M. Franke, "Multicast Extension for QUIC", Work in Progress, Internet-Draft, draft-jholland-quic-multicast-05, 7 July 2024, <https://datatracker.ietf.org/doc/html/draft-jholland-quic-multicast-05>.

[MAUD] "Multicast-Assisted Unicast Delivery", IBC2023 Tech Papers , n.d., <https://www.ibc.org/technical-papers/ibc2023-tech-papers-multicast-assisted-unicast-delivery/10235.article>.

[Multicast-Menu]
"Offnet Sourcing with the Multicast Menu", IETF114 Proceedings , n.d., <https://datatracker.ietf.org/meeting/114/materials/slides-114-mboned-offnet-sourcing-with-the-multicast-menu-01>.

[RFC4609] Savola, P., Lehtonen, R., and D. Meyer, "Protocol Independent Multicast - Sparse Mode (PIM-SM) Multicast Routing Security Issues and Enhancements", RFC 4609, DOI 10.17487/RFC4609, October 2006, <https://www.rfc-editor.org/info/rfc4609>.

[RFC5740] Adamson, B., Bormann, C., Handley, M., and J. Macker, "NACK-Oriented Reliable Multicast (NORM) Transport Protocol", RFC 5740, DOI 10.17487/RFC5740, November 2009, <https://www.rfc-editor.org/info/rfc5740>.

[RFC6513] Rosen, E., Ed. and R. Aggarwal, Ed., "Multicast in MPLS/BGP IP VPNs", RFC 6513, DOI 10.17487/RFC6513, February 2012, <https://www.rfc-editor.org/info/rfc6513>.

[RFC7716] Zhang, J., Giuliano, L., Rosen, E., Ed., Subramanian, K., and D. Pacella, "Global Table Multicast with BGP Multicast VPN (BGP-MVPN) Procedures", RFC 7716, DOI 10.17487/RFC7716, December 2015, <https://www.rfc-editor.org/info/rfc7716>.

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <https://www.rfc-editor.org/info/rfc8085>.

[RFC8279] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast Using Bit Index Explicit Replication (BIER)", RFC 8279, DOI 10.17487/RFC8279, November 2017, <https://www.rfc-editor.org/info/rfc8279>.

[RFC8777] Holland, J., "DNS Reverse IP Automatic Multicast Tunneling (AMT) Discovery", RFC 8777, DOI 10.17487/RFC8777, April 2020, <https://www.rfc-editor.org/info/rfc8777>.

[RFC8815] Abrahamsson, M., Chown, T., Giuliano, L., and T. Eckert, "Deprecating Any-Source Multicast (ASM) for Interdomain Multicast", BCP 229, RFC 8815, DOI 10.17487/RFC8815, August 2020, <https://www.rfc-editor.org/info/rfc8815>.

[RFC9049] Dawkins, S., Ed., "Path Aware Networking: Obstacles to Deployment (A Bestiary of Roads Not Taken)", RFC 9049, DOI 10.17487/RFC9049, June 2021, <https://www.rfc-editor.org/info/rfc9049>.

[RFC9300] Farinacci, D., Fuller, V., Meyer, D., Lewis, D., and A. Cabellos, Ed., "The Locator/ID Separation Protocol (LISP)", RFC 9300, DOI 10.17487/RFC9300, October 2022, <https://www.rfc-editor.org/info/rfc9300>.

Authors' Addresses

Lenny Giuliano
Juniper Networks
2251 Corporate Park Drive
Herndon, VA 20171,
United States of America
Email: lenny@juniper.net

Chris Lenart
Verizon
22001 Loudoun County Parkway
Ashburn, VA 20147,
United States of America
Email: chris.lenart@verizon.com

Rich Adam
GEANT
City House
126-130 Hills Road
Cambridge
CB2 1PQ
United Kingdom
Email: richard.adam@geant.org

OPSAWG WG
Internet-Draft
Intended status: Standards Track
Expires: 23 January 2025

T. Reddy
Nokia
D. Wing
Citrix
B. Anderson
Cisco

22 July 2024

Manufacturer Usage Description (MUD) (D)TLS Profiles for IoT Devices
draft-ietf-opsawg-mud-tls-15

Abstract

This memo extends the Manufacturer Usage Description (MUD) specification to incorporate (D)TLS profile parameters. This allows a network security service to identify unexpected (D)TLS usage, which can indicate the presence of unauthorized software or malware on an endpoint.

Status of This Memo

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

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

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

This Internet-Draft will expire on 23 January 2025.

Table of Contents

1. Introduction

Encryption is necessary to enhance the privacy of end users using IoT devices. TLS [RFC8446] and DTLS [RFC9147] are the dominant protocols (counting all (D)TLS versions) providing encryption for IoT device traffic. Unfortunately, in conjunction with IoT applications' rise of encryption, malware authors are also using encryption which thwarts network-based analysis such as deep packet inspection (DPI). Other mechanisms are thus needed to help detecting malware running on an IoT device.

Malware frequently uses proprietary libraries for its activities, and those libraries are reused much like any other software engineering project. [malware] indicates that there are observable differences in how malware uses encryption compared with how non-malware uses encryption. There are several interesting findings specific to (D)TLS which were found common to malware:

* Older and weaker cryptographic parameters (e.g., TLS_RSA_WITH_RC4_128_SHA).

* TLS server name indication (SNI) extension [RFC6066] and server certificates are composed of subjects with characteristics of a domain generation algorithm (DGA) (e.g., 'www.33mhwt2j.net').

* Higher use of self-signed certificates compared with typical legitimate software.

* Discrepancies in the SNI TLS extension and the DNS names in the SubjectAltName (SAN) X.509 extension in the server certificate message.

* Discrepancies in the key exchange algorithm and the client public key length in comparison with legitimate flows. As a reminder, the Client Key Exchange message has been removed from TLS 1.3.

* Lower diversity in TLS client advertised extensions compared to legitimate clients.

* Using privacy enhancing technologies like Tor, Psiphon, Ultrasurf (see [malware-tls]), and evasion techniques such as ClientHello randomization.

* Using DNS-over-HTTPS (DoH) [RFC8484] to avoid detection by malware DNS filtering services [malware-doh]. Specifically, malware may not use the DoH server provided by the local network.

If observable (D)TLS profile parameters are used, the following functions are possible which have a positive impact on the local network security:

* Permit intended DTLS or TLS use and block malicious DTLS or TLS use. This is superior to the layers 3 and 4 ACLs of Manufacturer Usage Description Specification (MUD) [RFC8520] which are not suitable for broad communication patterns.

* Ensure TLS certificates are valid. Several TLS deployments have been vulnerable to active Man-In-The-Middle (MITM) attacks because of the lack of certificate validation or vulnerability in the certificate validation function (see [cryto-vulnerability]). By observing (D)TLS profile parameters, a network element can detect when the TLS SNI mismatches the SubjectAltName and when the server's certificate is invalid. In (D)TLS 1.2 [RFC5246][RFC6347], the ClientHello, ServerHello and Certificate messages are all sent in clear-text. This check is not possible with (D)TLS 1.3, which encrypts the Certificate message thereby hiding the server identity from any intermediary. In (D)TLS 1.3, the server certificate validation functions should be executed within an on-path (D)TLS proxy, if such a proxy exists.

* Support new communication patterns. An IoT device can learn a new capability, and the new capability can change the way the IoT device communicates with other devices located in the local network and Internet. There would be an inaccurate policy if an IoT device rapidly changes the IP addresses and domain names it communicates with while the MUD ACLs were slower to update (see [clear-as-mud]). In such a case, observable (D)TLS profile parameters can be used to permit intended use and to block malicious behavior from the IoT device.

The YANG module specified in Section 5.2 of this document is an extension of YANG Data Model for Network Access Control Lists (ACLs) [RFC8519] to enhance MUD [RFC8520] to model observable (D)TLS profile parameters. Using these (D)TLS profile parameters, an active MUD-enforcing network security service (e.g., firewall) can identify MUD non-compliant (D)TLS behavior indicating outdated cryptography or malware. This detection can prevent malware downloads, block access to malicious domains, enforce use of strong ciphers, stop data exfiltration, etc. In addition, organizations may have policies around acceptable ciphers and certificates for the websites the IoT devices connect to. Examples include no use of old and less secure versions of TLS, no use of self-signed certificates, deny-list or accept-list of Certificate Authorities, valid certificate expiration time, etc. These policies can be enforced by observing the (D)TLS profile parameters. Network security services can use the IoT

device's (D)TLS profile parameters to identify legitimate flows by observing (D)TLS sessions, and can make inferences to permit legitimate flows and to block malicious or insecure flows. The proposed technique is also suitable in deployments where decryption techniques are not ideal due to privacy concerns, non-cooperating end-points, and expense.

2. Terminology

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

"(D)TLS" is used for statements that apply to both Transport Layer Security [RFC8446] and Datagram Transport Layer Security [RFC6347]. Specific terms "TLS" and "DTLS" are used for any statement that applies to either protocol alone.

'DoH/DoT' refers to DNS-over-HTTPS and/or DNS-over-TLS.

3. Overview of MUD (D)TLS profiles for IoT devices

In Enterprise networks, protection and detection are typically done both on end hosts and in the network. Host security agents have deep visibility on the devices where they are installed, whereas the network has broader visibility. Installing host security agents may not be a viable option on IoT devices, and network-based security is an efficient means to protect such IoT devices. If the IoT device supports a MUD (D)TLS profile, the (D)TLS profile parameters of the IoT device can be used by a middlebox to detect and block malware communication, while at the same time preserving the privacy of legitimate uses of encryption. The middlebox need not proxy (D)TLS but can passively observe the parameters of (D)TLS handshakes from IoT devices and gain visibility into TLS 1.2 parameters and partial visibility into TLS 1.3 parameters.

Malicious agents can try to use the (D)TLS profile parameters of legitimate agents to evade detection, but it becomes a challenge to mimic the behavior of various IoT device types and IoT device models from several manufacturers. In other words, malware developers will have to develop malicious agents per IoT device type, manufacturer and model, infect the device with the tailored malware agent and will have keep up with updates to the device's (D)TLS profile parameters over time. Furthermore, the malware's command and control server certificates need to be signed by the same certifying authorities trusted by the IoT devices. Typically, IoT devices have an

infrastructure that supports a rapid deployment of updates, and malware agents will have a near-impossible task of similarly deploying updates and continuing to mimic the TLS behavior of the IoT device it has infected. However, if the IoT device has reached end-of-life and the IoT manufacturer will not issue a firmware or software update to the Thing or will not update the MUD file, the "is-supported" attribute defined in Section 3.6 of [RFC8520] can be used by the MUD manager to identify the IoT manufacturer no longer supports the device.

The end-of-life of a device does not necessarily mean that it is defective; rather, it denotes a need to replace and upgrade the network to next-generation devices for additional functionality. The network security service will have to rely on other techniques discussed in Section 8 to identify malicious connections until the device is replaced.

Compromised IoT devices are typically used for launching DDoS attacks (Section 3 of [RFC8576]). For example, DDoS attacks like Slowloris and Transport Layer Security (TLS) re-negotiation can be blocked if the victim's server certificate is not be signed by the same certifying authorities trusted by the IoT device.

4. (D)TLS 1.3 Handshake

In (D)TLS 1.3, full (D)TLS handshake inspection is not possible since all (D)TLS handshake messages excluding the ClientHello message are encrypted. (D)TLS 1.3 has introduced new extensions in the handshake record layers called Encrypted Extensions. Using these extensions handshake messages will be encrypted and network security services (such as a firewall) are incapable of deciphering the handshake, and thus cannot view the server certificate. However, the ClientHello and ServerHello still have some fields visible, such as the list of supported versions, named groups, cipher suites, signature algorithms and extensions in ClientHello, and chosen cipher in the ServerHello. For instance, if the malware uses evasion techniques like ClientHello randomization, the observable list of cipher suites and extensions offered by the malware agent in the ClientHello message will not match the list of cipher suites and extensions offered by the legitimate client in the ClientHello message, and the middlebox can block malicious flows without acting as a (D)TLS 1.3 proxy.

4.1. Full (D)TLS 1.3 Handshake Inspection

To obtain more visibility into negotiated TLS 1.3 parameters, a middlebox can act as a (D)TLS 1.3 proxy. A middlebox can act as a (D)TLS proxy for the IoT devices owned and managed by the IT team in the Enterprise network and the (D)TLS proxy must meet the security and privacy requirements of the organization. In other words, the scope of middlebox acting as a (D)TLS proxy is restricted to Enterprise network owning and managing the IoT devices. The middlebox would have to follow the behaviour detailed in Section 9.3 of [RFC8446] to act as a compliant (D)TLS 1.3 proxy.

To further increase privacy, Encrypted Client Hello (ECH) extension [I-D.ietf-tls-esni] prevents passive observation of the TLS Server Name Indication extension and other potentially sensitive fields, such as the ALPN [RFC7301]. To effectively provide that privacy protection, ECH extension needs to be used in conjunction with DNS encryption (e.g., DoH). A middlebox (e.g., firewall) passively inspecting ECH extension cannot observe the encrypted SNI nor observe the encrypted DNS traffic. The middlebox acting as a (D)TLS 1.3 proxy that does not support ECH extension will act as if connecting to the public name and it follows the behaviour discussed in Section 6.1.6 of [I-D.ietf-tls-esni] to securely signal the client to disable ECH.

4.2. Encrypted DNS

A common usage pattern for certain type of IoT devices (e.g., light bulb) is for it to "call home" to a service that resides on the public Internet, where that service is referenced through a domain name (A or AAAA record). As discussed in Manufacturer Usage Description Specification [RFC8520], because these devices tend to require access to very few sites, all other access should be considered suspect. If an IoT device is pre-configured to use a DNS resolver not signaled by the network, the MUD policy enforcement point is moved to that resolver, which cannot enforce the MUD policy based on domain names (Section 8 of [RFC8520]). If the DNS query is not accessible for inspection, it becomes quite difficult for the infrastructure to suspect anything. Thus the use of a DNS resolver not signaled by the network is incompatible with MUD in general. A network-designated DoH/DoT server is necessary to allow MUD policy enforcement on the local network, for example, using the techniques specified in DNR[RFC9463] and DDR [RFC9462].

5. (D)TLS Profile of a IoT device

This document specifies a YANG module for representing (D)TLS profile. The (D)TLS profile YANG module provides a method for network security services to observe the (D)TLS profile parameters in the (D)TLS handshake to permit intended use and to block malicious behavior. This module uses the cryptographic types defined in [I-D.ietf-netconf-crypto-types]. See [RFC7925] for (D)TLS 1.2 and [I-D.ietf-uta-tls13-iot-profile] for DTLS 1.3 recommendations related to IoT devices, and [RFC7525] for additional (D)TLS 1.2 recommendations.

A companion YANG module is defined to include a collection of (D)TLS parameters and (D)TLS versions maintained by IANA: "iana-tls-profile" (Section 5.3).

The (D)TLS parameters in each (D)TLS profile include the following:

* Profile name
* (D)TLS versions supported by the IoT device.
* List of supported cipher suites (Section 11 of [RFC8446]). For (D)TLS1.2, [RFC7925] recommends AEAD ciphers for IoT devices.
* List of supported extension types
* List of trust anchor certificates used by the IoT device. If the server certificate is signed by one of the trust anchors, the middlebox continues with the connection as normal. Otherwise, the middlebox will react as if the server certificate validation has failed and takes appropriate action (e.g, block the (D)TLS session). An IoT device can use a private trust anchor to validate a server's certificate (e.g., the private trust anchor can be preloaded at manufacturing time on the IoT device and the IoT device fetches the firmware image from the Firmware server whose certificate is signed by the private CA). This empowers the middlebox to reject TLS sessions to servers that the IoT device does not trust.
* List of pre-shared key exchange modes
* List of named groups (DHE or ECDHE) supported by the client
* List of signature algorithms the client can validate in X.509 server certificates

* List of signature algorithms the client is willing to accept for CertificateVerify message (Section 4.2.3 of [RFC8446]). For example, a TLS client implementation can support different sets of algorithms for certificates and in TLS to signal the capabilities in "signature_algorithms_cert" and "signature_algorithms" extensions.

* List of supported application protocols (e.g., h3, h2, http/1.1 etc.)

* List of certificate compression algorithms (defined in [RFC8879])

* List of the distinguished names [X501] of acceptable certificate authorities, represented in DER-encoded format [X690] (defined in Section 4.2.4 of [RFC8446])

GREASE [RFC8701] defines a mechanism for TLS peers to send random values on TLS parameters to ensure future extensibility of TLS extensions. Similar random values might be extended to other TLS parameters. Thus, the (D)TLS profile parameters defined in the YANG module by this document MUST NOT include the GREASE values for extension types, named groups, signature algorithms, (D)TLS versions, pre-shared key exchange modes, cipher suites and for any other TLS parameters defined in future RFCs.

The (D)TLS profile does not include parameters like compression methods for data compression, [RFC7525] recommends disabling TLS-level compression to prevent compression-related attacks. In TLS 1.3, only the "null" compression method is allowed (Section 4.1.2 of [RFC8446]).

5.1. Tree Structure of the (D)TLS profile Extension to the ACL YANG Model

This document augments the "ietf-acl" ACL YANG module defined in [RFC8519] for signaling the IoT device (D)TLS profile. This document defines the YANG module "ietf-acl-tls". The meaning of the symbols in the YANG tree diagram are defined in [RFC8340] and it has the following tree structure:

```
module: ietf-acl-tls
  augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches:
    +--rw client-profiles {match-on-tls-dtls}?
       +--rw tls-dtls-profile* [name]
          +--rw name                              string
          +--rw supported-tls-version*        ianatp:tls-version
          +--rw supported-dtls-version*       ianatp:dtls-version
          +--rw cipher-suite*                 ianatp:cipher-algorithm
          +--rw extension-type*
          |       ianatp:extension-type
          +--rw accept-list-ta-cert*
          |       ct:trust-anchor-cert-cms
          +--rw psk-key-exchange-mode*
          |       ianatp:psk-key-exchange-mode
          |       {tls13 or dtls13}?
          +--rw supported-groups*
          |       ianatp:supported-group
          +--rw signature-algorithm-cert*
          |       ianatp:signature-algorithm
          |       {tls13 or dtls13}?
          +--rw signature-algorithm*
          |       ianatp:signature-algorithm
          +--rw application-protocol*
          |       ianatp:application-protocol
          +--rw cert-compression-algorithm*
          |       ianatp:cert-compression-algorithm
          |       {tls13 or dtls13}?
          +--rw certificate-authorities*
                  certificate-authority
                  {tls13 or dtls13}?
```

5.2. The (D)TLS profile Extension to the ACL YANG Model

```
<CODE BEGINS> file "ietf-acl-tls@2024-01-23.yang"
module ietf-acl-tls {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-acl-tls";
  prefix acl-tls;

  import iana-tls-profile {
    prefix ianatp;
    reference
      "RFC XXXX: Manufacturer Usage Description (MUD) (D)TLS
                 Profiles for IoT Devices";
  }
  import ietf-crypto-types {
    prefix ct;
    reference
```

```
        "draft-ietf-netconf-crypto-types: YANG Data Types and Groupings
              for Cryptography";
    }
    import ietf-access-control-list {
      prefix acl;
      reference
        "RFC 8519: YANG Data Model for Network Access
                   Control Lists (ACLs)";
    }

    organization
      "IETF OPSAWG (Operations and Management Area Working Group)";
    contact
      "WG Web: <https://datatracker.ietf.org/wg/opsawg/>
       WG List: opsawg@ietf.org

        Author: Konda, Tirumaleswar Reddy
                     kondtir@gmail.com
      ";
    description
      "This YANG module defines a component that augments the
        IETF description of an access list to allow (D)TLS profile
        as matching criteria.

       Copyright (c) 2024 IETF Trust and the persons identified as
       authors of the code.  All rights reserved.

       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject
       to the license terms contained in, the Revised BSD License
       set forth in Section 4.c of the IETF Trust's Legal Provisions
       Relating to IETF Documents
       (http://trustee.ietf.org/license-info).

       This version of this YANG module is part of RFC XXXX; see
       the RFC itself for full legal notices.";

    revision 2022-10-10 {
      description
        "Initial revision";
      reference
        "RFC XXXX: Manufacturer Usage Description (MUD) (D)TLS
                   Profiles for IoT Devices";
    }

    feature tls12 {
      description
        "TLS Protocol Version 1.2 is supported.";
```

```
      reference
        "RFC 5246: The Transport Layer Security (TLS) Protocol
                   Version 1.2";
    }

    feature tls13 {
      description
        "TLS Protocol Version 1.3 is supported.";
      reference
        "RFC 8446: The Transport Layer Security (TLS) Protocol
                   Version 1.3";
    }

    feature dtls12 {
      description
        "DTLS Protocol Version 1.2 is supported.";
      reference
        "RFC 6347: Datagram Transport Layer Security
                   Version 1.2";
    }

    feature dtls13 {
      description
        "DTLS Protocol Version 1.3 is supported.";
      reference
        "RFC 9147: Datagram Transport Layer
                  Security 1.3";
    }

    feature match-on-tls-dtls {
      description
        "The networking device can support matching on
         (D)TLS parameters.";
    }

    typedef spki-pin-set {
      type binary;
      description
        "Subject Public Key Info pin set as discussed in
         Section 2.4 of RFC7469.";
    }

    typedef certificate-authority {
      type string;
      description
        "Distinguished Name of Certificate authority as discussed
         in Section 4.2.4 of RFC8446.";
    }
```

```
augment "/acl:acls/acl:acl/acl:aces/acl:ace/acl:matches" {
  if-feature "match-on-tls-dtls";
  description
    "(D)TLS specific matches.";
  container client-profiles {
    description
      "A grouping for (D)TLS profiles.";
    list tls-dtls-profile {
      key "name";
      description
        "A list of (D)TLS version profiles supported by
         the client.";
      leaf name {
        type string {
        length "1..64";
      }
      description
        "The name of (D)TLS profile; space and special
         characters are not allowed.";
      }
     leaf-list supported-tls-version {
       type ianatp:tls-version;
       description
       "TLS versions supported by the client.";
     }
     leaf-list supported-dtls-version {
        type ianatp:dtls-version;
        description
          "DTLS versions supported by the client.";
     }
     leaf-list cipher-suite {
       type ianatp:cipher-algorithm;
       description
         "A list of Cipher Suites supported by the client.";
     }
     leaf-list extension-type {
        type ianatp:extension-type;
        description
          "A list of Extension Types supported by the client.";
     }
     leaf-list accept-list-ta-cert {
        type ct:trust-anchor-cert-cms;
        description
          "A list of trust anchor certificates used by the client.";
     }
     leaf-list psk-key-exchange-mode {
       if-feature "tls13 or dtls13";
       type ianatp:psk-key-exchange-mode;
```

```
              description
              "pre-shared key exchange modes.";
            }
          leaf-list supported-group {
            type ianatp:supported-group;
            description
              "A list of named groups supported by the client.";
          }
          leaf-list signature-algorithm-cert {
            if-feature "tls13 or dtls13";
            type ianatp:signature-algorithm;
            description
              "A list signature algorithms the client can validate
              in X.509 certificates.";
          }
          leaf-list signature-algorithm {
            type ianatp:signature-algorithm;
            description
              "A list signature algorithms the client can validate
               in the CertificateVerify message.";
          }
          leaf-list application-protocol {
            type ianatp:application-protocol;
            description
              "A list application protocols supported by the client.";
          }
          leaf-list cert-compression-algorithm {
            if-feature "tls13 or dtls13";
            type ianatp:cert-compression-algorithm;
            description
              "A list certificate compression algorithms
              supported by the client.";
          }
          leaf-list certificate-authorities {
            if-feature "tls13 or dtls13";
            type certificate-authority;
            description
              "A list of the distinguished names of certificate authorities
               acceptable to the client.";
          }
        }
      }
    }
  }
  <CODE ENDS>
```

5.3. IANA (D)TLS profile YANG Module

The TLS and DTLS IANA registries are available from https://www.iana.org/assignments/tls-parameters/tls-parameters.txt and https://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.txt. Changes to TLS and DTLS related IANA registries are discussed in [RFC8447].

The values for all the parameters in the "iana-tls-profile" YANG module are defined in the TLS and DTLS IANA registries excluding the tls-version, dtls-version, spki-pin-set, and certificate-authority parameters. The values of spki-pin-set and certificate-authority parameters will be specific to the IoT device.

The TLS and DTLS IANA registries do not maintain (D)TLS version numbers. In (D)TLS 1.2 and below, "legacy_version" field in the ClientHello message is used for version negotiation. However in (D)TLS 1.3, the "supported_versions" extension is used by the client to indicate which versions of (D)TLS it supports. TLS 1.3 ClientHello messages are identified as having a "legacy_version" of 0x0303 and a "supported_versions" extension present with 0x0304 as the highest version. DTLS 1.3 ClientHello messages are identified as having a "legacy_version" of 0xfefd and a "supported_versions" extension present with 0x0304 as the highest version.

In order to ease updating the "iana-tls-profile" YANG module with future (D)TLS versions, new (D)TLS version registries are defined in Section 10.3 and Section 10.4. Whenever a new (D)TLS protocol version is defined, the registry will be updated using expert review; the "iana-tls-profile" YANG module will be automatically updated by IANA.

Implementers or users of this specification must refer to the IANA-maintained "iana-tls-profile" YANG module available at XXXX [Note to RFC Editor to replace "XXXX" with the URL link of the IANA-maintained "iana-tls-profile" YANG module].

The initial version of the "iana-tls-profile" YANG module is defined as follows:

```
<CODE BEGINS> file "iana-tls-profile@2024-01-23.yang"
module iana-tls-profile {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-tls-profile";
  prefix ianatp;

  organization
    "IANA";
```

```
    contact
      "        Internet Assigned Numbers Authority

       Postal: ICANN
               12025 Waterfront Drive, Suite 300
               Los Angeles, CA  90094-2536
               United States

       Tel:    +1 310 301 5800
       E-Mail: iana@iana.org>";
    description
      "This module contains YANG definition for the (D)TLS profile.

       Copyright (c) 2024 IETF Trust and the persons identified as
       authors of the code.  All rights reserved.

       Redistribution and use in source and binary forms, with or
       without modification, is permitted pursuant to, and subject
       to the license terms contained in, the Revised BSD License
       set forth in Section 4.c of the IETF Trust's Legal Provisions
       Relating to IETF Documents
       (http://trustee.ietf.org/license-info).

       All revisions of IETF and IANA published modules can be found
       at the YANG Parameters registry
       (https://www.iana.org/assignments/yang-parameters).

       The initial version of this YANG module is part of RFC XXXX;
       see the RFC itself for full legal notices.

       // RFC Ed.: replace the IANA_TLS-PROFILE_URL and remove this note
       The latest version of this YANG module is available at
       <IANA_TLS-PROFILE_URL>.";

    revision 2022-10-10 {
      description
        "Initial revision";
      reference
        "RFC XXXX: Manufacturer Usage Description (MUD) (D)TLS Profiles
                   for IoT Devices";
    }

    typedef extension-type {
      type uint16;
      description
        "Extension type in the TLS ExtensionType Values registry as
         defined in Section 7 of RFC8447.";
    }
```

```
typedef supported-group {
  type uint16;
  description
    "Supported Group in the TLS Supported Groups registry as
     defined in Section 9 of RFC8447.";
}

typedef signature-algorithm {
  type uint16;
  description
    "Signature algorithm in the TLS SignatureScheme registry as
     defined in Section 11 of RFC8446.";
}

typedef psk-key-exchange-mode {
  type uint8;
  description
    "Pre-shared key exchange mode in the TLS PskKeyExchangeMode
     registry as defined in Section 11 of RFC8446.";
}

typedef application-protocol {
  type string;
  description
    "Application-Layer Protocol Negotiation (ALPN) Protocol ID
     registry as defined in Section 6 of RFC7301.";
}

typedef cert-compression-algorithm {
  type uint16;
  description
    "Certificate compression algorithm in TLS Certificate
     Compression Algorithm IDs registry as defined in
     Section 7.3 of RFC8879.";
}

typedef cipher-algorithm {
  type uint16;
  description
    "Cipher suite in TLS Cipher Suites registry
     as discussed in Section 11 of RFC8446.";
}

typedef tls-version {
  type enumeration {
    enum tls12 {
      value 1;
      description
```

```
            "TLS Protocol Version 1.2.

             TLS 1.2 ClientHello contains
             0x0303 in 'legacy_version'.";
          reference
            "RFC 5246: The Transport Layer Security (TLS) Protocol
                       Version 1.2";
        }
        enum tls13 {
          value 2;
          description
            "TLS Protocol Version 1.3.

             TLS 1.3 ClientHello contains a
             supported_versions extension with 0x0304
             contained in its body and the ClientHello contains
             0x0303 in 'legacy_version'.";
          reference
            "RFC 8446: The Transport Layer Security (TLS) Protocol
                       Version 1.3";
        }
      }
      description
        "Indicates the TLS version.";
    }

    typedef dtls-version {
      type enumeration {
        enum dtls12 {
          value 1;
          description
            "DTLS Protocol Version 1.2.

             DTLS 1.2 ClientHello contains
             0xfefd in 'legacy_version'.";
          reference
            "RFC 6347: Datagram Transport Layer Security 1.2";
        }
        enum dtls13 {
          value 2;
          description
            "DTLS Protocol Version 1.3.

             DTLS 1.3 ClientHello contains a
             supported_versions extension with 0x0304
             contained in its body and the ClientHello contains
             0xfefd in 'legacy_version'.";
          reference
```

```
              "RFC 9147: Datagram Transport Layer Security 1.3";
          }
        }
        description
          "Indicates the DTLS version.";
      }
    }
    <CODE ENDS>
```

5.4. MUD (D)TLS Profile Extension

This document augments the "ietf-mud" MUD YANG module to indicate whether the device supports (D)TLS profile. If the "ietf-mud-tls" extension is supported by the device, MUD file is assumed to implement the "match-on-tls-dtls" ACL model feature defined in this specification. Furthermore, only "accept" or "drop" actions SHOULD be included with the (D)TLS profile similar to the actions allowed in Section 2 of [RFC8520].

This document defines the YANG module "ietf-mud-tls", which has the following tree structure:

```
module: ietf-mud-tls
  augment /ietf-mud:mud:
    +--rw is-tls-dtls-profile-supported?   boolean
```

The model is defined as follows:

```
<CODE BEGINS> file "ietf-mud-tls@2020-10-20.yang"
module ietf-mud-tls {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-mud-tls";
  prefix ietf-mud-tls;

  import ietf-mud {
    prefix ietf-mud;
    reference
      "RFC 8520: Manufacturer Usage Description Specification";
  }

  organization
    "IETF OPSAWG (Operations and Management Area Working Group)";
  contact
    "WG Web: <https://datatracker.ietf.org/wg/opsawg/>
     WG List: opsawg@ietf.org

     Author: Konda, Tirumaleswar Reddy
             kondtir@gmail.com
```

```
       ";
      description
        "Extension to a MUD module to indicate (D)TLS
         profile support.

        Copyright (c) 2024 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Revised BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see
        the RFC itself for full legal notices.";

      revision 2022-10-10 {
        description
          "Initial revision.";
          reference
            "RFC XXXX: Manufacturer Usage Description (MUD) (D)TLS
             Profiles for IoT Devices";
      }

      augment "/ietf-mud:mud" {
        description
          "This adds a extension for a manufacturer
           to indicate whether (D)TLS profile is
           is supported by a device.";
        leaf is-tls-dtls-profile-supported {
          type boolean;
          default false;
            description
              "This value will equal 'true' if a device supports
               (D)TLS profile.";
        }
      }
   }
   <CODE ENDS>
```

6. Processing of the MUD (D)TLS Profile

The following text outlines the rules for a network security service (e.g., firewall) to follow to process the MUD (D)TLS Profile so as to avoid ossification:

* If the (D)TLS parameter observed in a (D)TLS session is not specified in the MUD (D)TLS profile and the parameter is recognized by the firewall, it can identify unexpected (D)TLS usage, which can indicate the presence of unauthorized software or malware on an endpoint. The firewall can take several actions like block the (D)TLS session or raise an alert to quarantine and remediate the compromised device. For example, if the cipher suite TLS_RSA_WITH_AES_128_CBC_SHA in the ClientHello message is not specified in the MUD (D)TLS profile and the cipher suite is recognized by the firewall, it can identify unexpected TLS usage.

* If the (D)TLS parameter observed in a (D)TLS session is not specified in the MUD (D)TLS profile and the (D)TLS parameter is not recognized by the firewall, it can ignore the unrecognized parameter and the correct behavior is not to block the (D)TLS session. The behaviour is functionally equivalent to the compliant TLS middlebox description in Section 9.3 of [RFC8446] to ignore all unrecognized cipher suites, extensions, and other parameters. For example, if the cipher suite TLS_CHACHA20_POLY1305_SHA256 in the ClientHello message is not specified in the MUD (D)TLS profile and the cipher suite is not recognized by the firewall, it can ignore the unrecognized cipher suite. This rule also ensures that the network security service will ignore the GREASE values advertised by TLS peers and interoperate with the implementations advertising GREASE values.

* Deployments update at different rates, so an updated MUD (D)TLS profile may support newer parameters. If the firewall does not recognize the newer parameters, an alert should be triggered to the firewall vendor and the IoT device owner or administrator. A firewall must be readily updatable, so that when new parameters in the MUD (D)TLS profile are discovered that are not recognized by the firewall, it can be updated quickly. Most importantly, if the firewall is not readily updatable, its protection efficacy to identify emerging malware will decrease with time. For example, if the cipher suite TLS_AES_128_CCM_8_SHA256 specified in the MUD (D)TLS profile is not recognized by the firewall, an alert will be triggered. Similarly, if the (D)TLS version specified in the MUD file is not recognized by the firewall, an alert will be triggered.

* If the MUD (D)TLS profile includes any parameters that are susceptible to attacks (e.g., weaker cryptographic parameters), an alert should be triggered to the firewall vendor and the IoT device owner or administrator.

7. MUD File Example

The example below contains (D)TLS profile parameters for a IoT device used to reach servers listening on port 443 using TCP transport. JSON encoding of YANG modelled data [RFC7951] is used to illustrate the example.

```
{
   "ietf-mud:mud": {
     "mud-version": 1,
      "mud-url": "https://example.com/IoTDevice",
      "last-update": "2019-18-06T03:56:40.105+10:00",
      "cache-validity": 100,
      "extensions": [
           "ietf-mud-tls"
       ],
      "ietf-mud-tls:is-tls-dtls-profile-supported": "true",
      "is-supported": true,
      "systeminfo": "IoT device name",
      "from-device-policy": {
         "access-lists": {
           "access-list": [
             {
               "name": "mud-7500-profile"
             }
           ]
         }
      },
     "ietf-access-control-list:acls": {
       "acl": [
         {
           "name": "mud-7500-profile",
           "type": "ipv6-acl-type",
           "aces": {
             "ace": [
               {
                 "name": "cl0-frdev",
                 "matches": {
                   "ipv6": {
                     "protocol": 6
                   },
                   "tcp": {
                     "ietf-mud:direction-initiated": "from-device",
                     "destination-port": {
                       "operator": "eq",
                       "port": 443
                     }
                   },
```

```
                    "ietf-acl-tls:client-profile" : {
                      "tls-dtls-profiles" : [
                         {
                            "name" : "profile1",
                            "supported-tls-versions" : ["tls13"],
                            "cipher-suite" : [4865, 4866],
                            "extension-types" : [10,11,13,16,24],
                            "supported-groups" : [29]
                         }
                       ]
                    },
                    "actions": {
                       "forwarding": "accept"
                    }
                 }
              }
           ]
         }
       ]
     }
   }
```

The following illustrates the example scenarios for processing the above profile:

* If the extension type "encrypt_then_mac" (code point 22) [RFC7366] in the ClientHello message is recognized by the firewall, it can identify unexpected TLS usage.

* If the extension type "token_binding" (code point 24) [RFC8472] in the MUD (D)TLS profile is not recognized by the firewall, it can ignore the unrecognized extension. Because the extension type "token_binding" is specified in the profile, an alert will be triggered to the firewall vendor and the IoT device owner or administrator to notify the firewall is not up to date.

* The two-byte values assigned by IANA for the cipher suites TLS_AES_128_GCM_SHA256 and TLS_AES_256_GCM_SHA384 are represented in decimal format.

8. Security Considerations

Security considerations in [RFC8520] need to be taken into consideration. The middlebox must adhere to the invariants discussed in Section 9.3 of [RFC8446] to act as a compliant proxy.

Although it is challenging for a malware to mimic the TLS behavior of various IoT device types and IoT device models from several manufacturers, malicious agents have a very low probability of using the same (D)TLS profile parameters as legitimate agents on the IoT device to evade detection. Network security services should also rely on contextual network data to detect false negatives. In order to detect such malicious flows, anomaly detection (deep learning techniques on network data) can be used to detect malicious agents using the same (D)TLS profile parameters as legitimate agent on the IoT device. In anomaly detection, the main idea is to maintain rigorous learning of "normal" behavior and where an "anomaly" (or an attack) is identified and categorized based on the knowledge about the normal behavior and a deviation from this normal behavior.

8.1. Considerations for the "iana-tls-profile" Module

This section follows the template defined in Section 3.7.1 of [RFC8407].

The YANG module specified in this document defines a schema for data can possibly be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. These network management protocols are required to use a secure transport layer and mutual authentication, e.g., SSH [RFC6242] without the "none" authentication option, Transport Layer Security (TLS) [RFC8446] with mutual X.509 authentication, and HTTPS with HTTP authentication (Section 11 of [RFC9110]).

The Network Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

This YANG module defines YANG enumerations, for a public IANA-maintained registry.

YANG enumerations are not security-sensitive, as they are statically defined in the publicly-accessible YANG module. IANA MAY deprecate and/or obsolete enumerations over time as needed to address security issues.

This module does not define any writable-nodes, RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

8.2. Considerations for the "ietf-acl-tls" Module

This section follows the template defined in Section 3.7.1 of [RFC8407].

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. These network management protocols are required to use a secure transport layer and mutual authentication, e.g., SSH [RFC6242] without the "none" authentication option, Transport Layer Security (TLS) [RFC8446] with mutual X.509 authentication, and HTTPS with HTTP authentication (Section 11 of [RFC9110]).

The Network Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Please be aware that this YANG module uses groupings from other YANG modules that define nodes that may be considered sensitive or vulnerable in network environments. Please review the Security Considerations for dependent YANG modules for information as to which nodes may be considered sensitive or vulnerable in network environments.

All the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, the addition or removal of references to trusted anchors, (D)TLS versions, cipher suites etc., can dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

8.3. Considerations for the "ietf-mud-tls" Module

This section follows the template defined in Section 3.7.1 of [RFC8407].

The YANG module specified in this document defines a schema for data can possibly be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. These network management protocols are required to use a secure transport layer and mutual authentication, e.g., SSH [RFC6242] without the "none" authentication option, Transport Layer Security (TLS) [RFC8446] with mutual X.509 authentication, and HTTPS with HTTP authentication (Section 11 of [RFC9110]). Note that the YANG module is not intended to be accessed via NETCONF and RESTCONF. This has already been discussed in [RFC8520], and we are reiterating it here for the sake of completeness.

The Network Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular users to a pre-configured subset of all available protocol operations and content.

Please be aware that this YANG module uses groupings from other YANG modules that define nodes that may be considered sensitive or vulnerable in network environments. Please review the Security Considerations for dependent YANG modules for information as to which nodes may be considered sensitive or vulnerable in network environments.

All the writable data nodes defined by this module may be considered sensitive or vulnerable in some network environments. For instance, update that the device does not support (D)TLS profile can dramatically alter the implemented security policy. For this reason, the NACM extension "default-deny-write" has been set for all data nodes defined in this module.

This module does not define any RPCs, actions, or notifications, and thus the security consideration for such is not provided here.

9. Privacy Considerations

Privacy considerations discussed in Section 16 of [RFC8520] to not reveal the MUD URL to an attacker need to be taken into consideration. The MUD URL can be stored in Trusted Execution Environment (TEE) for secure operation, enhanced data security, and prevent exposure to unauthorized software. The MUD URL MUST be encrypted and shared only with the authorized components in the network (see Section 1.5 and Section 1.8 of [RFC8520]) so that an on-path attacker cannot read the MUD URL and identify the IoT device. Otherwise, it provides the attacker with guidance on what vulnerabilities may be present on the IoT device. Note that while protecting the MUD URL is valuable as described above, a compromised IoT device may be susceptible to malware performing vulnerability analysis (and version mapping) of the legitimate software located in memory or on non-volatile storage (e.g., disk, NVRAM). However, the malware on the IoT device won't be able to establish a (D)TLS connection with the C&C server to reveal this information because the connection would be blocked by the network security service supporting this specification.

Full handshake inspection (Section 4.1) requires a (D)TLS proxy device which needs to decrypt traffic between the IoT device and its server(s). There is a tradeoff between privacy of the data carried inside (D)TLS (especially e.g., personally identifiable information and protected health information) and efficacy of endpoint security. It is strongly RECOMMENDED to avoid a (D)TLS proxy whenever possible.

For example, an enterprise firewall administrator can configure the middlebox to bypass (D)TLS proxy functionality or payload inspection for connections destined to specific well-known services. Alternatively, a IoT device could be configured to reject all sessions that involve proxy servers to specific well-known services. In addition, mechanisms based on object security can be used by IoT devices to enable end-to-end security and the middlebox will not have any access to the packet data. For example, Object Security for Constrained RESTful Environments (OSCORE) [RFC8613] is a proposal that protects CoAP messages by wrapping them in the COSE format [RFC8152].

10. IANA Considerations

10.1. (D)TLS Profile YANG Modules

This document requests IANA to register the following URIs in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

```
URI: urn:ietf:params:xml:ns:yang:iana-tls-profile
Registrant Contact: IANA.
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-acl-tls
Registrant Contact: IESG.
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-mud-tls
Registrant Contact: IESG.
XML: N/A; the requested URI is an XML namespace.
```

IANA is requested to create an IANA-maintained YANG Module called "iana-tls-profile", based on the contents of Section 5.3, which will allow for new (D)TLS parameters and (D)TLS versions to be added to "client-profile".

This document requests IANA to register the following YANG modules in the "YANG Module Names" subregistry [RFC6020] within the "YANG Parameters" registry.

```
name: iana-tls-profile
namespace: urn:ietf:params:xml:ns:yang:iana-tls-profile
maintained by IANA: Y
prefix: ianatp
reference: RFC XXXX
```

```
name: ietf-acl-tls
namespace: urn:ietf:params:xml:ns:yang:ietf-acl-tls
maintained by IANA: N
prefix: ietf-acl-tls
reference: RFC XXXX

name: ietf-mud-tls
namespace: urn:ietf:params:xml:ns:yang:ietf-mud-tls
maintained by IANA: N
prefix: ietf-mud-tls
reference: RFC XXXX
```

10.2. Considerations for the iana-tls-profile Module

IANA is requested to create an the initial version of the IANA-maintained YANG Module called "iana-tls-profile", based on the contents of Section 5.3, which will allow for new (D)TLS parameters and (D)TLS versions to be added. IANA is requested to add this note:

* tls-version and dtls-version values must not be directly added to the iana-tls-profile YANG module. They must instead be respectively added to the "ACL TLS Version Codes", and "ACL DTLS Version Codes" registries provided the new (D)TLS version specification is adopted by the TLS WG. It allows new (D)TLS version to be added to the "iana-tls-profile" YANG Module.

* (D)TLS parameters must not be directly added to the iana-tls-profile YANG module. They must instead be added to the "ACL (D)TLS Parameters" registry if the new (D)TLS parameters can be used by a middlebox to identify a MUD non-compliant (D)TLS behavior. It allows new (D)TLS parameters to be added to the "iana-tls-profile" YANG Module,

When a 'tls-version' or 'dtls-version' value is respectively added to the "ACL TLS Version Codes" or "ACL DTLS Version Codes" registry, a new "enum" statement must be added to the iana-tls-profile YANG module. The following "enum" statement, and substatements thereof, should be defined:

"enum": Replicates the label from the registry.

"value": Contains the IANA-assigned value corresponding to the 'tls-version' or 'dtls-version'.

"description": Replicates the description from the registry.

"reference": Replicates the reference from the registry and adds the title of the document.

When a (D)TLS parameter is added to "ACL (D)TLS Parameters" registry, a new "type" statement must be added to the iana-tls-profile YANG module. The following "type" statement, and substatements thereof, should be defined:

"derived type": Replicates the parameter name from the registry.

"built-in type": Contains the built-in YANG type.

"description": Replicates the description from the registry.

When the iana-tls-profile YANG module is updated, a new "revision" statement must be added in front of the existing revision statements.

IANA is requested to add this note to "ACL TLS Version Codes", "ACL DTLS Version Codes", and "ACL (D)TLS Parameters" registries:

> When this registry is modified, the YANG module iana-tls-profile must be updated as defined in [RFCXXXX].

10.3. ACL TLS Version registry

IANA is requested to create a new registry titled "ACL TLS Version Codes". Codes in this registry are used as valid values of 'tls-version' parameter. Further assignments are to be made through Expert Review [RFC8126].

Value	Label	Description	Reference
1	tls12	TLS Version 1.2	[RFC5246]
2	tls13	TLS Version 1.3	[RFC8446]

10.4. ACL DTLS version registry

IANA is requested to create a new registry titled "ACL DTLS Version Codes". Codes in this registry are used as valid values of 'dtls-version' parameter. Further assignments are to be made through Expert Review [RFC8126].

Value	Label	Description	Reference
1	dtls12	DTLS Version 1.2	[RFC6347]
2	dtls13	DTLS Version 1.3	[RFC9147]

10.5. ACL (D)TLS Parameters registry

IANA is requested to create a new registry titled "ACL (D)TLS parameters".

The values for all the (D)TLS parameters in the registry are defined in the TLS and DTLS IANA registries (https://www.iana.org/assignments/tls-parameters/tls-parameters.txt and https://www.iana.org/assignments/tls-extensiontype-values/tls-extensiontype-values.txt) excluding the tls-version and dtls-version parameters. Further assignments are to be made through Expert Review [RFC8126]. The registry is initially populated with the following parameters:

Parameter Name	YANG Type	JSON Type	Description
extension-type	uint16	Number	Extension type
supported-group	uint16	Number	Supported group
signature-algorithm	uint16	Number	Signature algorithm
psk-key-exchange-mode	uint8	Number	pre-shared key exchange mode
application-protocol	string	String	Application protocol
cert-compression-algorithm	uint16	Number	Certificate compression algorithm
cipher-algorithm	uint16	Number	Cipher Suite

```
   | tls-version                | enumeration | String | TLS version
                    |
   +---------------------------+-------------+--------+-------------------------
--------------------+
   | dtls-version               | enumeration | String | DTLS version
                    |
   +---------------------------+-------------+--------+-------------------------
--------------------+
```

10.6. MUD Extensions registry

IANA is requested to create a new MUD Extension Name "ietf-mud-tls" in the MUD Extensions IANA registry https://www.iana.org/assignments/mud/mud.xhtml.

11. Acknowledgments

Thanks to Flemming Andreasen, Shashank Jain, Michael Richardson, Piyush Joshi, Eliot Lear, Harsha Joshi, Qin Wu, Mohamed Boucadair, Ben Schwartz, Eric Rescorla, Panwei William, Nick Lamb, Tom Petch, Paul Wouters and Nick Harper for the discussion and comments.

Thanks to Xufeng Liu for YANGDOCTOR review. Thanks to Linda Dunbar for SECDIR review. Thanks to Qin Wu for OPSDIR review.

12. References

12.1. Normative References

[I-D.ietf-netconf-crypto-types]
Watsen, K., "YANG Data Types and Groupings for Cryptography", Work in Progress, Internet-Draft, draft-ietf-netconf-crypto-types-34, 16 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-crypto-types-34>.

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <https://www.rfc-editor.org/info/rfc6234>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

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

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8519] Jethanandani, M., Agarwal, S., Huang, L., and D. Blair, "YANG Data Model for Network Access Control Lists (ACLs)", RFC 8519, DOI 10.17487/RFC8519, March 2019, <https://www.rfc-editor.org/info/rfc8519>.

[RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <https://www.rfc-editor.org/info/rfc8520>.

[RFC8701] Benjamin, D., "Applying Generate Random Extensions And Sustain Extensibility (GREASE) to TLS Extensibility", RFC 8701, DOI 10.17487/RFC8701, January 2020, <https://www.rfc-editor.org/info/rfc8701>.

[RFC8879] Ghedini, A. and V. Vasiliev, "TLS Certificate Compression", RFC 8879, DOI 10.17487/RFC8879, December 2020, <https://www.rfc-editor.org/info/rfc8879>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/info/rfc9147>.

[X690] ITU-T, "Information technology - ASN.1 encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ISO/IEC 8825-1:2002, 2002.

12.2. Informative References

[clear-as-mud]
"Clear as MUD: Generating, Validating and Applying IoT Behaviorial Profiles", October 2019, <https://arxiv.org/pdf/1804.04358.pdf>.

[cryto-vulnerability]
Perez, B., "Exploiting the Windows CryptoAPI Vulnerability", January 2020, <https://media.defense.gov/2020/Jan/14/2002234275/-1/-1/0/CSA-WINDOWS-10-CRYPT-LIB-20190114.PDF>.

[I-D.ietf-tls-esni]
Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-18, 4 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-18>.

[I-D.ietf-uta-tls13-iot-profile]
Tschofenig, H., Fossati, T., and M. Richardson, "TLS/DTLS 1.3 Profiles for the Internet of Things", Work in Progress, Internet-Draft, draft-ietf-uta-tls13-iot-profile-09, 3 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-uta-tls13-iot-profile-09>.

[malware] Anderson, B., Paul, S., and D. McGrew, "Deciphering Malware's use of TLS (without Decryption)", July 2016, <https://arxiv.org/abs/1607.01639>.

[malware-doh]
Cimpanu, C., "First-ever malware strain spotted abusing new DoH (DNS over HTTPS) protocol", July 2019, <https://www.zdnet.com/article/first-ever-malware-strain-spotted-abusing-new-doh-dns-over-https-protocol/>.

[malware-tls]
Anderson, B. and D. McGrew, "TLS Beyond the Browser: Combining End Host and Network Data to Understand Application Behavior", October 2019, <https://dl.acm.org/citation.cfm?id=3355601>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <https://www.rfc-editor.org/info/rfc6066>.

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/info/rfc7301>.

[RFC7366] Gutmann, P., "Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7366, DOI 10.17487/RFC7366, September 2014, <https://www.rfc-editor.org/info/rfc7366>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7525, DOI 10.17487/RFC7525, May 2015, <https://www.rfc-editor.org/info/rfc7525>.

[RFC7925] Tschofenig, H., Ed. and T. Fossati, "Transport Layer Security (TLS) / Datagram Transport Layer Security (DTLS) Profiles for the Internet of Things", RFC 7925, DOI 10.17487/RFC7925, July 2016, <https://www.rfc-editor.org/info/rfc7925>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", RFC 8152, DOI 10.17487/RFC8152, July 2017, <https://www.rfc-editor.org/info/rfc8152>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[RFC8447] Salowey, J. and S. Turner, "IANA Registry Updates for TLS and DTLS", RFC 8447, DOI 10.17487/RFC8447, August 2018, <https://www.rfc-editor.org/info/rfc8447>.

[RFC8472] Popov, A., Ed., Nystroem, M., and D. Balfanz, "Transport Layer Security (TLS) Extension for Token Binding Protocol Negotiation", RFC 8472, DOI 10.17487/RFC8472, October 2018, <https://www.rfc-editor.org/info/rfc8472>.

[RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://www.rfc-editor.org/info/rfc8484>.

[RFC8576] Garcia-Morchon, O., Kumar, S., and M. Sethi, "Internet of Things (IoT) Security: State of the Art and Challenges", RFC 8576, DOI 10.17487/RFC8576, April 2019, <https://www.rfc-editor.org/info/rfc8576>.

[RFC8613] Selander, G., Mattsson, J., Palombini, F., and L. Seitz, "Object Security for Constrained RESTful Environments (OSCORE)", RFC 8613, DOI 10.17487/RFC8613, July 2019, <https://www.rfc-editor.org/info/rfc8613>.

[RFC9462] Pauly, T., Kinnear, E., Wood, C. A., McManus, P., and T. Jensen, "Discovery of Designated Resolvers", RFC 9462, DOI 10.17487/RFC9462, November 2023, <https://www.rfc-editor.org/info/rfc9462>.

[RFC9463] Boucadair, M., Ed., Reddy.K, T., Ed., Wing, D., Cook, N., and T. Jensen, "DHCP and Router Advertisement Options for the Discovery of Network-designated Resolvers (DNR)", RFC 9463, DOI 10.17487/RFC9463, November 2023, <https://www.rfc-editor.org/info/rfc9463>.

[X501] "Information Technology - Open Systems Interconnection - The Directory: Models", ITU-T X.501, 1993.

Authors' Addresses

Tirumaleswar Reddy
Nokia
India
Email: kondtir@gmail.com

Dan Wing
Citrix Systems, Inc.
4988 Great America Pkwy
Santa Clara, CA 95054
United States of America
Email: danwing@gmail.com

Blake Anderson
Cisco Systems, Inc.
170 West Tasman Dr
San Jose, CA 95134
United States of America
Email: blake.anderson@cisco.com

Network Working Group
Internet-Draft
Obsoletes: 3228 (if approved)
Intended status: Best Current Practice
Expires: 15 December 2024

B. Haberman, Ed.
JHU APL

13 June 2024

IANA Considerations for Internet Group Management Protocols
draft-ietf-pim-3228bis-06

Abstract

This document specifies revised IANA Considerations for the Internet Group Management Protocol and the Multicast Listener Discovery protocol. This document specifies the guidance provided to IANA to manage values associated with various fields within the protocol headers of the group management protocols.

This document obsoletes RFC 3228 and unifies guidelines for IPv4 and IPv6 group management protocols.

Status of This Memo

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

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

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

This Internet-Draft will expire on 15 December 2024.

Copyright Notice

Table of Contents

1. Introduction

The following sections describe the allocation guidelines associated with the specified fields within the Internet Group Management Protocol (IGMP) [I-D.ietf-pim-3376bis] and the Multicast Listener Discovery (MLD) [I-D.ietf-pim-3810bis] headers. Some of these registries were created previously, while others are created by this document.

This document obsoletes [RFC3228] and unifies guidelines for IPv4 and IPv6 group management protocols.

1.1. Conventions Used in This Document

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

2. IANA Considerations

The registration procedures used in this document are defined in [RFC8126].

2.1. Type and Code Fields

2.1.1. Internet Group Management Protocol

The IGMP header contains the following fields that carry values assigned from IANA-managed name spaces: Type and Code. Code field values are defined relative to a specific Type value.

[RFC3228] created an IANA registry for the IGMP Type field. This document updates that registry in two ways:

The registration procedure is changed to Standards Action.

The reference for the registry is changed to this document.

[RFC3228] created an IANA registry for Code values for existing IGMP Type fields. The registration procedure for the existing registries is changed to Standards Action. The policy for assigning Code values for new IGMP Types MUST be defined in the document defining the new Type value.

2.1.2. Multicast Listener Discovery

As with IGMP, the MLD header also contains Type and Code fields. Assignment of those fields within the MLD header is defined in [RFC4443].

2.2. Query Message Flags

The IANA is requested to create a single registry for the bits in the Flags field of the Multicast Listener Query Message [I-D.ietf-pim-3810bis] and the IGMPv3 Query Message [I-D.ietf-pim-3376bis]. The format for the registry is:

Resv Bit	Short Name	Description	Reference
0	E	Extension	RFC 9279
1			
2			
3			

The initial contents of this requested registry should contain the E-bit defined in [RFC9279].

The assignment of new bit flags within the Flags field requires Standards Action.

2.3. Report Message Flags

The IANA is requested to create a single registry for the bits in the Flags field of the Multicast Listener Report Message and the IGMPv3 Report Message. The format for the registry is:

Flags Bit	Short Name	Description	Reference
0	E	Extension	RFC 9279
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			

The initial contents of this requested registry should contain the E-bit defined in [RFC9279].

The assignment of new bit flags within the Flags field require Standards Action.

3. Security Considerations

Security analyzers such as firewalls and network intrusion detection monitors often rely on unambiguous interpretations of the fields described in this memo. As new values for the fields are assigned, existing security analyzers that do not understand the new values may fail, resulting in either loss of connectivity if the analyzer declines to forward the unrecognized traffic, or loss of security if it does forward the traffic and the new values are used as part of an attack. This vulnerability argues for high visibility (which the Standards Action process ensures) for the assignments whenever possible.

4. Contributors

Bill Fenner was the author of RFC 3228, which forms a portion of the content contained herein.

5. Acknowledgments

6. References

6.1. Normative References

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

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

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

6.2. Informative References

[I-D.ietf-pim-3376bis] Haberman, B., "Internet Group Management Protocol, Version 3", Work in Progress, Internet-Draft, draft-ietf-pim-3376bis-10, 21 May 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-pim-3376bis-10>.

[I-D.ietf-pim-3810bis]
Haberman, B., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", Work in Progress, Internet-Draft, draft-ietf-pim-3810bis-10, 21 May 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-pim-3810bis-10>.

[RFC3228] Fenner, B., "IANA Considerations for IPv4 Internet Group Management Protocol (IGMP)", BCP 57, RFC 3228, DOI 10.17487/RFC3228, February 2002, <https://www.rfc-editor.org/info/rfc3228>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[RFC9279] Sivakumar, M., Venaas, S., Zhang, Z., and H. Asaeda, "Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Version 2 (MLDv2) Message Extension", RFC 9279, DOI 10.17487/RFC9279, August 2022, <https://www.rfc-editor.org/info/rfc9279>.

Author's Address

Brian Haberman (editor)
Johns Hopkins University Applied Physics Lab
Email: brian@innovationslab.net

Network Working Group
Internet-Draft
Obsoletes: 3376 (if approved)
Updates: 2236 (if approved)
Intended status: Standards Track
Expires: 15 December 2024

B. Haberman, Ed.
JHU APL

13 June 2024

Internet Group Management Protocol, Version 3
draft-ietf-pim-3376bis-11

Abstract

IGMP is the protocol used by IPv4 systems to report their IP multicast group memberships to neighboring multicast routers. Version 3 of IGMP adds support for source filtering, that is, the ability for a system to report interest in receiving packets only from specific source addresses, or from all but specific source addresses, sent to a particular multicast address. That information may be used by multicast routing protocols to avoid delivering multicast packets from specific sources to networks where there are no interested receivers.

This document specifies Version 3 of the Internet Group Management Protocol, IGMPv3. It is a revised version of the specification to include clarifications and fixes for errata in RFC 3376 and is backwards compatible with RFC 3376.

This document updates RFC 2236 and obsoletes RFC 3376.

Status of This Memo

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

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

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

This Internet-Draft will expire on 15 December 2024.

Table of Contents

1. Introduction

The Internet Group Management Protocol (IGMP) is used by IPv4 systems (hosts and routers) to report their IP multicast group memberships to any neighboring multicast routers. Note that an IP multicast router may itself be a member of one or more multicast groups, in which case it performs both the multicast router part of the protocol (to collect the membership information needed by its multicast routing protocol) and the group member part of the protocol (to inform itself and other, neighboring multicast routers of its memberships).

IGMP is also used for other IP multicast management functions, using message types other than those used for group membership reporting. This document specifies only the group membership reporting functions and messages.

This document specifies Version 3 of IGMP. Version 1, specified in [RFC1112], was the first widely-deployed version and the first version to become an Internet Standard. Version 2, specified in [RFC2236], added support for low leave latency, that is, a reduction in the time it takes for a multicast router to learn that there are no longer any members of a particular group present on an attached network. Version 3 adds support for source filtering, that is, the ability for a system to report interest in receiving packets only from specific source addresses, as required to support Source-Specific Multicast [RFC3569], or from all but specific source addresses, sent to a particular multicast address. Version 3 is designed to be interoperable with Versions 1 and 2.

This document uses SSM-aware to refer to systems that support Source-Specific Multicast (SSM) as defined in [RFC4607].

This document updates [RFC2236] as a proper implementation of Version 3 of IGMP needs to implement Version 2 Report and Leave message handling.

This document obsoletes [RFC3376] as it provides clarifications and fixes for errata in RFC 3376. Detailed updates for those changes are described in Appendix C.

1.1. Conventions Used in This Document

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

2. The Service Interface for Requesting IP Multicast Reception

Within an IP system, there is (at least conceptually) a service interface used by upper-layer protocols or application programs to ask the IP layer to enable and disable reception of packets sent to specific IP multicast addresses. In order to take full advantage of the capabilities of IGMPv3, a system's IP service interface must support the following operation:

```
IPMulticastListen ( socket, interface, multicast-address,
                    filter-mode, source-list )
```

where:

* "socket" is an implementation-specific parameter used to distinguish among different requesting entities (e.g., programs or processes) within the system; the socket parameter of BSD Unix system calls is a specific example.

* "interface" is a local identifier of the network interface on which reception of the specified multicast address is to be enabled or disabled. Interfaces may be physical (e.g., an Ethernet interface) or virtual (e.g., the endpoint of a Frame Relay virtual circuit or the endpoint of an IP-in-IP "tunnel"). An implementation may allow a special "unspecified" value to be passed as the interface parameter, in which case the request would apply to the "primary" or "default" interface of the system (perhaps established by system configuration). If reception of the same multicast address is desired on more than one interface, IPMulticastListen is invoked separately for each desired interface.

* "multicast-address" is the IP multicast address, or group, to which the request pertains. If reception of more than one multicast address on a given interface is desired, IPMulticastListen is invoked separately for each desired multicast address.

* "filter-mode" may be either INCLUDE or EXCLUDE. In INCLUDE mode, reception of packets sent to the specified multicast address is requested only from those IP source addresses listed in the source-list parameter. In EXCLUDE mode, reception of packets sent to the given multicast address is requested from all IP source addresses except those listed in the source-list parameter.

* "source-list" is an unordered list of zero or more IP unicast addresses from which multicast reception is desired or not desired, depending on the filter mode. An implementation MAY impose a limit on the size of source lists, but that limit MUST NOT be less than 64 addresses per list. When an operation causes the source list size limit to be exceeded, the service interface MUST return an error.

For a given combination of socket, interface, and multicast address, only a single filter mode and source list can be in effect at any one time. However, either the filter mode or the source list, or both, may be changed by subsequent IPMulticastListen requests that specify the same socket, interface, and multicast address. Each subsequent request completely replaces any earlier request for the given socket, interface and multicast address.

Previous versions of IGMP did not support source filters and had a simpler service interface consisting of Join and Leave operations to enable and disable reception of a given multicast address (from all sources) on a given interface. The equivalent operations in the new service interface follow:

The Join operation is equivalent to:

```
IPMulticastListen ( socket, interface, multicast-address,
                    EXCLUDE, {} )
```

and the Leave operation is equivalent to:

```
IPMulticastListen ( socket, interface, multicast-address,
                    INCLUDE, {} )
```

where {} is an empty source list.

An example of an API providing the capabilities outlined in this service interface is in [RFC3678].

3. Multicast Reception State Maintained by Systems

3.1. Socket State

For each socket on which IPMulticastListen has been invoked, the system records the desired multicast reception state for that socket. That state conceptually consists of a set of records of the form:

```
(interface, multicast-address, filter-mode, source-list)
```

The socket state evolves in response to each invocation of IPMulticastListen on the socket, as follows:

* If the requested filter mode is INCLUDE and the requested source list is empty, then the entry corresponding to the requested interface and multicast address is deleted if present. If no such entry is present, the request is ignored.

* If the requested filter mode is EXCLUDE or the requested source list is non-empty, then the entry corresponding to the requested interface and multicast address, if present, is changed to contain the requested filter mode and source list. If no such entry is present, a new entry is created, using the parameters specified in the request.

3.2. Interface State

In addition to the per-socket multicast reception state, a system must also maintain or compute multicast reception state for each of its interfaces. That state conceptually consists of a set of records of the form:

(multicast-address, filter-mode, source-list)

At most one record per multicast-address exists for a given interface. This per-interface state is derived from the per-socket state, but may differ from the per-socket state when different sockets have differing filter modes and/or source lists for the same multicast address and interface. For example, suppose one application or process invokes the following operation on socket s1:

IPMulticastListen (s1, i, m, INCLUDE, {a, b, c})

requesting reception on interface i of packets sent to multicast address m, only if they come from source a, b, or c. Suppose another application or process invokes the following operation on socket s2:

IPMulticastListen (s2, i, m, INCLUDE, {b, c, d})

requesting reception on the same interface i of packets sent to the same multicast address m, only if they come from sources b, c, or d. In order to satisfy the reception requirements of both sockets, it is necessary for interface i to receive packets sent to m from any one of the sources a, b, c, or d. Thus, in this example, the reception state of interface i for multicast address m has filter mode INCLUDE and source list {a, b, c, d}.

After a multicast packet has been accepted from an interface by the IP layer, its subsequent delivery to the application or process listening on a particular socket depends on the multicast reception state of that socket [and possibly also on other conditions, such as what transport-layer port the socket is bound to]. So, in the above example, if a packet arrives on interface i, destined to multicast address m, with source address a, it will be delivered on socket s1 but not on socket s2. Note that IGMP Queries and Reports are not subject to source filtering and must always be processed by hosts and routers.

Filtering of packets based upon a socket's multicast reception state is a new feature of this service interface. The previous service interface [RFC1112] described no filtering based upon multicast join state; rather, a join on a socket simply caused the host to join a group on the given interface, and packets destined for that group could be delivered to all sockets whether they had joined or not.

The general rules for deriving the per-interface state from the per-socket state are as follows: For each distinct (interface, multicast-address) pair that appears in any socket state, a per- interface record is created for that multicast address on that interface. Considering all socket records containing the same (interface, multicast-address) pair,

* if any such record has a filter mode of EXCLUDE, then the filter mode of the interface record is EXCLUDE, and the source list of the interface record is the intersection of the source lists of all socket records in EXCLUDE mode, minus those source addresses that appear in any socket record in INCLUDE mode. For example, if the socket records for multicast address m on interface i are:

  from socket s1: (i, m, EXCLUDE, {a, b, c, d})

  from socket s2: (i, m, EXCLUDE, {b, c, d, e})

  from socket s3: (i, m, INCLUDE, {d, e, f})

  then the corresponding interface record on interface i is:

  (m, EXCLUDE, {b, c})

  If a fourth socket is added, such as:

  from socket s4: (i, m, EXCLUDE, {})

  then the interface record becomes:

  (m, EXCLUDE, {})

* if all such records have a filter mode of INCLUDE, then the filter mode of the interface record is INCLUDE, and the source list of the interface record is the union of the source lists of all the socket records. For example, if the socket records for multicast address m on interface i are:

  from socket s1: (i, m, INCLUDE, {a, b, c})

  from socket s2: (i, m, INCLUDE, {b, c, d})

from socket s3: (i, m, INCLUDE, {e, f})

then the corresponding interface record on interface i is:

(m, INCLUDE, {a, b, c, d, e, f})

An implementation MUST NOT use an EXCLUDE interface record to represent a group when all sockets for this group are in INCLUDE state. If system resource limits are reached when an interface state source list is calculated, an error MUST be returned to the application which requested the operation.

The above rules for deriving the interface state are (re-)evaluated whenever an IPMulticastListen invocation modifies the socket state by adding, deleting, or modifying a per-socket state record. Note that a change of socket state does not necessarily result in a change of interface state.

4. Message Formats

IGMP messages are encapsulated in IPv4 datagrams, with an IP protocol number of 2. Every IGMP message described in this document is sent with an IP Time-to-Live of 1, IP Precedence of Internetwork Control (e.g., Type of Service 0xc0), and carries an IP Router Alert option [RFC2113] in its IP header. IGMP message types are registered per [I-D.ietf-pim-3228bis].

There are two IGMP message types of concern to the IGMPv3 protocol described in this document:

Type Number (hex)	Message Name
0x11	Membership Query
0x22	Version 3 Membership Report

Table 1: New messages introduced by IGMP3

An implementation of IGMPv3 MUST also support the following three message types, for interoperation with previous versions of IGMP (see Section 7):

Type Number (hex)	Message Name	Reference
0x12	Version 1 Membership Report	[RFC1112]
0x16	Version 2 Membership Report	[RFC2236]
0x17	Version 2 Leave Group	[RFC2236]

Table 2: Legacy IGMP messages

Unrecognized message types MUST be silently ignored. Other message types may be used by newer versions or extensions of IGMP, by multicast routing protocols, or for other uses.

In this document, unless otherwise qualified, the capitalized words "Query" and "Report" refer to IGMP Membership Queries and IGMP Version 3 Membership Reports, respectively.

4.1. Membership Query Message

Membership Queries are sent by IP multicast routers to query the multicast reception state of neighboring interfaces. Queries have the following format:

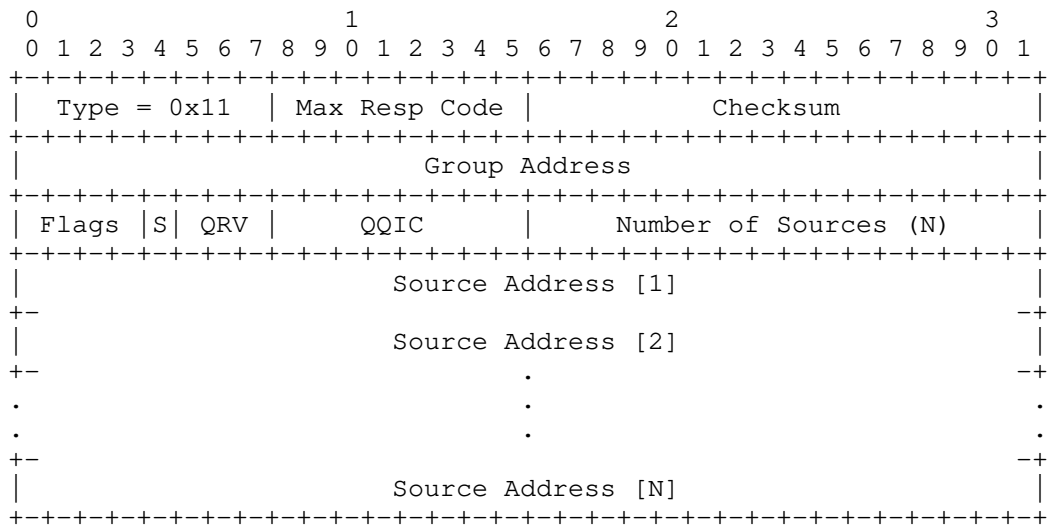


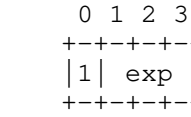
Figure 1: IGMPv3 Query Message

4.1.1. Max Resp Code

The Max Resp Code field specifies the maximum time allowed before sending a responding report. The actual time allowed, called the Max Resp Time, is represented in units of 1/10 second and is derived from the Max Resp Code as follows:

If Max Resp Code < 128, Max Resp Time = Max Resp Code

If Max Resp Code >= 128, Max Resp Code represents a floating-point value as follows:




```
Max Resp Time = (mant | 0x10) << (exp + 3)
```

Figure 2: Max Resp Code Representation

Small values of Max Resp Time allow IGMPv3 routers to tune the "leave latency" (the time between the moment the last host leaves a group and the moment the routing protocol is notified that there are no more members). Larger values, especially in the exponential range, allow tuning of the burstiness of IGMP traffic on a network.

4.1.2. Checksum

The Checksum is the 16-bit one's complement of the one's complement sum of the whole IGMP message (the entire IP payload). For computing the checksum, the Checksum field is set to zero. When receiving packets, the checksum MUST be verified before processing a packet [RFC1071].

4.1.3. Group Address

The Group Address field is set to zero when sending a General Query, and set to the IP multicast address being queried when sending a Group-Specific Query or Group-and-Source-Specific Query (see Section Section 4.1.9, below).

4.1.4. Flags

The Flags field is a bitstring managed by an IANA registry defined in [I-D.ietf-pim-3228bis].

4.1.5. S Flag (Suppress Router-Side Processing)

When set to one, the S Flag indicates to any receiving multicast routers that they are to suppress the normal timer updates they perform upon hearing a Query. It does not, however, suppress the querier election or the normal "host-side" processing of a Query that a router may be required to perform as a consequence of itself being a group member.

4.1.6. QRV (Querier's Robustness Variable)

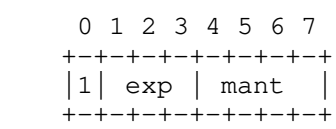
If non-zero, the QRV field contains the [Robustness Variable] value used by the querier, i.e., the sender of the Query. If the querier's [Robustness Variable] exceeds 7, the maximum value of the QRV field, the QRV is set to zero. Routers adopt the QRV value from the most recently received Query as their own [Robustness Variable] value, unless that most recently received QRV was zero, in which case the receivers use the default [Robustness Variable] value specified in section Section 8.1 or a statically configured value.

4.1.7. QQIC (Querier's Query Interval Code)

The Querier's Query Interval Code field specifies the [Query Interval] used by the querier. The actual interval, called the Querier's Query Interval (QQI), is represented in units of seconds and is derived from the Querier's Query Interval Code as follows:

If QQIC < 128, QQI = QQIC

If QQIC >= 128, QQIC represents a floating-point value as follows:



QQI = (mant | 0x10) << (exp + 3)

Figure 3: QQIC Representation

Multicast routers that are not the current querier adopt the QQI value from the most recently received Query as their own [Query Interval] value, unless that most recently received QQI was zero, in which case the receiving routers use the default [Query Interval] value specified in Section 8.2.

4.1.8. Number of Sources (N)

The Number of Sources (N) field specifies how many source addresses are present in the Query. This number is zero in a General Query or a Group-Specific Query, and non-zero in a Group-and-Source-Specific Query. This number is limited by the MTU of the network over which the Query is transmitted. For example, on an Ethernet with an MTU of 1500 octets, the IP header including the Router Alert option consumes 24 octets, and the IGMP fields up to including the Number of Sources (N) field consume 12 octets, leaving 1464 octets for source addresses, which limits the number of source addresses to 366 (1464/4).

4.1.9. Source Address [i]

The Source Address [i] fields are a vector of n IP unicast addresses, where n is the value in the Number of Sources (N) field.

4.1.10. Additional Data

If the Packet Length field in the IP header of a received Query indicates that there are additional octets of data present, beyond the fields described here, IGMPv3 implementations MUST include those octets in the computation to verify the received IGMP Checksum, but MUST otherwise ignore those additional octets. When sending a Query, an IGMPv3 implementation MUST NOT include additional octets beyond the fields described here.

4.1.11. Query Variants

There are three variants of the Query message:

1. A General Query is sent by a multicast router to learn the complete multicast reception state of the neighboring interfaces (that is, the interfaces attached to the network on which the Query is transmitted). In a General Query, both the Group Address field and the Number of Sources (N) field are zero.

2. A Group-Specific Query is sent by a multicast router to learn the reception state, with respect to a single multicast address, of the neighboring interfaces. In a Group-Specific Query, the Group Address field contains the multicast address of interest, and the Number of Sources (N) field contains zero.

3. A Group-and-Source-Specific Query is sent by a multicast router to learn if any neighboring interface desires reception of packets sent to a specified multicast address, from any of a specified list of sources. In a Group-and-Source-Specific Query,

the Group Address field contains the multicast address of interest, and the Source Address [i] fields contain the source address(es) of interest.

4.1.12. IP Destination Addresses for Queries

In IGMPv3, General Queries are sent with an IP destination address of 224.0.0.1, the all-systems multicast address. Group-Specific and Group-and-Source-Specific Queries are sent with an IP destination address equal to the multicast address of interest. However, a system MUST accept and process any Query whose IP Destination Address field contains any of the addresses (unicast or multicast) assigned to the interface on which the Query arrives.

4.2. Version 3 Membership Report Message

Version 3 Membership Reports are sent by IP systems to report (to neighboring routers) the current multicast reception state, or changes in the multicast reception state, of their interfaces. Reports have the following format:

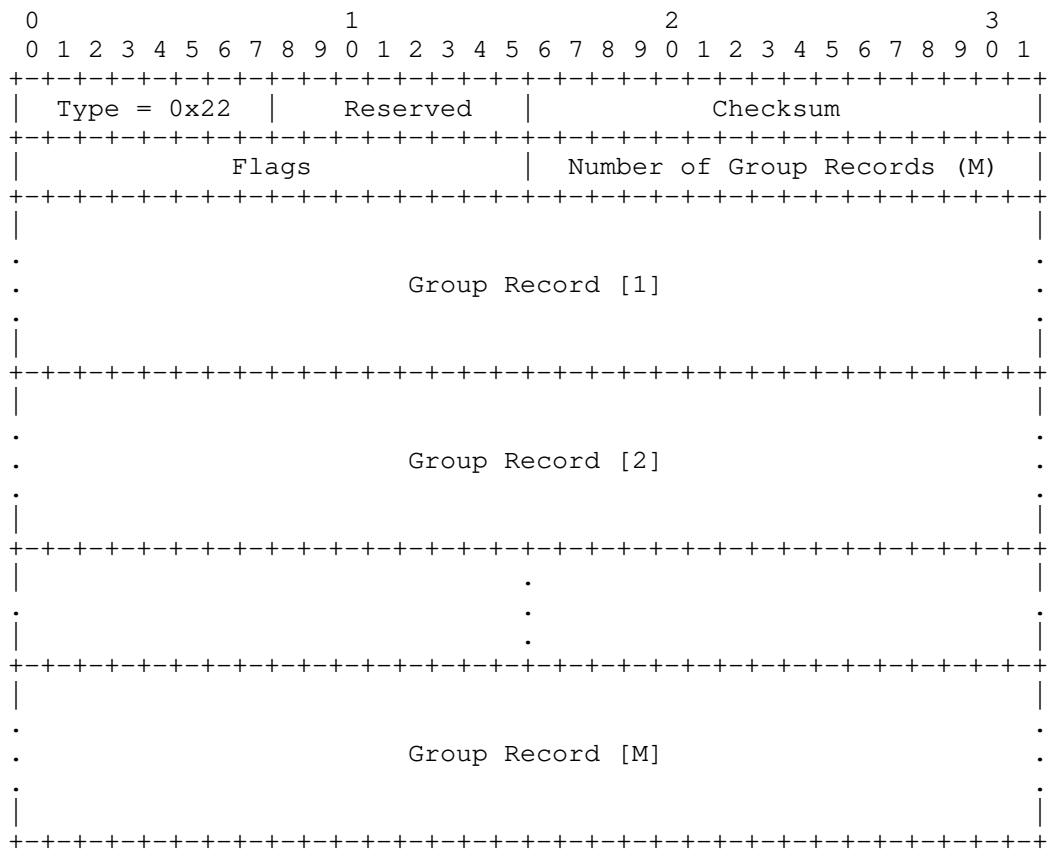


Figure 4: IGMPv3 Report Message

where each Group Record has the following internal format:

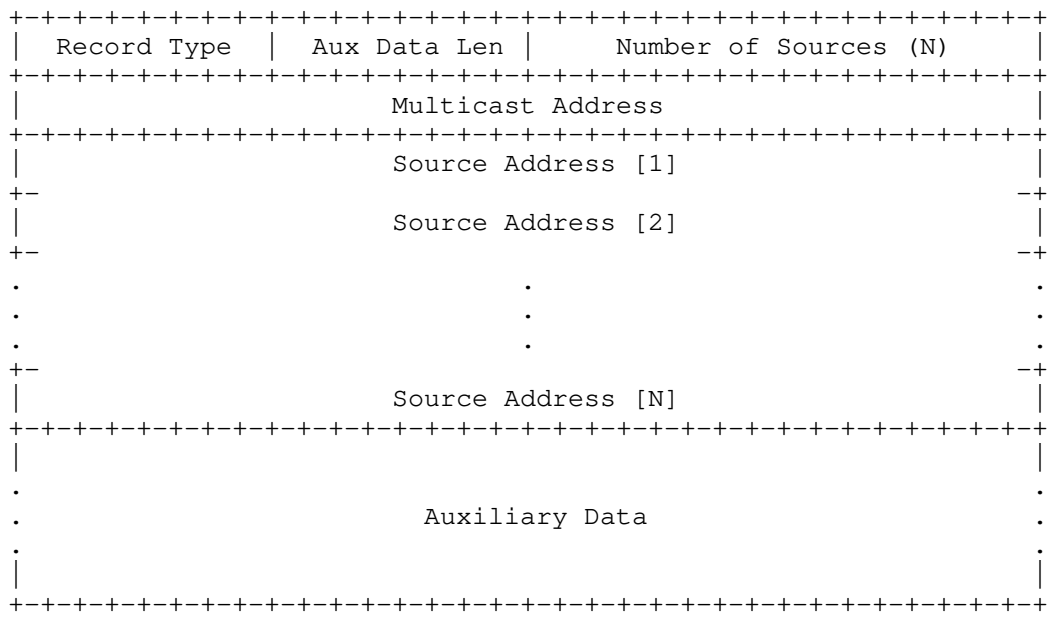


Figure 5: IGMPv3 Report Group Record

4.2.1. Reserved

The Reserved field is set to zero on transmission, and ignored on reception.

4.2.2. Checksum

The Checksum is the 16-bit one's complement of the one's complement sum of the whole IGMP message (the entire IP payload). For computing the checksum, the Checksum field is set to zero. When receiving packets, the checksum MUST be verified before processing a message.

4.2.3. Flags

The Flags field is a bitstring managed by an IANA registry defined in [I-D.ietf-pim-3228bis].

4.2.4. Number of Group Records (M)

The Number of Group Records (M) field specifies how many Group Records are present in this Report.

4.2.5. Group Record

Each Group Record is a block of fields containing information pertaining to the sender's membership in a single multicast group on the interface from which the Report is sent.

4.2.6. Record Type

See section Section 4.2.13, below.

4.2.7. Aux Data Len

The Aux Data Len field contains the length of the Auxiliary Data field in this Group Record, in units of 32-bit words. It may contain zero, to indicate the absence of any auxiliary data.

4.2.8. Number of Sources (N)

The Number of Sources (N) field specifies how many source addresses are present in this Group Record.

4.2.9. Multicast Address

The Multicast Address field contains the IP multicast address to which this Group Record pertains.

4.2.10. Source Address [i]

The Source Address [i] fields are a vector of n IP unicast addresses, where n is the value in this record's Number of Sources (N) field.

4.2.11. Auxiliary Data

The Auxiliary Data field, if present, contains additional information pertaining to this Group Record. The protocol specified in this document, IGMPv3, does not define any auxiliary data. Therefore, implementations of IGMPv3 MUST NOT include any auxiliary data (i.e., MUST set the Aux Data Len field to zero) in any transmitted Group Record, and MUST ignore any auxiliary data present in any received Group Record. The semantics and internal encoding of the Auxiliary Data field are to be defined by any future version or extension of IGMP that uses this field.

4.2.12. Additional Data

If the Packet Length field in the IP header of a received Report indicates that there are additional octets of data present, beyond the last Group Record, IGMPv3 implementations MUST include those octets in the computation to verify the received IGMP Checksum, but MUST otherwise ignore those additional octets. When sending a Report, an IGMPv3 implementation MUST NOT include additional octets beyond the last Group Record.

4.2.13. Group Record Types

There are a number of different types of Group Records that may be included in a Report message:

* A Current-State Record is sent by a system in response to a Query received on an interface. It reports the current reception state of that interface, with respect to a single multicast address. The Record Type of a Current-State Record may be one of the following two values:

  1 - MODE_IS_INCLUDE - indicates that the interface has a filter mode of INCLUDE for the specified multicast address. The Source Address [i] fields in this Group Record contain the interface's source list for the specified multicast address, if it is non-empty.

  2 - MODE_IS_EXCLUDE - indicates that the interface has a filter mode of EXCLUDE for the specified multicast address. The Source Address [i] fields in this Group Record contain the interface's source list for the specified multicast address, if it is non-empty. An SSM-aware host SHOULD NOT send a MODE_IS_EXCLUDE record type for multicast addresses that fall within the SSM address range as they will be ignored by SSM-aware routers [RFC4604].

* A Filter-Mode-Change Record is sent by a system whenever a local invocation of IPMulticastListen causes a change of the filter mode (i.e., a change from INCLUDE to EXCLUDE, or from EXCLUDE to INCLUDE), of the interface-level state entry for a particular multicast address. The Record is included in a Report sent from the interface on which the change occurred. The Record Type of a Filter-Mode-Change Record may be one of the following two values:

3 - CHANGE_TO_INCLUDE_MODE - indicates that the interface has changed to INCLUDE filter mode for the specified multicast address. The Source Address [i] fields in this Group Record contain the interface's new source list for the specified multicast address, if it is non-empty.

4 - CHANGE_TO_EXCLUDE_MODE - indicates that the interface has changed to EXCLUDE filter mode for the specified multicast address. The Source Address [i] fields in this Group Record contain the interface's new source list for the specified multicast address, if it is non-empty. An SSM-aware host SHOULD NOT send a CHANGE_TO_EXCLUDE_MODE record type for multicast addresses that fall within the SSM address range.

* A Source-List-Change Record is sent by a system whenever a local invocation of IPMulticastListen causes a change of source list that is not coincident with a change of filter mode, of the interface-level state entry for a particular multicast address. The Record is included in a Report sent from the interface on which the change occurred. The Record Type of a Source-List-Change Record may be one of the following two values:

5 - ALLOW_NEW_SOURCES - indicates that the Source Address [i] fields in this Group Record contain a list of the additional sources that the system wishes to hear from, for packets sent to the specified multicast address. If the change was to an INCLUDE source list, these are the addresses that were added to the list; if the change was to an EXCLUDE source list, these are the addresses that were deleted from the list.

6 - BLOCK_OLD_SOURCES - indicates that the Source Address [i] fields in this Group Record contain a list of the sources that the system no longer wishes to hear from, for packets sent to the specified multicast address. If the change was to an INCLUDE source list, these are the addresses that were deleted from the list; if the change was to an EXCLUDE source list, these are the addresses that were added to the list.

If a change of source list results in both allowing new sources and blocking old sources, then two Group Records are sent for the same multicast address, one of type ALLOW_NEW_SOURCES and one of type BLOCK_OLD_SOURCES.

We use the term State-Change Record to refer to either a Filter-Mode-Change Record or a Source-List-Change Record.

Unrecognized Record Type values MUST be silently ignored.

4.2.14. IP Source Addresses for Reports

An IGMP report is sent with a valid IP source address for the destination subnet. The 0.0.0.0 source address may be used by a system that has not yet acquired an IP address. Note that the 0.0.0.0 source address may simultaneously be used by multiple systems on a LAN. Routers MUST accept a report with a source address of 0.0.0.0.

4.2.15. IP Destination Addresses for Reports

Version 3 Reports are sent with an IP destination address of 224.0.0.22, to which all IGMPv3-capable multicast routers listen. A system that is operating in version 1 or version 2 compatibility modes sends version 1 or version 2 Reports to the multicast group specified in the Group Address field of the Report. In addition, a system MUST accept and process any version 1 or version 2 Report whose IP Destination Address field contains any of the addresses (unicast or multicast) assigned to the interface on which the Report arrives.

4.2.16. Notation for Group Records

In the rest of this document, we use the following notation to describe the contents of a Group Record pertaining to a particular multicast address:

```
IS_IN ( x )  -  Type MODE_IS_INCLUDE, source addresses x
IS_EX ( x )  -  Type MODE_IS_EXCLUDE, source addresses x
TO_IN ( x )  -  Type CHANGE_TO_INCLUDE_MODE, source addresses x
TO_EX ( x )  -  Type CHANGE_TO_EXCLUDE_MODE, source addresses x
ALLOW ( x )  -  Type ALLOW_NEW_SOURCES, source addresses x
BLOCK ( x )  -  Type BLOCK_OLD_SOURCES, source addresses x
```

where x is either:

* a capital letter (e.g., "A") to represent the set of source addresses, or

* a set expression (e.g., "A+B"), where "A+B" means the union of sets A and B, "A*B" means the intersection of sets A and B, and "A-B" means the removal of all elements of set B from set A.

4.2.17. Membership Report Size

If the set of Group Records required in a Report does not fit within the size limit of a single Report message (as determined by the MTU of the network on which it will be sent), the Group Records are sent in as many Report messages as needed to report the entire set.

If a single Group Record contains so many source addresses that it does not fit within the size limit of a single Report message, if its Type is not MODE_IS_EXCLUDE or CHANGE_TO_EXCLUDE_MODE, it is split into multiple Group Records, each containing a different subset of the source addresses and each sent in a separate Report message. If its Type is MODE_IS_EXCLUDE or CHANGE_TO_EXCLUDE_MODE, a single Group Record is sent, containing as many source addresses as can fit, and

the remaining source addresses are not reported; though the choice of which sources to report is arbitrary, it is preferable to report the same set of sources in each subsequent report, rather than reporting different sources each time.

5. Description of the Protocol for Group Members

IGMP is an asymmetric protocol, specifying separate behaviors for group members -- that is, hosts or routers that wish to receive multicast packets -- and multicast routers. This section describes the part of IGMPv3 that applies to all group members. (Note that a multicast router that is also a group member performs both parts of IGMPv3, receiving and responding to its own IGMP message transmissions as well as those of its neighbors. The multicast router part of IGMPv3 is described in Section 6.)

A system performs the protocol described in this section over all interfaces on which multicast reception is supported, even if more than one of those interfaces is connected to the same network.

For interoperability with multicast routers running older versions of IGMP, systems maintain a MulticastRouterVersion variable for each interface on which multicast reception is supported. This section describes the behavior of group member systems on interfaces for which MulticastRouterVersion = 3. The algorithm for determining MulticastRouterVersion, and the behavior for versions other than 3, are described in Section 7.

The all-systems multicast address, 224.0.0.1, is handled as a special case. On all systems -- that is all hosts and routers, including multicast routers -- reception of packets destined to the all-systems multicast address, from all sources, is permanently enabled on all interfaces on which multicast reception is supported. No IGMP messages are ever sent regarding the all-systems multicast address.

There are two types of events that trigger IGMPv3 protocol actions on an interface:

* a change of the interface reception state, caused by a local invocation of IPMulticastListen.

* reception of a Query.

(Received IGMP messages of types other than Query are silently ignored, except as required for interoperation with earlier versions of IGMP.)

The following subsections describe the actions to be taken for each of these two cases. In those descriptions, timer and counter names appear in square brackets. The default values for those timers and counters are specified in Section 8.

5.1. Action on Change of Interface State

An invocation of IPMulticastListen may cause the multicast reception state of an interface to change, according to the rules in Section Section 3.2. Each such change affects the per-interface entry for a single multicast address.

A change of interface state causes the system to immediately transmit a State-Change Report from that interface. The type and contents of the Group Record(s) in that Report are determined by comparing the filter mode and source list for the affected multicast address before and after the change, according to the table below. If no interface state existed for that multicast address before the change (i.e., the change consisted of creating a new per-interface record), or if no state exists after the change (i.e., the change consisted of deleting a per-interface record), then the "non-existent" state is considered to have a filter mode of INCLUDE and an empty source list.

Old State	New State	State-Change Record Sent
INCLUDE (A)	INCLUDE (B)	ALLOW (B-A), BLOCK (A-B)
EXCLUDE (A)	EXCLUDE (B)	ALLOW (A-B), BLOCK (B-A)
INCLUDE (A)	EXCLUDE (B)	TO_EX (B)
EXCLUDE (A)	INCLUDE (B)	TO_IN (B)

Table 3

If the computed source list for either an ALLOW or a BLOCK State-Change Record is empty, that record is omitted from the Report message.

To cover the possibility of the State-Change Report being missed by one or more multicast routers, it is retransmitted [Robustness Variable] - 1 more times, at intervals chosen at random from the range (0, [Unsolicited Report Interval]).

If more changes to the same interface state entry occur before all the retransmissions of the State-Change Report for the first change have been completed, each such additional change triggers the immediate transmission of a new State-Change Report.

The contents of the new transmitted report are calculated as follows. As was done with the first report, the interface state for the affected group before and after the latest change is compared. The report records expressing the difference are built according to the table above. However these records are not transmitted in a message but instead merged with the contents of the pending report, to create the new State-Change report. The rules for merging the difference report resulting from the state change and the pending report are described below.

The transmission of the merged State-Change Report terminates retransmissions of the earlier State-Change Reports for the same multicast address, and becomes the first of [Robustness Variable] transmissions of State-Change Reports.

Each time a source is included in the difference report calculated above, retransmission state for that source needs to be maintained until [Robustness Variable] State-Change reports have been sent by the host. This is done in order to ensure that a series of successive state changes do not break the protocol robustness.

If the interface reception-state change that triggers the new report is a filter-mode change, then the next [Robustness Variable] State-Change Reports will include a Filter-Mode-Change record. This applies even if any number of source-list changes occur in that period. The host has to maintain retransmission state for the group until the [Robustness Variable] State-Change reports have been sent. When [Robustness Variable] State-Change reports with Filter-Mode-Change records have been transmitted after the last filter-mode change, and if source-list changes to the interface reception have scheduled additional reports, then the next State-Change report will include Source-List-Change records.

Each time a State-Change Report is transmitted, the contents are determined as follows. If the report should contain a Filter-Mode-Change record, then if the current filter-mode of the interface is INCLUDE, a TO_IN record is included in the report, otherwise a TO_EX record is included. If instead the report should contain Source-List-Change records, an ALLOW and a BLOCK record are included. The contents of these records are built according to the table below.

Record	Sources Included
TO_IN	All in the current interface state that must be forwarded
TO_EX	All in the current interface state that must be blocked
ALLOW	All with retransmission state that must be forwarded
BLOCK	All with retransmission state that must be blocked

Table 4

If the computed source list for either an ALLOW or a BLOCK record is empty, that record is omitted from the State-Change report.

Note: When the first State-Change report is sent, the non-existent pending report to merge with, can be treated as a source-change report with empty ALLOW and BLOCK records (no sources have retransmission state).

5.2. Action on Reception of a Query

When a system receives a Query, it does not respond immediately. Instead, it delays its response by a random amount of time, bounded by the Max Resp Time value derived from the Max Resp Code in the received Query message. A system may receive a variety of Queries on different interfaces and of different kinds (e.g., General Queries, Group-Specific Queries, and Group-and-Source-Specific Queries), each of which may require its own delayed response.

Before scheduling a response to a Query, the system must first consider previously scheduled pending responses and in many cases schedule a combined response. Therefore, the system must be able to maintain the following state:

* A timer per interface for scheduling responses to General Queries.

* A per-group and interface timer for scheduling responses to Group-Specific and Group-and-Source-Specific Queries.

* A per-group and interface list of sources to be reported in the response to a Group-and-Source-Specific Query.

When a new Query with the Router-Alert option arrives on an interface, provided the system has state to report, a delay for a response is randomly selected in the range (0, [Max Resp Time]) where Max Resp Time is derived from Max Resp Code in the received Query message. The following rules are then used to determine if a Report needs to be scheduled and the type of Report to schedule. The rules are considered in order and only the first matching rule is applied.

1. If there is a pending response to a previous General Query scheduled sooner than the selected delay, no additional response needs to be scheduled.

2. If the received Query is a General Query, the interface timer is used to schedule a response to the General Query after the selected delay. Any previously pending response to a General Query is canceled.

3. If the received Query is a Group-Specific Query or a Group-and-Source-Specific Query and there is no pending response to a previous Query for this group, then the group timer is used to schedule a report. If the received Query is a Group-and-Source-Specific Query, the list of queried sources is recorded to be used when generating a response.

4. If there already is a pending response to a previous Query scheduled for this group, and either the new Query is a Group-Specific Query or the recorded source-list associated with the group is empty, then the group source-list is cleared and a single response is scheduled using the group timer. The new response is scheduled to be sent at the earliest of the remaining time for the pending report and the selected delay.

5. If the received Query is a Group-and-Source-Specific Query and there is a pending response for this group with a non-empty source-list, then the group source list is augmented to contain the list of sources in the new Query and a single response is scheduled using the group timer. The new response is scheduled to be sent at the earliest of the remaining time for the pending report and the selected delay.

When the timer in a pending response record expires, the system transmits, on the associated interface, one or more Report messages carrying one or more Current-State Records (see section Section 4.2.13), as follows:

1. If the expired timer is the interface timer (i.e., it is a pending response to a General Query), then one Current-State Record is sent for each multicast address for which the specified interface has reception state, as described in Section 3.2. The Current- State Record carries the multicast address and its associated filter mode (MODE_IS_INCLUDE or MODE_IS_EXCLUDE) and source list. Multiple Current-State Records are packed into individual Report messages, to the extent possible.

   This naive algorithm may result in bursts of packets when a system is a member of a large number of groups. Instead of using a single interface timer, implementations are recommended to spread transmission of such Report messages over the interval (0, [Max Resp Time]). Note that any such implementation MUST avoid the "ack-implosion" problem, i.e., MUST NOT send a Report immediately on reception of a General Query.

2. If the expired timer is a group timer and the list of recorded sources for the that group is empty (i.e., it is a pending response to a Group-Specific Query), then if and only if the interface has reception state for that group address, a single Current-State Record is sent for that address. The Current-State Record carries the multicast address and its associated filter mode (MODE_IS_INCLUDE or MODE_IS_EXCLUDE) and source list.

3. If the expired timer is a group timer and the list of recorded sources for that group is non-empty (i.e., it is a pending response to a Group-and-Source-Specific Query), then if and only if the interface has reception state for that group address, the contents of the responding Current-State Record is determined from the interface state and the pending response record, as specified in the following table:

Per-Interface State	Set of Sources in the Pending Response Record	Current-State Record
INCLUDE (A)	B	IS_IN (A*B)
EXCLUDE (A)	B	IS_IN (B-A)

Table 5

If the resulting Current-State Record has an empty set of source addresses, then no response is sent.

Finally, after any required Report messages have been generated, the source lists associated with any reported groups are cleared.

6. Description of the Protocol for Multicast Routers

The purpose of IGMP is to enable each multicast router to learn, for each of its directly attached networks, which multicast addresses are of interest to the systems attached to those networks. IGMP version 3 adds the capability for a multicast router to also learn which sources are of interest to neighboring systems, for packets sent to any particular multicast address. The information gathered by IGMP is provided to whichever multicast routing protocol is being used by the router, in order to ensure that multicast packets are delivered to all networks where there are interested receivers.

This section describes the part of IGMPv3 that is performed by multicast routers. Multicast routers may also themselves become members of multicast groups, and therefore also perform the group member part of IGMPv3, described in Section 5.

A multicast router performs the protocol described in this section over each of its directly-attached networks. If a multicast router has more than one interface to the same network, it only needs to operate this protocol over one of those interfaces. On each interface over which this protocol is being run, the router MUST enable reception of multicast address 224.0.0.22, from all sources (and MUST perform the group member part of IGMPv3 for that address on that interface).

Multicast routers need to know only that at least one system on an attached network is interested in packets to a particular multicast address from a particular source; a multicast router is not required to keep track of the interests of each individual neighboring system. (However, see Appendix A.2 point 1 for discussion.)

IGMPv3 is backward compatible with previous versions of the IGMP protocol. In order to remain backward compatible with older IGMP systems, IGMPv3 multicast routers MUST also implement versions 1 and 2 of the protocol (see section Section 7).

6.1. Conditions for IGMP Queries

Multicast routers send General Queries periodically to request group membership information from an attached network. These queries are used to build and refresh the group membership state of systems on attached networks. Systems respond to these queries by reporting their group membership state (and their desired set of sources) with Current-State Group Records in IGMPv3 Membership Reports.

As a member of a multicast group, a system may express interest in receiving or not receiving traffic from particular sources. As the desired reception state of a system changes, it reports these changes using Filter-Mode-Change Records or Source-List-Change Records. These records indicate an explicit state change in a group at a system in either the group record's source list or its filter-mode. When a group membership is terminated at a system or traffic from a particular source is no longer desired, a multicast router must query for other members of the group or listeners of the source before deleting the group (or source) and pruning its traffic.

To enable all systems on a network to respond to changes in group membership, multicast routers send specific queries. A Group-Specific Query is sent to verify there are no systems that desire reception of the specified group or to "rebuild" the desired reception state for a particular group. Group-Specific Queries are sent when a router receives a State-Change record indicating a system is leaving a group.

A Group-and-Source Specific Query is used to verify there are no systems on a network which desire to receive traffic from a set of sources. Group-and-Source Specific Queries list sources for a particular group which have been requested to no longer be forwarded. This query is sent by a multicast router to learn if any systems desire reception of packets to the specified group address from the specified source addresses. Group-and-Source Specific Queries are only sent in response to State-Change Records and never in response to Current-State Records. Section 4.1.11 describes each query in more detail.

6.2. IGMP State Maintained by Multicast Routers

Multicast routers implementing IGMPv3 keep state per group per attached network. This group state consists of a filter-mode, a list of sources, and various timers. For each attached network running IGMP, a multicast router records the desired reception state for that network. That state conceptually consists of a set of records of the form:

(multicast address, group timer, filter-mode, (source records))

Each source record is of the form:

(source address, source timer)

If all sources within a given group are desired, an empty source record list is kept with filter-mode set to EXCLUDE. This means hosts on this network want all sources for this group to be forwarded. This is the IGMPv3 equivalent to a IGMPv1 or IGMPv2 group join.

6.2.1. Definition of Router Filter-Mode

To reduce internal state, IGMPv3 routers keep a filter-mode per group per attached network. This filter-mode is used to condense the total desired reception state of a group to a minimum set such that all systems' memberships are satisfied. This filter-mode may change in response to the reception of particular types of group records or when certain timer conditions occur. In the following sections, we use the term "router filter-mode" to refer to the filter-mode of a particular group within a router. Section 6.4 describes the changes of a router filter-mode per group record received.

Conceptually, when a group record is received, the router filter-mode for that group is updated to cover all the requested sources using the least amount of state. As a rule, once a group record with a filter-mode of EXCLUDE is received, the router filter-mode for that group will be EXCLUDE.

When a router filter-mode for a group is EXCLUDE, the source record list contains two types of sources. The first type is the set which represents conflicts in the desired reception state; this set must be forwarded by some router on the network. The second type is the set of sources which hosts have requested to not be forwarded. Appendix A describes the reasons for keeping two different sets when in EXCLUDE mode.

When a router filter-mode for a group is INCLUDE, the source record list is the list of sources desired for the group. This is the total desired set of sources for that group. Each source in the source record list must be forwarded by some router on the network.

Because a reported group record with a filter-mode of EXCLUDE will cause a router to transition its filter-mode for that group to EXCLUDE, a mechanism for transitioning a router's filter-mode back to INCLUDE must exist. If all systems with a group record in EXCLUDE filter-mode cease reporting, it is desirable for the router filter-mode for that group to transition back to INCLUDE mode. This transition occurs when the group timer expires and is explained in detail in Section 6.5.

6.2.2. Definition of Group Timers

The group timer is only used when a group is in EXCLUDE mode and it represents the time for the filter-mode of the group to expire and switch to INCLUDE mode. We define a group timer as a decrementing timer with a lower bound of zero kept per group per attached network. Group timers are updated according to the types of group records received.

A group timer expiring when a router filter-mode for the group is EXCLUDE means there are no listeners on the attached network in EXCLUDE mode. At this point, a router will transition to INCLUDE filter-mode. Section 6.5 describes the actions taken when a group timer expires while in EXCLUDE mode.

The following table summarizes the role of the group timer. Section Section 6.4 describes the details of setting the group timer per type of group record received.

Group Filter-Mode	Group Timer Value	Actions/Comments
INCLUDE	Timer >= 0	All members in INCLUDE mode.
EXCLUDE	Timer > 0	At least one member in EXCLUDE mode.
EXCLUDE	Timer == 0	No more listeners to group. If all source timers have expired then delete Group Record. If there are still source record timers running, switch to INCLUDE filter-mode using those source records with running timers as the INCLUDE source record state.

Table 6

6.2.3. Definition of Source Timers

A source timer is kept per source record and is a decrementing timer with a lower bound of zero. Source timers are updated according to the type and filter-mode of the group record received. Source timers are always updated (for a particular group) whenever the source is present in a received record for that group. Section 6.4 describes the setting of source timers per type of group records received.

A source record with a running timer with a router filter-mode for the group of INCLUDE means that there is currently one or more systems (in INCLUDE filter-mode) which desire to receive that source. If a source timer expires with a router filter-mode for the group of INCLUDE, the router concludes that traffic from this particular source is no longer desired on the attached network, and deletes the associated source record.

Source timers are treated differently when a router filter-mode for a group is EXCLUDE. If a source record has a running timer with a router filter-mode for the group of EXCLUDE, it means that at least one system desires the source. It should therefore be forwarded by a router on the network. Appendix A describes the reasons for keeping state for sources that have been requested to be forwarded while in EXCLUDE state.

If a source timer expires with a router filter-mode for the group of EXCLUDE, the router informs the routing protocol that there is no longer a receiver on the network interested in traffic from this source.

When a router filter-mode for a group is EXCLUDE, source records are only deleted when the group timer expires. Section 6.3 describes the actions that should be taken dependent upon the value of a source timer.

6.3. IGMPv3 Source-Specific Forwarding Rules

When a multicast router receives a datagram from a source destined to a particular group, a decision has to be made whether to forward the datagram onto an attached network or not. The multicast routing protocol in use is in charge of this decision, and should use the IGMPv3 information to ensure that all sources/groups desired on a subnetwork are forwarded to that subnetwork. IGMPv3 information does not override multicast routing information; for example, if the IGMPv3 filter-mode group for G is EXCLUDE, a router may still forward packets for excluded sources to a transit subnet.

To summarize, the following table describes the forwarding suggestions made by IGMP to the routing protocol for traffic originating from a source destined to a group. It also summarizes the actions taken upon the expiration of a source timer based on the router filter-mode of the group.

Group Filter-Mode	Group Timer Value	Action
INCLUDE	TIMER > 0	Suggest to forward traffic from source
INCLUDE	TIMER == 0	Suggest to stop forwarding traffic from source and remove source record. If there are no more source records for the group, delete group record.
INCLUDE	No Source Elements	Suggest to not forward source
EXCLUDE	TIMER > 0	Suggest to forward traffic from source
EXCLUDE	TIMER == 0	Suggest to not forward traffic from source (DO NOT remove record)
EXCLUDE	No Source Elements	Suggest to forward traffic from source

Table 7

6.4. Action on Reception of Reports

SSM-aware routers SHOULD ignore records that contain multicast addresses in the SSM address range if the record type is MODE_IS_EXCLUDE or CHANGE_TO_EXCLUDE_MODE. SSM-aware routers SHOULD ignore IGMPv1/IGMPv2 Report and IGMPv2 DONE messages that contain multicast addresses in the SSM address range, SHOULD NOT use such Reports to establish IP forwarding state, and MAY log an error if it receives such a message.

6.4.1. Reception of Current-State Records

When receiving Current-State Records, a router updates both its group and source timers. In some circumstances, the reception of a type of group record will cause the router filter-mode for that group to change. The table below describes the actions, with respect to state and timers that occur to a router's state upon reception of Current-State Records.

The following notation is used to describe the updating of source timers. The notation (A, B) will be used to represent the total number of sources for a particular group, where

A = set of source records whose source timers > 0 (Sources that at least one host has requested to be forwarded)
B = set of source records whose source timers = 0 (Sources that IGMP will suggest to the routing protocol not to forward)

Note that there will only be two sets when a router's filter-mode for a group is EXCLUDE. When a router's filter-mode for a group is INCLUDE, a single set is used to describe the set of sources requested to be forwarded (e.g., simply (A)).

In the following tables, abbreviations are used for several variables (all of which are described in detail in Section 8). The variable GMI is an abbreviation for the Group Membership Interval, which is the time in which group memberships will time out. The variable LMQT is an abbreviation for the Last Member Query Time, which is the total time spent after Last Member Query Count retransmissions. LMQT represents the "leave latency", or the difference between the transmission of a membership change and the change in the information given to the routing protocol.

Within the "Actions" section of the router state tables, we use the notation 'A=J', which means that the set A of source records should have their source timers set to value J. 'Delete A' means that the set A of source records should be deleted. 'Group Timer=J' means that the Group Timer for the group should be set to value J.

Router State	Report Rec'd	New Router State	Actions
INCLUDE (A)	IS_IN (B)	INCLUDE (A+B)	(B)=GMI
INCLUDE (A)	IS_EX (B)	EXCLUDE (A*B,B-A)	(B-A)=0 Delete (A-B) Group Timer=GMI
EXCLUDE (X,Y)	IS_IN (A)	EXCLUDE (X+A,Y-A)	(A)=GMI
EXCLUDE (X,Y)	IS_EX (A)	EXCLUDE (A-Y,Y*A)	(A-X-Y)=GMI Delete (X-A) Delete (Y-A) Group Timer=GMI

6.4.2. Reception of Filter-Mode-Change and Source-List-Change Records

When a change in the global state of a group occurs in a system, the system sends either a Source-List-Change Record or a Filter-Mode-Change Record for that group. As with Current-State Records, routers must act upon these records and possibly change their own state to reflect the new desired membership state of the network.

Routers must query sources that are requested to be no longer forwarded to a group. When a router queries or receives a query for a specific set of sources, it lowers its source timers for those sources to a small interval of Last Member Query Time seconds. If group records are received in response to the queries which express interest in receiving traffic from the queried sources, the corresponding timers are updated.

Similarly, when a router queries a specific group, it lowers its group timer for that group to a small interval of Last Member Query Time seconds. If any group records expressing EXCLUDE mode interest in the group are received within the interval, the group timer for the group is updated and the suggestion to the routing protocol to forward the group stands without any interruption.

During a query period (i.e., Last Member Query Time seconds), the IGMP component in the router continues to suggest to the routing protocol that it forwards traffic from the groups or sources that it is querying. It is not until after Last Member Query Time seconds without receiving a record expressing interest in the queried group or sources that the router may prune the group or sources from the network.

The following table describes the changes in group state and the action(s) taken when receiving either Filter-Mode-Change or Source-List-Change Records. This table also describes the queries which are sent by the querier when a particular report is received.

We use the following notation for describing the queries which are sent. We use the notation 'Q(G)' to describe a Group-Specific Query to G. We use the notation 'Q(G,A)' to describe a Group-and-Source Specific Query to G with source-list A. If source-list A is null as a result of the action (e.g., A*B) then no query is sent as a result of the operation.

In order to maintain protocol robustness, queries sent by actions in the table below need to be transmitted [Last Member Query Count] times, once every [Last Member Query Interval].

If while scheduling new queries, there are already pending queries to be retransmitted for the same group, the new and pending queries have to be merged. In addition, received host reports for a group with pending queries may affect the contents of those queries. Section Section 6.6.3 describes the process of building and maintaining the state of pending queries.

Router State	Report Rec'd	New Router State	Actions
INCLUDE (A)	ALLOW (B)	INCLUDE (A+B)	(B)=GMI
INCLUDE (A)	BLOCK (B)	INCLUDE (A)	Send Q(G,A*B)
INCLUDE (A)	TO_EX (B)	EXCLUDE (A*B,B-A)	(B-A)=0 Delete (A-B) Send Q(G,A*B) Group Timer=GMI
INCLUDE (A)	TO_IN (B)	INCLUDE (A+B)	(B)=GMI Send Q(G,A-B)
EXCLUDE (X,Y)	ALLOW (A)	EXCLUDE (X+A,Y-A)	(A)=GMI
EXCLUDE (X,Y)	BLOCK (A)	EXCLUDE (X+(A-Y),Y)	(A-X-Y)=Group Timer Send Q(G,A-Y)
EXCLUDE (X,Y)	TO_EX (A)	EXCLUDE (A-Y,Y*A)	(A-X-Y)=Group Timer Delete (X-A) Delete (Y-A) Send Q(G,A-Y) Group Timer=GMI
EXCLUDE (X,Y)	TO_IN (A)	EXCLUDE (X+A,Y-A)	(A)=GMI Send Q(G,X-A) Send Q(G)

6.5. Switching Router Filter-Modes

The group timer is used as a mechanism for transitioning the router filter-mode from EXCLUDE to INCLUDE.

When a group timer expires with a router filter-mode of EXCLUDE, a router assumes that there are no systems with a filter-mode of EXCLUDE present on the attached network. When a router's filter-mode for a group is EXCLUDE and the group timer expires, the router filter-mode for the group transitions to INCLUDE.

A router uses source records with running source timers as its state for the switch to a filter-mode of INCLUDE. If there are any source records with source timers greater than zero (i.e., requested to be forwarded), a router switches to filter-mode of INCLUDE using those source records. Source records whose timers are zero (from the previous EXCLUDE mode) are deleted.

For example, if a router's state for a group is EXCLUDE(X,Y) and the group timer expires for that group, the router switches to filter-mode of INCLUDE with state INCLUDE(X).

6.6. Action on Reception of Queries

6.6.1. Timer Updates

When a router sends or receives a query with a clear Suppress Router-Side Processing flag, it must update its timers to reflect the correct timeout values for the group or sources being queried. The following table describes the timer actions when sending or receiving a Group-Specific or Group-and-Source Specific Query with the Suppress Router-Side Processing flag not set.

Query	Action
Q(G,A)	Source Timer for sources in A are lowered to LMQT
Q(G)	Group Timer is lowered to LMQT

Table 8

When a router sends or receives a query with the Suppress Router-Side Processing flag set, it will not update its timers.

6.6.2. Querier Election

IGMPv3 elects a single querier per subnet using the same querier election mechanism as IGMPv2, namely by IP address. When a router receives a general query with a lower IP address, it sets the Other-Querier- Present timer to Other Querier Present Interval and ceases to send general queries on the network if it was the previously elected querier. After its Other-Querier Present timer expires, it should begin sending General Queries.

If a router receives an older version general query, it MUST use the oldest version of IGMP on the network. For a detailed description of compatibility issues between IGMP versions see section Section 7.

6.6.3. Building and Sending Specific Queries

6.6.3.1. Building and Sending Group Specific Queries

When a table action "Send Q(G)" is encountered, then the group timer must be lowered to LMQT. The router must then immediately send a group specific query as well as schedule [Last Member Query Count - 1] query retransmissions to be sent every [Last Member Query Interval] over [Last Member Query Time].

When transmitting a group specific query, if the group timer is larger than LMQT, the "Suppress Router-Side Processing" bit is set in the query message.

6.6.3.2. Building and Sending Group and Source Specific Queries

When a table action "Send Q(G,X)" is encountered by a querier in the table in Section 6.4.2, the following actions must be performed for each of the sources in X of group G, with source timer larger than LMQT:

* Set number of retransmissions for each source to [Last Member Query Count].

* Lower source timer to LMQT.

The router must then immediately send a group and source specific query as well as schedule [Last Member Query Count - 1] query retransmissions to be sent every [Last Member Query Interval] over [Last Member Query Time]. The contents of these queries are calculated as follows.

When building a group and source specific query for a group G, two separate query messages are sent for the group. The first one has the "Suppress Router-Side Processing" bit set and contains all the sources with retransmission state and timers greater than LMQT. The second has the "Suppress Router-Side Processing" bit clear and contains all the sources with retransmission state and timers lower or equal to LMQT. If either of the two calculated messages does not contain any sources, then its transmission is suppressed.

Note: If a group specific query is scheduled to be transmitted at the same time as a group and source specific query for the same group, then transmission of the group and source specific message with the "Suppress Router-Side Processing" bit set may be suppressed.

7. Interoperation With Older Versions of IGMP

IGMP version 3 hosts and routers interoperate with hosts and routers that have not yet been upgraded to IGMPv3. This compatibility is maintained by hosts and routers taking appropriate actions depending on the versions of IGMP operating on hosts and routers within a network.

7.1. Query Version Distinctions

The IGMP version of a Membership Query message is determined as follows:

IGMPv1 Query: length = 8 octets AND Max Resp Code field is zero

IGMPv2 Query: length = 8 octets AND Max Resp Code field is non-zero

IGMPv3 Query: length >= 12 octets

Query messages that do not match any of the above conditions (e.g., a Query of length 10 octets) MUST be silently ignored.

7.2. Group Member Behavior

7.2.1. In the Presence of Older Version Queriers

In order to be compatible with older version routers, IGMPv3 hosts MUST operate in version 1 and version 2 compatibility modes. IGMPv3 hosts MUST keep state per local interface regarding the compatibility mode of each attached network. A host's compatibility mode is determined from the Host Compatibility Mode variable which can be in one of three states: IGMPv1, IGMPv2 or IGMPv3. This variable is kept per interface and is dependent on the version of General Queries heard on that interface as well as the Older Version Querier Present timers for the interface.

In order to switch gracefully between versions of IGMP, hosts keep both an IGMPv1 Querier Present timer and an IGMPv2 Querier Present timer per interface. IGMPv1 Querier Present is set to Older Version Querier Present Timeout seconds whenever an IGMPv1 Membership Query is received. IGMPv2 Querier Present is set to Older Version Querier Present Timeout seconds whenever an IGMPv2 General Query is received.

The Host Compatibility Mode of an interface changes whenever an older version query (than the current compatibility mode) is heard or when certain timer conditions occur. When the IGMPv1 Querier Present timer expires, a host switches to Host Compatibility mode of IGMPv2

if it has a running IGMPv2 Querier Present timer. If it does not have a running IGMPv2 Querier Present timer then it switches to Host Compatibility of IGMPv3. When the IGMPv2 Querier Present timer expires, a host switches to Host Compatibility mode of IGMPv3.

The Host Compatibility Mode variable is based on whether an older version General query was heard in the last Older Version Querier Present Timeout seconds. The Host Compatibility Mode is set depending on the following:

Host Compatibility Mode	Timer State
IGMPv3 (default)	IGMPv2 Querier Present not running and IGMPv1 Querier Present not running
IGMPv2	IGMPv2 Querier Present running and IGMPv1 Querier Present not running
IGMPv1	IGMPv1 Querier Present running

Table 9

If a host receives a query which causes its Querier Present timers to be updated and correspondingly its compatibility mode, it should switch compatibility modes immediately.

When Host Compatibility Mode is IGMPv3, a host acts using the IGMPv3 protocol on that interface. When Host Compatibility Mode is IGMPv2, a host acts in IGMPv2 compatibility mode, using only the IGMPv2 protocol, on that interface. When Host Compatibility Mode is IGMPv1, a host acts in IGMPv1 compatibility mode, using only the IGMPv1 protocol on that interface.

An IGMPv1 router will send General Queries with the Max Resp Code set to 0. This MUST be interpreted as a value of 100 (10 seconds).

An IGMPv2 router will send General Queries with the Max Resp Code set to the desired Max Resp Time, i.e., the full range of this field is linear and the exponential algorithm described in Section 4.1.1 is not used.

Whenever a host changes its compatibility mode, it cancels all its pending response and retransmission timers.

An SSM-aware host that receives an IGMPv1 Query, an IGMPv2 General Query, or an IGMPv2 Group Specific Query for a multicast address in the SSM address range SHOULD log an error. It is RECOMMENDED that implementions provide a configuration option to disable use of Host Compatibility Mode to allow networks to operate only in SSM mode. This configuration option SHOULD be disabled by default.

7.2.2. In the Presence of Older Version Group Members

An IGMPv3 host may be placed on a network where there are hosts that have not yet been upgraded to IGMPv3. A host MAY allow its IGMPv3 Membership Record to be suppressed by either a Version 1 Membership Report, or a Version 2 Membership Report. SSM-aware hosts MUST NOT allow its IGMPv3 Membership Record to be suppressed.

7.3. Multicast Router Behavior

7.3.1. In the Presence of Older Version Queriers

IGMPv3 routers may be placed on a network where at least one router on the network has not yet been upgraded to IGMPv3. The following requirements apply:

* If any older versions of IGMP are present on routers, the querier MUST use the lowest version of IGMP present on the network. This must be administratively assured; routers that desire to be compatible with IGMPv1 and IGMPv2 MUST have a configuration option to act in IGMPv1 or IGMPv2 compatibility modes. When in IGMPv1 mode, routers MUST send Periodic Queries with a Max Resp Code of 0 and truncated at the Group Address field (i.e., 8 bytes long), and MUST ignore Leave Group messages. They SHOULD also warn about receiving an IGMPv2 or IGMPv3 query, although such warnings MUST be rate-limited. When in IGMPv2 mode, routers MUST send Periodic Queries truncated at the Group Address field (i.e., 8 bytes long), and SHOULD also warn about receiving an IGMPv3 query (such warnings MUST be rate-limited). They also MUST fill in the Max Resp Time in the Max Resp Code field, i.e., the exponential algorithm described in Section 4.1.1 is not used.

* If a router is not explicitly configured to use IGMPv1 or IGMPv2 and hears an IGMPv1 Query or IGMPv2 General Query, it SHOULD log a warning. These warnings MUST be rate-limited.

* It is RECOMMENDED that implementions provide a configuration option to disable use of compatibility mode to allow networks to operate only in SSM mode. This configuration option SHOULD be disabled by default.

7.3.2. In the Presence of Older Version Group Members

IGMPv3 routers may be placed on a network where there are hosts that have not yet been upgraded to IGMPv3. In order to be compatible with older version hosts, IGMPv3 routers MUST operate in version 1 and version 2 compatibility modes. IGMPv3 routers keep a compatibility mode per group record. A group's compatibility mode is determined from the Group Compatibility Mode variable which can be in one of three states: IGMPv1, IGMPv2 or IGMPv3. This variable is kept per group record and is dependent on the version of Membership Reports heard for that group as well as the Older Version Host Present timer for the group.

In order to switch gracefully between versions of IGMP, routers keep an IGMPv1 Host Present timer and an IGMPv2 Host Present timer per group record. The IGMPv1 Host Present timer is set to Older Version Host Present Timeout seconds whenever an IGMPv1 Membership Report is received. The IGMPv2 Host Present timer is set to Older Version Host Present Timeout seconds whenever an IGMPv2 Membership Report is received.

The Group Compatibility Mode of a group record changes whenever an older version report (than the current compatibility mode) is heard or when certain timer conditions occur. When the IGMPv1 Host Present timer expires, a router switches to Group Compatibility mode of IGMPv2 if it has a running IGMPv2 Host Present timer. If it does not have a running IGMPv2 Host Present timer then it switches to Group Compatibility of IGMPv3. When the IGMPv2 Host Present timer expires and the IGMPv1 Host Present timer is not running, a router switches to Group Compatibility mode of IGMPv3. Note that when a group switches back to IGMPv3 mode, it takes some time to regain source-specific state information. Source-specific information will be learned during the next General Query, but sources that should be blocked will not be blocked until [Group Membership Interval] after that.

The Group Compatibility Mode variable is based on whether an older version report was heard in the last Older Version Host Present Timeout seconds. The Group Compatibility Mode is set depending on the following:

Group Compatibility Mode	Timer State
IGMPv3 (default)	IGMPv2 Host Present not running and IGMPv1 Host Present not running
IGMPv2	IGMPv2 Host Present running and IGMPv1 Host Present not running
IGMPv1	IGMPv1 Host Present running

Table 10

If a router receives a report which causes its older Host Present timers to be updated and correspondingly its compatibility mode, it SHOULD switch compatibility modes immediately.

When Group Compatibility Mode is IGMPv3, a router acts using the IGMPv3 protocol for that group.

When Group Compatibility Mode is IGMPv2, a router internally translates the following IGMPv2 messages for that group to their IGMPv3 equivalents:

IGMPv2 Message	IGMPv3 Equivalent
Report	IS_EX({})
Leave	TO_IN({})

Table 11

IGMPv3 BLOCK messages are ignored, as are source-lists in TO_EX() messages (i.e., any TO_EX() message is treated as TO_EX({})).

When Group Compatibility Mode is IGMPv1, a router internally translates the following IGMPv1 and IGMPv2 messages for that group to their IGMPv3 equivalents:

IGMPv2 Message	IGMPv3 Equivalent
v1 Report	IS_EX({})
v2 Report	IS_EX({})

Table 12

In addition to ignoring IGMPv3 BLOCK messages and source-lists in TO_EX() messages as in IGMPv2 Group Compatibility Mode, IGMPv2 Leave messages and IGMPv3 TO_IN() messages are also ignored.

8. List of Timers, Counters and Their Default Values

Most of these timers are configurable. If non-default settings are used, they MUST be consistent among all systems on a single link. Note that parentheses are used to group expressions to make the algebra clear.

8.1. Robustness Variable

The Robustness Variable allows tuning for the expected packet loss on a network. If a network is expected to be lossy, the Robustness Variable may be increased. IGMP is robust to (Robustness Variable - 1) packet losses. The Robustness Variable MUST NOT be zero, and SHOULD NOT be one. Default: 2

8.2. Query Interval

The Query Interval is the interval between General Queries sent by the Querier. Default: 125 seconds.

By varying the [Query Interval], an administrator may tune the number of IGMP messages on the network; larger values cause IGMP Queries to be sent less often.

8.3. Query Response Interval

The Max Response Time used to calculate the Max Resp Code inserted into the periodic General Queries. Default: 100 (10 seconds)

By varying the [Query Response Interval], an administrator may tune the burstiness of IGMP messages on the network; larger values make the traffic less bursty, as host responses are spread out over a larger interval. The number of seconds represented by the [Query Response Interval] must be less than the [Query Interval].

8.4. Group Membership Interval

The Group Membership Interval is the amount of time that must pass before a multicast router decides there are no more members of a group or a particular source on a network.

This value MUST be ((the Robustness Variable) times (the Query Interval)) plus (2 * Query Response Interval).

8.5. Other Querier Present Interval

The Other Querier Present Interval is the length of time that must pass before a multicast router decides that there is no longer another multicast router which should be the querier. This value MUST be ((the Robustness Variable) times (the Query Interval)) plus (one half of one Query Response Interval).

8.6. Startup Query Interval

The Startup Query Interval is the interval between General Queries sent by a Querier on startup. Default: 1/4 the Query Interval.

8.7. Startup Query Count

The Startup Query Count is the number of Queries sent out on startup, separated by the Startup Query Interval. Default: the Robustness Variable.

8.8. Last Member Query Interval

The Last Member Query Interval is the Max Response Time used to calculate the Max Resp Code inserted into Group-Specific Queries sent in response to Leave Group messages. It is also the Max Response Time used in calculating the Max Resp Code for Group-and-Source-Specific Query messages. Default: 10 (1 second)

Note that for values of LMQI greater than 12.8 seconds, a limited set of values can be represented, corresponding to sequential values of Max Resp Code. When converting a configured time to a Max Resp Code value, it is recommended to use the exact value if possible, or the next lower value if the requested value is not exactly representable.

This value may be tuned to modify the "leave latency" of the network. A reduced value results in reduced time to detect the loss of the last member of a group or source.

8.9. Last Member Query Count

The Last Member Query Count is the number of Group-Specific Queries sent before the router assumes there are no local members. The Last Member Query Count is also the number of Group-and-Source-Specific Queries sent before the router assumes there are no listeners for a particular source. Default: the Robustness Variable.

8.10. Last Member Query Time

The Last Member Query Time is the time value represented by the Last Member Query Interval, multiplied by the Last Member Query Count. It is not a tunable value, but may be tuned by changing its components.

8.11. Unsolicited Report Interval

The Unsolicited Report Interval is the time between repetitions of a host's initial report of membership in a group. Default: 1 second.

8.12. Older Version Querier Present Interval

The Older Version Querier Present Interval is the timeout for transitioning a host back to IGMPv3 mode once an older version query is heard. When an older version query is received, hosts set their Older Version Querier Present Timer to Older Version Querier Present Interval.

It is RECOMMENDED to use the default values for calculating the interval value as hosts do not know the values configured on the querying routers. This value SHOULD be [Robustness Variable] times [Query Interval] plus (10 times the Max Resp Time in the last received query message).

8.13. Older Host Present Interval

The Older Host Present Interval is the time-out for transitioning a group back to IGMPv3 mode once an older version report is sent for that group. When an older version report is received, routers set their Older Host Present Timer to Older Host Present Interval.

This value MUST be ((the Robustness Variable) times (the Query Interval)) plus (one Query Response Interval).

8.14. Configuring Timers

This section is meant to provide advice to network administrators on how to tune these settings to their network. Ambitious router implementations might tune these settings dynamically based upon changing characteristics of the network.

8.14.1. Robustness Variable

The Robustness Variable tunes IGMP to expected losses on a link. IGMPv3 is robust to (Robustness Variable - 1) packet losses, e.g., if the Robustness Variable is set to the default value of 2, IGMPv3 is robust to a single packet loss but may operate imperfectly if more losses occur. On lossy subnetworks, the Robustness Variable should be increased to allow for the expected level of packet loss. However, increasing the Robustness Variable increases the leave latency of the subnetwork. (The leave latency is the time between when the last member stops listening to a source or group and when the traffic stops flowing.)

8.14.2. Query Interval

The overall level of periodic IGMP traffic is inversely proportional to the Query Interval. A longer Query Interval results in a lower overall level of IGMP traffic. The Query Interval MUST be equal to or longer than the Max Response Time inserted in General Query messages.

8.14.3. Max Response Time

The burstiness of IGMP traffic is inversely proportional to the Max Response Time. A longer Max Response Time will spread Report messages over a longer interval. However, a longer Max Response Time in Group-Specific and Source-and-Group-Specific Queries extends the leave latency. (The leave latency is the time between when the last member stops listening to a source or group and when the traffic stops flowing.) The expected rate of Report messages can be calculated by dividing the expected number of Reporters by the Max Response Time. The Max Response Time may be dynamically calculated per Query by using the expected number of Reporters for that Query as follows:

Query Type	Expected number of Reporters
General Query	All systems on subnetwork
Group-Specific Query	All systems that had expressed interest in the group on the subnetwork
Source-and-Group-Specific Query	All systems on the subnetwork that had expressed interest in the source and group

Table 13

A router is not required to calculate these populations or tune the Max Response Time dynamically; these are simply guidelines.

9. Security Considerations

We consider the ramifications of a forged message of each type, and describe the usage of IPSEC AH to authenticate messages if desired.

9.1. Query Message

A forged Query message from a machine with a lower IP address than the current Querier will cause Querier duties to be assigned to the forger. If the forger then sends no more Query messages, other routers' Other Querier Present timer will time out and one will resume the role of Querier. During this time, if the forger ignores Leave Messages, traffic might flow to groups with no members for up to [Group Membership Interval].

A DoS attack on a host could be staged through forged Group-and-Source-Specific Queries. The attacker can find out about membership of a specific host with a general query. After that it could send a large number of Group-and-Source-Specific queries, each with a large source list and the Maximum Response Time set to a large value. The host will have to store and maintain the sources specified in all of those queries for as long as it takes to send the delayed response. This would consume both memory and CPU cycles in order to augment the recorded sources with the source lists included in the successive queries.

To protect against such a DoS attack, a host stack implementation could restrict the number of Group-and-Source-Specific Queries per group membership within this interval, and/or record only a limited number of sources.

Forged Query messages from the local network can be easily traced. There are three measures necessary to defend against externally forged Queries:

* Routers SHOULD NOT forward Queries. This is easier for a router to accomplish if the Query carries the Router-Alert option.

* Hosts SHOULD ignore v2 or v3 Queries without the Router-Alert option.

* Hosts SHOULD ignore v1, v2 or v3 General Queries sent to a multicast address other than 224.0.0.1, the all-systems address.

9.2. Current-State Report messages

A forged Report message may cause multicast routers to think there are members of a group on a network when there are not. Forged Report messages from the local network are meaningless, since joining a group on a host is generally an unprivileged operation, so a local user may trivially gain the same result without forging any messages. Forged Report messages from external sources are more troublesome; there are two defenses against externally forged Reports:

* Ignore the Report if you cannot identify the source address of the packet as belonging to a network assigned to the interface on which the packet was received. This solution means that Reports sent by mobile hosts without addresses on the local network will be ignored. Report messages with a source address of 0.0.0.0 SHOULD be accepted on any interface.

* Ignore Report messages without Router Alert options [RFC2113], and require that routers not forward Report messages. (The requirement is not a requirement of generalized filtering in the forwarding path, since the packets already have Router Alert options in them.) This solution breaks backwards compatibility with implementations of IGMPv1 or earlier versions of IGMPv2 which did not require Router Alert.

A forged Version 1 Report Message may put a router into "version 1 members present" state for a particular group, meaning that the router will ignore Leave messages. This can cause traffic to flow to groups with no members for up to [Group Membership Interval]. This can be solved by providing routers with a configuration switch to

ignore Version 1 messages completely. This breaks automatic compatibility with Version 1 hosts, so should only be used in situations where "fast leave" is critical.

A forged Version 2 Report Message may put a router into "version 2 members present" state for a particular group, meaning that the router will ignore IGMPv3 source-specific state messages. This can cause traffic to flow from unwanted sources for up to [Group Membership Interval]. This can be solved by providing routers with a configuration switch to ignore Version 2 messages completely. This breaks automatic compatibility with Version 2 hosts, so should only be used in situations where source include and exclude is critical.

9.3. State-Change Report Messages

A forged State-Change Report message will cause the Querier to send out Group-Specific or Source-and-Group-Specific Queries for the group in question. This causes extra processing on each router and on each member of the group, but can not cause loss of desired traffic. There are two defenses against externally forged State-Change Report messages:

* Ignore the State-Change Report message if you cannot identify the source address of the packet as belonging to a subnet assigned to the interface on which the packet was received. This solution means that State-Change Report messages sent by mobile hosts without addresses on the local subnet will be ignored. State-Change Report messages with a source address of 0.0.0.0 SHOULD be accepted on any interface.

* Ignore State-Change Report messages without Router Alert options [RFC2113], and require that routers not forward State-Change Report messages. (The requirement is not a requirement of generalized filtering in the forwarding path, since the packets already have Router Alert options in them.)

9.4. IPSEC Usage

In addition to these measures, IPSEC in Authentication Header mode [RFC4302] may be used to protect against remote attacks by ensuring that IGMPv3 messages came from a system on the LAN (or, more specifically, a system with the proper key). When using IPSEC, the messages sent to 224.0.0.1 and 224.0.0.22 should be authenticated using AH. When keying, there are two possibilities:

1. Use a symmetric signature algorithm with a single key for the LAN (or a key for each group). This allows validation that a packet was sent by a system with the key. This has the limitation that

any system with the key can forge a message; it is not possible to authenticate the individual sender precisely. It also requires disabling IPSec's Replay Protection.

2. When appropriate key management standards have been developed, use an asymmetric signature algorithm. All systems need to know the public key of all routers, and all routers need to know the public key of all systems. This requires a large amount of key management but has the advantage that senders can be authenticated individually so e.g., a host cannot forge a message that only routers should be allowed to send.

This solution only directly applies to Query and Leave messages in IGMPv1 and IGMPv2, since Reports are sent to the group being reported and it is not feasible to agree on a key for host-to-router communication for arbitrary multicast groups.

10. IANA Considerations

All IGMP types described in this document are managed via [I-D.ietf-pim-3228bis].

11. Contributors

Brad Cain, Steve Deering, Isidor Kouvelas, Bill Fenner, and Ajit Thyagarajan are the authors of RFC 3376, which forms the bulk of the content contained herein.

Anuj Budhiraja, Toerless Eckert, Olufemi Komolafe and Tim Winters have contributed valuable content to this version of the specification.

12. Acknowledgments

We would like to thank Ran Atkinson, Luis Costa, Toerless Eckert, Dino Farinacci, Serge Fdida, Wilbert de Graaf, Sumit Gupta, Mark Handley, Bob Quinn, Michael Speer, Dave Thaler and Rolland Vida for comments and suggestions on RFC 3376.

Stig Venaas, Hitoshi Asaeda, and Mike McBride have provided valuable feedback on this version of the specification and we thank them for their input.

13. References

13.1. Normative References

[I-D.ietf-pim-3228bis]
Haberman, B., "IANA Considerations for Internet Group Management Protocols", Work in Progress, Internet-Draft, draft-ietf-pim-3228bis-06, 13 June 2024, <https://datatracker.ietf.org/api/v1/doc/document/draft-ietf-pim-3228bis/>.

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <https://www.rfc-editor.org/info/rfc1112>.

[RFC2113] Katz, D., "IP Router Alert Option", RFC 2113, DOI 10.17487/RFC2113, February 1997, <https://www.rfc-editor.org/info/rfc2113>.

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

[RFC2236] Fenner, W., "Internet Group Management Protocol, Version 2", RFC 2236, DOI 10.17487/RFC2236, November 1997, <https://www.rfc-editor.org/info/rfc2236>.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <https://www.rfc-editor.org/info/rfc4302>.

[RFC4604] Holbrook, H., Cain, B., and B. Haberman, "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast", RFC 4604, DOI 10.17487/RFC4604, August 2006, <https://www.rfc-editor.org/info/rfc4604>.

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <https://www.rfc-editor.org/info/rfc4607>.

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

13.2. Informative References

[RFC1071] Braden, R., Borman, D., and C. Partridge, "Computing the Internet checksum", RFC 1071, DOI 10.17487/RFC1071, September 1988, <https://www.rfc-editor.org/info/rfc1071>.

[RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, DOI 10.17487/RFC3376, October 2002, <https://www.rfc-editor.org/info/rfc3376>.

[RFC3569] Bhattacharyya, S., Ed., "An Overview of Source-Specific Multicast (SSM)", RFC 3569, DOI 10.17487/RFC3569, July 2003, <https://www.rfc-editor.org/info/rfc3569>.

[RFC3678] Thaler, D., Fenner, B., and B. Quinn, "Socket Interface Extensions for Multicast Source Filters", RFC 3678, DOI 10.17487/RFC3678, January 2004, <https://www.rfc-editor.org/info/rfc3678>.

Appendix A. Design Rationale

A.1. The Need for State-Change Messages

IGMPv3 specifies two types of Membership Reports: Current-State and State Change. This section describes the rationale for the need for both these types of Reports.

Routers need to distinguish Membership Reports that were sent in response to Queries from those that were sent as a result of a change in interface state. Membership reports that are sent in response to Membership Queries are used mainly to refresh the existing state at the router; they typically do not cause transitions in state at the router. Membership Reports that are sent in response to changes in interface state require the router to take some action in response to the received report (see Section 6.4).

The inability to distinguish between the two types of reports would force a router to treat all Membership Reports as potential changes in state and could result in increased processing at the router as well as an increase in IGMP traffic on the network.

A.2. Host Suppression

In IGMPv1 and IGMPv2, a host would cancel sending a pending membership reports if a similar report was observed from another member on the network. In IGMPv3, this suppression of host membership reports has been removed. The following points explain the reasons behind this decision.

1. Routers may want to track per-host membership status on an interface. This allows routers to implement fast leaves (e.g., for layered multicast congestion control schemes) as well as track membership status for possible accounting purposes.

2. Membership Report suppression does not work well on bridged LANs. Many bridges and Layer2/Layer3 switches that implement IGMP snooping do not forward IGMP messages across LAN segments in order to prevent membership report suppression. Removing membership report suppression eases the job of these IGMP snooping devices.

3. By eliminating membership report suppression, hosts have fewer messages to process; this leads to a simpler state machine implementation.

4. In IGMPv3, a single membership report now bundles multiple multicast group records to decrease the number of packets sent. In comparison, the previous versions of IGMP required that each multicast group be reported in a separate message.

A.3. Switching Router Filter Modes from EXCLUDE to INCLUDE

If there exist hosts in both EXCLUDE and INCLUDE modes for a single multicast group in a network, the router must be in EXCLUDE mode as well (see section 6.2.1). In EXCLUDE mode, a router forwards traffic from all sources unless that source exists in the exclusion source list. If all hosts in EXCLUDE mode cease to exist, it would be desirable for the router to switch back to INCLUDE mode seamlessly without interrupting the flow of traffic to existing receivers.

One of the ways to accomplish this is for routers to keep track of all sources desired by hosts that are in INCLUDE mode even though the router itself is in EXCLUDE mode. If the group timer now expires in EXCLUDE mode, it implies that there are no hosts in EXCLUDE mode on the network (otherwise a membership report from that host would have refreshed the group timer). The router can then switch to INCLUDE mode seamlessly with the list of sources currently being forwarded in its source list.

Appendix B. Summary of Changes from IGMPv2

While the main additional feature of IGMPv3 is the addition of source filtering, the following is a summary of other changes from RFC 2236.

* State is maintained as Group + List-of-Sources, not simply Group as in IGMPv2.

* Interoperability with IGMPv1 and IGMPv2 systems is defined as operations on the IGMPv3 state.

* The IP Service Interface has changed to allow specification of source-lists.

* The Querier includes its Robustness Variable and Query Interval in Query packets to allow synchronization of these variables on non-Queriers.

* The Max Response Time in Query messages has an exponential range, changing the maximum from 25.5 seconds to about 53 minutes, for use on links with huge numbers of systems.

* Hosts retransmit state-change messages for increased robustness.

* Additional data sections are defined to allow later extensions.

* Report packets are sent to 224.0.0.22, to assist layer-2 switches in snooping.

* Report packets can contain multiple group records, to allow reporting of full current state using fewer packets.

* Hosts no longer perform suppression, to simplify implementations and permit explicit membership tracking.

* New Suppress Router-Side Processing (S) flag in Query messages fixes robustness issues which were also present in IGMPv2.

Appendix C. Summary of Changes from RFC 3376

The following is a list of changes made since RFC 3376.

* Modified definition of Older Version Querier Present Interval to address Erratum 4375.

* Modified metadata to fix Obsoletes vs Updates relationship with RFC 2236 per Erratum 1501.

* Updated introductory text to describe Updates relationship with RFC 2236 per Erratum 7339.

* Updated Group Membership Interval definition to address Erratum 6725.

* Updated text for Router Filter-Mode to address Erratum 5562.

* Clarified the use of General Queries in the Querier election process.

Author's Address

Brian Haberman (editor)
Johns Hopkins University Applied Physics Lab
Email: brian@innovationslab.net

Network Working Group
Internet-Draft
Obsoletes: 3810 (if approved)
Updates: 2710 (if approved)
Intended status: Standards Track
Expires: 15 December 2024

B. Haberman, Ed.
JHU APL

13 June 2024

Multicast Listener Discovery Version 2 (MLDv2) for IPv6
draft-ietf-pim-3810bis-11

Abstract

This document updates RFC 2710, and it specifies Version 2 of the Multicast Listener Discovery Protocol (MLDv2). MLD is used by an IPv6 router to discover the presence of multicast listeners on directly attached links, and to discover which multicast addresses are of interest to those neighboring nodes. MLDv2 is designed to be interoperable with MLDv1. MLDv2 adds the ability for a node to report interest in listening to packets with a particular multicast address only from specific source addresses or from all sources except for specific source addresses.

This document obsoletes RFC 3810.

Status of This Memo

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

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

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

This Internet-Draft will expire on 15 December 2024.

Copyright Notice

Table of Contents

1. Introduction

The Multicast Listener Discovery Protocol (MLD) is used by IPv6 routers to discover the presence of multicast listeners (i.e., nodes that wish to receive multicast packets) on their directly attached links, and to discover specifically which multicast addresses are of interest to those neighboring nodes. Note that a multicast router may itself be a listener of one or more multicast addresses; in this case it performs both the "multicast router part" and the "multicast address listener part" of the protocol, to collect the multicast listener information needed by its multicast routing protocol on the one hand, and to inform itself and other neighboring multicast routers of its listening state on the other hand.

This document specifies Version 2 of MLD. The previous version of MLD is specified in [RFC2710]. In this document we will refer to it as MLDv1. MLDv2 is a translation of the IGMPv3 protocol [RFC3376] for IPv6 semantics.

The MLDv2 protocol, when compared to MLDv1, adds support for "source filtering", i.e., the ability for a node to report interest in listening to packets only from specific source addresses, as required to support Source-Specific Multicast [RFC3569], or from *all but* specific source addresses, sent to a particular multicast address. MLDv2 is designed to be interoperable with MLDv1.

This document uses SSM-aware to refer to systems that support Source-Specific Multicast (SSM) as defined in [RFC4607].

This document obsoletes [RFC3810]. Appendix B.2 lists the main changes from [RFC3810].

1.1. Conventions Used in This Document

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

2. Protocol Overview

This section gives a brief description of the protocol operation. The following sections present the protocol details.

MLD is an asymmetric protocol; it specifies separate behaviors for multicast address listeners (i.e., hosts or routers that listen to multicast packets) and multicast routers. The purpose of MLD is to

enable each multicast router to learn, for each of its directly attached links, which multicast addresses and which sources have interested listeners on that link. The information gathered by MLD is provided to whichever multicast routing protocol is used by the router, in order to ensure that multicast packets are delivered to all links where there are listeners interested in such packets.

Multicast routers only need to know that at least one node on an attached link is listening to packets for a particular multicast address, from a particular source; a multicast router is not required to individually keep track of the interests of each neighboring node. (Nevertheless, see Appendix A.2 item 1 for discussion.)

A multicast router performs the router part of the MLDv2 protocol (described in details in Section 7) on each of its directly attached links. If a multicast router has more than one interface connected to the same link, it only needs to operate the protocol on one of those interfaces. The router behavior depends on whether there are several multicast routers on the same subnet, or not. If that is the case, a querier election mechanism (described in Section 7.6.2) is used to elect a single multicast router to be in Querier state. This router is called the Querier. All multicast routers on the subnet listen to the messages sent by multicast address listeners, and maintain the same multicast listening information state, so that they can take over the querier role, should the present Querier fail. Nevertheless, only the Querier sends periodical or triggered query messages on the subnet, as described in Section 7.1.

A multicast address listener performs the listener part of the MLDv2 protocol (described in details in Section 6) on all interfaces on which multicast reception is supported, even if more than one of those interfaces are connected to the same link.

2.1. Building Multicast Listening State on Multicast Address Listeners

Upper-layer protocols and applications that run on a multicast address listener node use specific service interface calls (described in Section 3) to ask the IP layer to enable or disable reception of packets sent to specific multicast addresses. The node keeps Multicast Address Listening state for each socket on which the service interface calls have been invoked (Section 4.1). In addition to this per-socket multicast listening state, a node must also maintain or compute multicast listening state for each of its interfaces (Section 4.2). Conceptually, that state consists of a set of records, with each record containing an IPv6 multicast address, a filter mode, and a source list. The filter mode may be either INCLUDE or EXCLUDE. In INCLUDE mode, reception of packets sent to the specified multicast address is enabled only from the source

addresses listed in the source list. In EXCLUDE mode, reception of packets sent to the given multicast address is enabled from all source addresses except those listed in the source list.

At most one record per multicast address exists for a given interface. This per-interface state is derived from the per-socket state, but may differ from it when different sockets have differing filter modes and/or source lists for the same multicast address and interface. After a multicast packet has been accepted from an interface by the IP layer, its subsequent delivery to the application connected to a particular socket depends on the multicast listening state of that socket (and possibly also on other conditions, such as what transport-layer port the socket is bound to). Note that MLDv2 messages are not subject to source filtering and must always be processed by hosts and routers.

2.2. Exchanging Messages between the Querier and the Listening Nodes

There are three types of MLDv2 query messages: General Queries, Multicast Address Specific Queries, and Multicast Address and Source Specific Queries. The Querier periodically sends General Queries, to learn multicast address listener information from an attached link. These queries are used to build and refresh the Multicast Address Listener state inside all multicast routers on the link.

Nodes respond to these queries by reporting their per-interface Multicast Address Listening state, through Current State Report messages sent to a specific multicast address all MLDv2 routers on the link listen to. On the other hand, if the listening state of a node changes, the node immediately reports these changes through a State Change Report message. The State Change Report contains either Filter Mode Change records, Source List Change records, or records of both types. A detailed description of the report messages is presented in Section 5.2.13.

Both router and listener state changes are mainly triggered by the expiration of a specific timer, or the reception of an MLD message (listener state change can be also triggered by the invocation of a service interface call). Therefore, to enhance protocol robustness, in spite of the possible unreliability of message exchanges, messages are retransmitted several times. Furthermore, timers are set so as to take into account the possible message losses, and to wait for retransmissions.

Periodical General Queries and Current State Reports do not apply this rule, in order not to overload the link; it is assumed that in general these messages do not generate state changes, their main purpose being to refresh existing state. Thus, even if one such message is lost, the corresponding state will be refreshed during the next reporting period.

As opposed to Current State Reports, State Change Reports are retransmitted several times, in order to avoid them being missed by one or more multicast routers. The number of retransmissions depends on the so-called Robustness Variable. This variable allows tuning the protocol according to the expected packet loss on a link. If a link is expected to be lossy (e.g., a wireless connection), the value of the Robustness Variable may be increased. MLD is robust to [Robustness Variable]-1 packet losses. This document recommends a default value of 2 for the Robustness Variable (see Section 9.1).

If more changes to the same per-interface state entry occur before all the retransmissions of the State Change Report for the first change have been completed, each additional change triggers the immediate transmission of a new State Change Report. Section 6.1 shows how the content of this new report is computed. Retransmissions of the new State Change Report will be scheduled as well, in order to ensure that each instance of state change is transmitted at least [Robustness Variable] times.

If a node on a link expresses, through a State Change Report, its desire to no longer listen to a particular multicast address (or source), the Querier must query for other listeners of the multicast address (or source) before deleting the multicast address (or source) from its Multicast Address Listener state and stopping the corresponding traffic. Thus, the Querier sends a Multicast Address Specific Query to verify whether there are nodes still listening to a specified multicast address or not. Similarly, the Querier sends a Multicast Address and Source Specific Query to verify whether, for a specified multicast address, there are nodes still listening to a specific set of sources, or not. Section 5.1.13 describes each query in more detail.

Both Multicast Address Specific Queries and Multicast Address and Source Specific Queries are only sent in response to State Change Reports, never in response to Current State Reports. This distinction between the two types of reports is needed to avoid the router treating all Multicast Listener Reports as potential changes in state. By doing so, the fast leave mechanism of MLDv2, described in more detail in Section 2.2, might not be effective if a State Change Report is lost, and only the following Current State Report is received by the router. Nevertheless, it avoids an increased

processing at the router and it reduces the MLD traffic on the link. More details on the necessity of distinguishing between the two report types can be found in Appendix A.1.

Nodes respond to the above queries through Current State Reports, that contain their per-interface Multicast Address Listening state only for the multicast addresses (or sources) being queried.

As stated earlier, in order to ensure protocol robustness, all the queries, except the periodical General Queries, are retransmitted several times within a given time interval. The number of retransmissions depends on the Robustness Variable. If, while scheduling new queries, there are pending queries to be retransmitted for the same multicast address, the new queries and the pending queries have to be merged. In addition, host reports received for a multicast address with pending queries may affect the contents of those queries. The process of building and maintaining the state of pending queries is presented in Section 7.6.3.

Protocol robustness is also enhanced through the use of the S flag (Suppress Router-Side Processing). As described above, when a Multicast Address Specific or a Multicast Address and Source Specific Query is sent by the Querier, a number of retransmissions of the query are scheduled. In the original (first) query the S flag is clear. When the Querier sends this query, it lowers the timers for the concerned multicast address (or source) to a given value; similarly, any non-querier multicast router that receives the query lowers its timers in the same way. Nevertheless, while waiting for the next scheduled queries to be sent, the Querier may receive a report that updates the timers. The scheduled queries still have to be sent, in order to ensure that a non-querier router keeps its state synchronized with the current Querier (the non-querier router might have missed the first query). Nevertheless, the timers should not be lowered again, as a valid answer was already received. Therefore, in subsequent queries the Querier sets the S flag.

2.3. Building Multicast Address Listener State on Multicast Routers

Multicast routers that implement MLDv2 (whether they are in Querier state or not) keep state per multicast address per attached link. This multicast address listener state consists of a Filter Mode, a Filter Timer, and a Source List, with a timer associated to each source from the list. The Filter Mode is used to summarize the total listening state of a multicast address to a minimum set, such that all nodes' listening states are respected. The Filter Mode may change in response to the reception of particular types of report messages, or when certain timer conditions occur.

A router is in INCLUDE mode for a specific multicast address on a given interface if all the listeners on the link interested in that address are in INCLUDE mode. The router state is represented through the notation INCLUDE (A), where A is a list of sources, called the "Include List". The Include List is the set of sources that one or more listeners on the link have requested to receive. All the sources from the Include List will be forwarded by the router. Any other source that is not in the Include List will be blocked by the router.

A source can be added to the current Include List if a listener in INCLUDE mode sends a Current State or a State Change Report that includes that source. Each source from the Include List is associated with a source timer that is updated whenever a listener in INCLUDE mode sends a report that confirms its interest in that specific source. If the timer of a source from the Include List expires, the source is deleted from the Include List.

Besides this "soft leave" mechanism, there is also a "fast leave" scheme in MLDv2; it is also based on the use of source timers. When a node in INCLUDE mode expresses its desire to stop listening to a specific source, all the multicast routers on the link lower their timers for that source to a given value. The Querier then sends a Multicast Address and Source Specific Query, to verify whether there are other listeners for that source on the link, or not. If a report that includes this source is received before the timer expiration, all the multicast routers on the link update the source timer. If not, the source is deleted from the Include List. The handling of the Include List, according to the received reports, is detailed in Section 7.4.1 and Section 7.4.2.

A router is in EXCLUDE mode for a specific multicast address on a given interface if there is at least one listener in EXCLUDE mode for that address on the link. When the first report is received from such a listener, the router sets the Filter Timer that corresponds to that address. This timer is reset each time an EXCLUDE mode listener confirms its listening state through a Current State Report. The timer is also updated when a listener, formerly in INCLUDE mode, announces its filter mode change through a State Change Report message. If the Filter Timer expires, it means that there are no more listeners in EXCLUDE mode on the link. In this case, the router switches back to INCLUDE mode for that multicast address.

When the router is in EXCLUDE mode, the router state is represented by the notation EXCLUDE (X,Y), where X is called the "Requested List" and Y is called the "Exclude List". All sources, except those from the Exclude List, will be forwarded by the router. The Requested List has no effect on forwarding. Nevertheless, the router has to maintain the Requested List for two reasons:

* To keep track of sources that listeners in INCLUDE mode listen to. This is necessary to assure a seamless transition of the router to INCLUDE mode, when there is no listener in EXCLUDE mode left. This transition should not interrupt the flow of traffic to listeners in INCLUDE mode for that multicast address. Therefore, at the time of the transition, the Requested List should contain the set of sources that nodes in INCLUDE mode have explicitly requested.

  When the router switches to INCLUDE mode, the sources in the Requested List are moved to the Include List, and the Exclude List is deleted. Before switching, the Requested List can contain an inexact guess of the sources listeners in INCLUDE mode listen to - might be too large or too small. These inexactitudes are due to the fact that the Requested List is also used for fast blocking purposes, as described below. If such a fast blocking is required, some sources may be deleted from the Requested List (as shown in Section 7.4.1 and Section 7.4.2) in order to reduce router state. Nevertheless, in each such case the Filter Timer is updated as well. Therefore, listeners in INCLUDE mode will have enough time, before an eventual switching, to reconfirm their interest in the eliminated source(s), and rebuild the Requested List accordingly. The protocol ensures that when a switch to INCLUDE mode occurs, the Requested List will be accurate. Details about the transition of the router to INCLUDE mode are presented in Appendix A.3.

* To allow the fast blocking of previously unblocked sources. If the router receives a report that contains such a request, the concerned sources are added to the Requested List. Their timers are set to a given small value, and a Multicast Address and Source Specific Query is sent by the Querier, to check whether there are nodes on the link still interested in those sources, or not. If no node announces its interest in receiving those specific source, the timers of those sources expire. Then, the sources are moved from the Requested List to the Exclude List. From then on, the sources will be blocked by the router.

The handling of the EXCLUDE mode router state, according to the received reports, is detailed in Section 7.4.1 and Section 7.4.2.

Both the MLDv2 router and listener behaviors described in this document were defined to ensure backward interoperability with MLDv1 hosts and routers. Interoperability issues are detailed in Section 8.

3. The Service Interface for Requesting IP Multicast Reception

Within an IP system, there is (at least conceptually) a service interface used by upper-layer protocols or application programs to ask the IP layer to enable or disable reception of packets sent to specific IP multicast addresses. In order to take full advantage of the capabilities of MLDv2, a node's IP service interface must support the following operation:

```
IPv6MulticastListen ( socket, interface, IPv6 multicast-address,
                      filter-mode, source-list )
```

where:

* "socket" is an implementation-specific parameter used to distinguish among different requesting entities (e.g., programs, processes) within the node; the socket parameter of BSD Unix system calls is a specific example.

* "interface" is a local identifier of the network interface on which reception of the specified multicast address is to be enabled or disabled. Interfaces may be physical (e.g., an Ethernet interface) or virtual (e.g., the endpoint of a Frame Relay virtual circuit or an IP-in-IP "tunnel"). An implementation may allow a special "unspecified" value to be passed as the interface parameter, in which case the request would apply to the "primary" or "default" interface of the node (perhaps established by system configuration). If reception of the same multicast address is desired on more than one interface, IPv6MulticastListen is invoked separately for each desired interface.

* "IPv6 multicast address" is the multicast address to which the request pertains. If reception of more than one multicast address on a given interface is desired, IPv6MulticastListen is invoked separately for each desired address.

* "filter mode" may be either INCLUDE or EXCLUDE. In INCLUDE mode, reception of packets sent to the specified multicast address is requested only from the source addresses listed in the source list parameter. In EXCLUDE mode, reception of packets sent to the given multicast address is requested from all source addresses except those listed in the source list parameter.

* "source list" is an unordered list of zero or more unicast addresses from which multicast reception is desired or not desired, depending on the filter mode. An implementation MAY impose a limit on the size of source lists. When an operation causes the source list size limit to be exceeded, the service interface SHOULD return an error.

For a given combination of socket, interface, and IPv6 multicast address, only a single filter mode and source list can be in effect at any one time. Nevertheless, either the filter mode or the source list, or both, may be changed by subsequent IPv6MulticastListen requests that specify the same socket, interface, and IPv6 multicast address. Each subsequent request completely replaces any earlier request for the given socket, interface, and multicast address.

The MLDv1 protocol did not support source filters, and had a simpler service interface; it consisted of Start Listening and Stop Listening operations to enable and disable listening to a given multicast address (from all sources) on a given interface. The equivalent operations in the new service interface are as follows:

The Start Listening operation is equivalent to:

```
IPv6MulticastListen ( socket, interface, IPv6 multicast address,
                      EXCLUDE, {} )
```

and the Stop Listening operation is equivalent to:

```
IPv6MulticastListen ( socket, interface, IPv6 multicast address,
                      INCLUDE, {} )
```

where {} is an empty source list.

An example of an API that provides the capabilities outlined in this service interface is given in [RFC3678].

4. Multicast Listening State Maintained by Nodes

4.1. Per-Socket State

For each socket on which IPv6MulticastListen has been invoked, the node records the desired multicast listening state for that socket. That state conceptually consists of a set of records of the form:

```
(interface, IPv6 multicast address, filter mode, source list)
```

The per-socket state evolves in response to each invocation of IPv6MulticastListen on the socket, as follows:

* If the requested filter mode is INCLUDE and the requested source list is empty, then the entry that corresponds to the requested interface and multicast address is deleted, if present. If no such entry is present, the request has no effect.

* If the requested filter mode is EXCLUDE or the requested source list is non-empty, then the entry that corresponds to the requested interface and multicast address, if present, is changed to contain the requested filter mode and source list. If no such entry is present, a new entry is created, using the parameters specified in the request.

4.2. Per-Interface State

In addition to the per-socket multicast listening state, a node must also maintain or compute multicast listening state for each of its interfaces. That state conceptually consists of a set of records of the form:

(IPv6 multicast address, filter mode, source list)

At most one record per multicast address exists for a given interface. This per-interface state is derived from the per-socket state, but may differ from it when different sockets have differing filter modes and/or source lists for the same multicast address and interface. For example, suppose one application or process invokes the following operation on socket s1:

IPv6MulticastListen (s1, i, m, INCLUDE, {a, b, c})

requesting reception on interface i of packets sent to multicast address m, only if they come from the sources a, b, or c. Suppose another application or process invokes the following operation on socket s2:

IPv6MulticastListen (s2, i, m, INCLUDE, {b, c, d})

requesting reception on the same interface i of packets sent to the same multicast address m, only if they come from sources b, c, or d. In order to satisfy the reception requirements of both sockets, it is necessary for interface i to receive packets sent to m from any one of the sources a, b, c, or d. Thus, in this example, the listening state of interface i for multicast address m has filter mode INCLUDE and source list {a, b, c, d}.

After a multicast packet has been accepted from an interface by the IP layer, its subsequent delivery to the application or process that listens on a particular socket depends on the multicast listening

state of that socket (and possibly also on other conditions, such as what transport-layer port the socket is bound to). So, in the above example, if a packet arrives on interface i, destined to multicast address m, with source address a, it may be delivered on socket s1 but not on socket s2. Note that MLDv2 messages are not subject to source filtering and must always be processed by hosts and routers.

Requiring the filtering of packets based upon a socket's multicast reception state is a new feature of this service interface. The previous service interface described no filtering based upon multicast listening state; rather, a Start Listening operation on a socket simply caused the node to start to listen to a multicast address on the given interface; packets sent to that multicast address could be delivered to all sockets, whether they had started to listen or not.

The general rules for deriving the per-interface state from the per-socket state are as follows: for each distinct (interface, IPv6 multicast address) pair that appears in any per-socket state, a per-interface record is created for that multicast address on that interface. Considering all socket records that contain the same (interface, IPv6 multicast address) pair,

* if any such record has a filter mode of EXCLUDE, then the filter mode of the interface record is EXCLUDE, and the source list of the interface record is the intersection of the source lists of all socket records in EXCLUDE mode, minus those source addresses that appear in any socket record in INCLUDE mode. For example, if the socket records for multicast address m on interface i are:

  from socket s1: (i, m, EXCLUDE, {a, b, c, d})

  from socket s2: (i, m, EXCLUDE, {b, c, d, e})

  from socket s3: (i, m, INCLUDE, {d, e, f})

  then the corresponding interface record on interface i is:

  (m, EXCLUDE, {b, c})

  If a fourth socket is added, such as:

  From socket s4: (i, m, EXCLUDE, {})

  then the interface record becomes:

  (m, EXCLUDE, {})

* if all such records have a filter mode of INCLUDE, then the filter mode of the interface record is INCLUDE, and the source list of the interface record is the union of the source lists of all the socket records. For example, if the socket records for multicast address m on interface i are:

  from socket s1: (i, m, INCLUDE, {a, b, c})

  from socket s2: (i, m, INCLUDE, {b, c, d})

  from socket s3: (i, m, INCLUDE, {e, f})

  then the corresponding interface record on interface i is:

  (m, INCLUDE, {a, b, c, d, e, f})

  An implementation MUST NOT use an EXCLUDE interface record for a multicast address if all sockets for this multicast address are in INCLUDE state. If system resource limits are reached when a per-interface state source list is calculated, an error MUST be returned to the application which requested the operation.

The above rules for deriving the per-interface state are (re)evaluated whenever an IPv6MulticastListen invocation modifies the per-socket state by adding, deleting, or modifying a per-socket state record. Note that a change of the per-socket state does not necessarily result in a change of the per-interface state.

5. Message Formats

MLDv2 is a sub-protocol of ICMPv6, that is, MLDv2 message types are a subset of ICMPv6 messages, and MLDv2 messages are identified in IPv6 packets by a preceding Next Header value of 58. All MLDv2 messages described in this document MUST be sent with a link-local IPv6 Source Address, an IPv6 Hop Limit of 1, and an IPv6 Router Alert option [RFC2711] in a Hop-by-Hop Options header. (The Router Alert option is necessary to cause routers to examine MLDv2 messages sent to IPv6 multicast addresses in which the routers themselves have no interest.) MLDv2 Reports can be sent with the source address set to the unspecified address [RFC4291], if a valid link-local IPv6 source address has not been acquired yet for the sending interface. (See Section 5.2.14. for details.)

There are two MLD message types of concern to the MLDv2 protocol described in this document:

* Multicast Listener Query (Type = decimal 130)

* Version 2 Multicast Listener Report (Type = decimal 143). See Section 11 for IANA considerations.

To assure the interoperability with nodes that implement MLDv1 (see Section 8), an implementation of MLDv2 must also support the following two message types:

* Version 1 Multicast Listener Report (Type = decimal 131) [RFC2710]

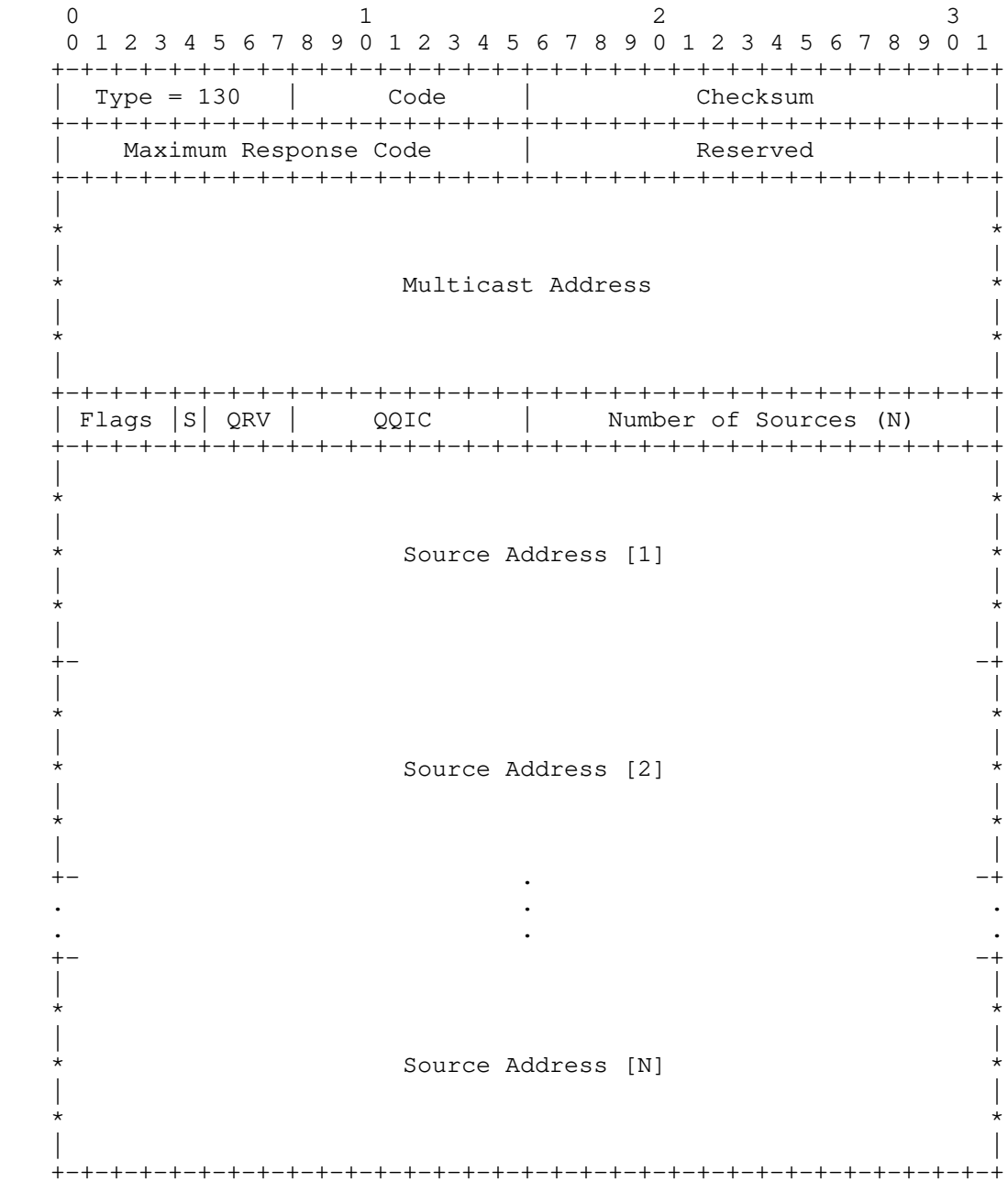
* Version 1 Multicast Listener Done (Type = decimal 132) [RFC2710]

Unrecognized message types MUST be silently ignored. Other message types may be used by newer versions or extensions of MLD, by multicast routing protocols, or for other uses.

In this document, unless otherwise qualified, the capitalized words "Query" and "Report" refer to MLD Multicast Listener Queries and MLD Version 2 Multicast Listener Reports, respectively.

5.1. Multicast Listener Query Message

Multicast Listener Queries are sent by multicast routers in Querier State to query the multicast listening state of neighboring interfaces. Queries have the following format:


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 = 130
Code
Checksum
Maximum Response Code
Reserved
Multicast Address
Flags
S
QRV
QQIC
Number of Sources (N)
Source Address [1]
Source Address [2]
Source Address [N]

5.1.1. Code

Initialized to zero by the sender; ignored by receivers.

5.1.2. Checksum

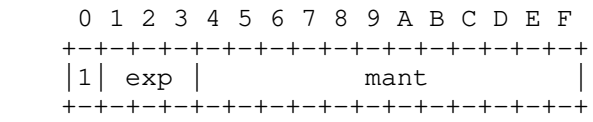
The standard ICMPv6 checksum; it covers the entire MLDv2 message, plus a "pseudo-header" of IPv6 header fields [RFC4443]. For computing the checksum, the Checksum field is set to zero. When a packet is received, the checksum MUST be verified before processing it.

5.1.3. Maximum Response Code

The Maximum Response Code field specifies the maximum time allowed before sending a responding Report. The actual time allowed, called the Maximum Response Delay, is represented in units of milliseconds, and is derived from the Maximum Response Code as follows:

If Maximum Response Code < 32768, Maximum Response Delay = Maximum Response Code

If Maximum Response Code >=32768, Maximum Response Code represents a floating-point value as follows:




```
Maximum Response Delay = (mant | 0x1000) << (exp+3)
```

Small values of Maximum Response Delay allow MLDv2 routers to tune the "leave latency" (the time between the moment the last node on a link ceases to listen to a specific multicast address and the moment the routing protocol is notified that there are no more listeners for that address). Larger values, especially in the exponential range, allow the tuning of the burstiness of MLD traffic on a link.

5.1.4. Reserved

The Reserved field is set to zero on transmission, and ignored on reception.

5.1.5. Multicast Address

For a General Query, the Multicast Address field is set to zero. For a Multicast Address Specific Query or Multicast Address and Source Specific Query, it is set to the multicast address being queried (see Section 5.1.10, below).

5.1.6. Flags

Allocation of individual bits within the Flags field is described in Section 2.2 of [I-D.ietf-pim-3228bis]. Future specifications will define the associated meaning tied to any such allocation.

5.1.7. S Flag (Suppress Router-Side Processing)

When set to one, the S Flag indicates to any receiving multicast routers that they have to suppress the normal timer updates they perform upon hearing a Query. Nevertheless, it does not suppress the querier election or the normal "host-side" processing of a Query that a router may be required to perform as a consequence of itself being a multicast listener.

5.1.8. QRV (Querier's Robustness Variable)

If non-zero, the QRV field contains the [Robustness Variable] value used by the Querier. If the Querier's [Robustness Variable] exceeds 7 (the maximum value of the QRV field), the QRV field is set to zero.

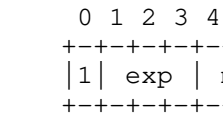
Routers adopt the QRV value from the most recently received Query as their own [Robustness Variable] value, unless that most recently received QRV was zero, in which case they use the default [Robustness Variable] value specified in Section 9.1, or a statically configured value.

5.1.9. QQIC (Querier's Query Interval Code)

The Querier's Query Interval Code field specifies the [Query Interval] used by the Querier. The actual interval, called the Querier's Query Interval (QQI), is represented in units of seconds, and is derived from the Querier's Query Interval Code as follows:

If QQIC < 128, QQI = QQIC

If QQIC >= 128, QQIC represents a floating-point value as follows:



QQI = (mant | 0x10) << (exp + 3)

Multicast routers that are not the current Querier adopt the QQI value from the most recently received Query as their own [Query Interval] value, unless that most recently received QQI was zero, in which case the receiving routers use the default [Query Interval] value specified in Section 9.2.

5.1.10. Number of Sources (N)

The Number of Sources (N) field specifies how many source addresses are present in the Query. This number is zero in a General Query or a Multicast Address Specific Query, and non-zero in a Multicast Address and Source Specific Query. This number is limited by the MTU of the link over which the Query is transmitted. For example, on an Ethernet link with an MTU of 1500 octets, the IPv6 header (40 octets) together with the Hop-By-Hop Extension Header (8 octets) that includes the Router Alert option consume 48 octets; the MLD fields up to the Number of Sources (N) field consume 28 octets; thus, there are 1424 octets left for source addresses, which limits the number of source addresses to 89 (1424/16).

5.1.11. Source Address [i]

The Source Address [i] fields are a vector of n unicast addresses, where n is the value in the Number of Sources (N) field.

5.1.12. Additional Data

If the Payload Length field in the IPv6 header of a received Query indicates that there are additional octets of data present, beyond the fields described here, MLDv2 implementations MUST include those octets in the computation to verify the received MLD Checksum, but MUST otherwise ignore those additional octets. When sending a Query, an MLDv2 implementation MUST NOT include additional octets beyond the fields described above.

5.1.13. Query Variants

There are three variants of the Query message:

* A "General Query" is sent by the Querier to learn which multicast addresses have listeners on an attached link. In a General Query, both the Multicast Address field and the Number of Sources (N) field are zero.

* A "Multicast Address Specific Query" is sent by the Querier to learn if a particular multicast address has any listeners on an attached link. In a Multicast Address Specific Query, the Multicast Address field contains the multicast address of interest, while the Number of Sources (N) field is set to zero.

* A "Multicast Address and Source Specific Query" is sent by the Querier to learn if any of the sources from the specified list for the particular multicast address has any listeners on an attached link or not. In a Multicast Address and Source Specific Query the Multicast Address field contains the multicast address of interest, while the Source Address [i] field(s) contain(s) the source address(es) of interest.

5.1.14. Source Addresses for Queries

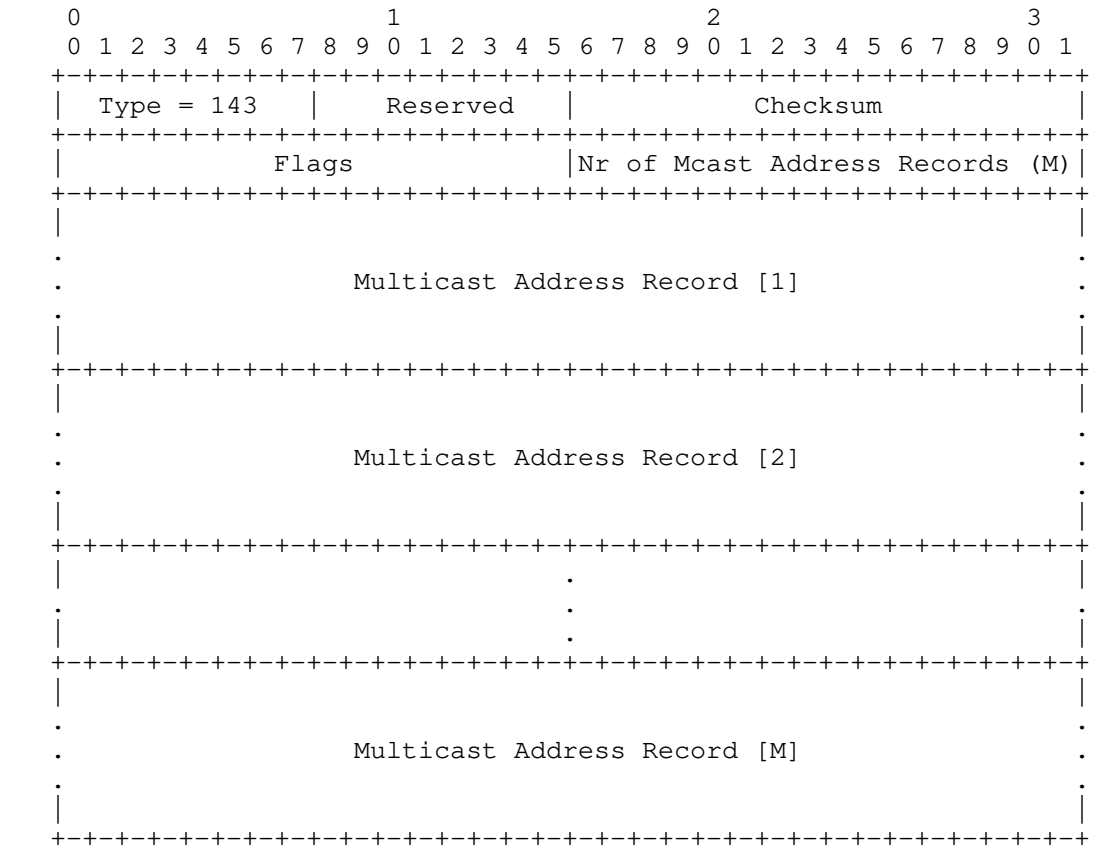
All MLDv2 Queries MUST be sent with a valid IPv6 link-local source address. If a node (router or host) receives a Query message with the IPv6 Source Address set to the unspecified address (::), or any other address that is not a valid IPv6 link-local address, it MUST silently discard the message and SHOULD log a warning.

5.1.15. Destination Addresses for Queries

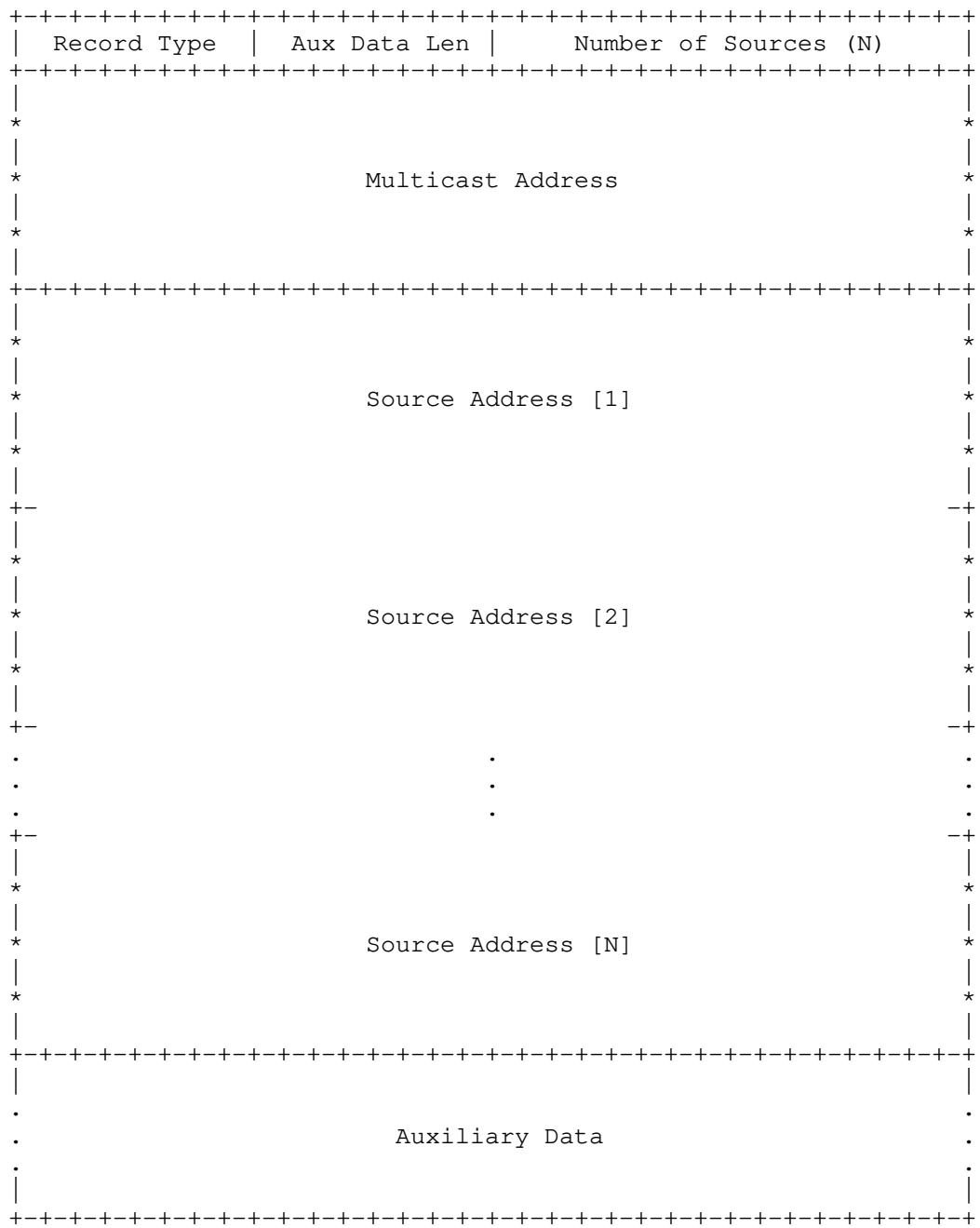
In MLDv2, General Queries are sent to the link-scope all-nodes multicast address (ff02::1). Multicast Address Specific and Multicast Address and Source Specific Queries are sent with an IP destination address equal to the multicast address of interest. However, a node MUST accept and process any Query whose IP Destination Address field contains any of the addresses (unicast or multicast) assigned to the interface on which the Query arrives. This might be useful, e.g., for debugging purposes.

5.2. Version 2 Multicast Listener Report Message

Version 2 Multicast Listener Reports are sent by IP nodes to report (to neighboring routers) the current multicast listening state, or changes in the multicast listening state, of their interfaces. Reports have the following format:



Each Multicast Address Record has the following internal format:


Record Type
Aux Data Len
Number of Sources (N)
Multicast Address
Source Address [1]
Source Address [2]
Source Address [N]
Auxiliary Data

5.2.1. Reserved

The Reserved field is set to zero on transmission, and ignored on reception.

5.2.2. Checksum

The standard ICMPv6 checksum; it covers the entire MLDv2 message, plus a "pseudo-header" of IPv6 header fields [RFC8200][RFC4443]. In order to compute the checksum, the Checksum field is set to zero. When a packet is received, the checksum MUST be verified before processing it.

5.2.3. Flags

Allocation of individual bits within the Flags field is described in Section 2.3 of [I-D.ietf-pim-3228bis]. Future specifications will define the associated meaning tied to any such allocation.

5.2.4. Nr of Mcast Address Records (M)

The Nr of Mcast Address Records (M) field specifies how many Multicast Address Records are present in this Report.

5.2.5. Multicast Address Record

Each Multicast Address Record is a block of fields that contain information on the sender listening to a single multicast address on the interface from which the Report is sent.

5.2.6. Record Type

It specifies the type of the Multicast Address Record. See Section 5.2.13 for a detailed description of the different possible Record Types.

5.2.7. Aux Data Len

The Aux Data Len field contains the length of the Auxiliary Data Field in this Multicast Address Record, in units of 32-bit words. It may contain zero, to indicate the absence of any auxiliary data.

5.2.8. Number of Sources (N)

The Number of Sources (N) field specifies how many source addresses are present in this Multicast Address Record.

5.2.9. Multicast Address

The Multicast Address field contains the multicast address to which this Multicast Address Record pertains.

5.2.10. Source Address [i]

The Source Address [i] fields are a vector of n unicast addresses, where n is the value in this record's Number of Sources (N) field.

5.2.11. Auxiliary Data

The Auxiliary Data field, if present, contains additional information that pertain to this Multicast Address Record. The protocol specified in this document, MLDv2, does not define any auxiliary data. Therefore, implementations of MLDv2 MUST NOT include any auxiliary data (i.e., MUST set the Aux Data Len field to zero) in any transmitted Multicast Address Record, and MUST ignore any such data present in any received Multicast Address Record. The semantics and the internal encoding of the Auxiliary Data field are to be defined by any future version or extension of MLD that uses this field.

5.2.12. Additional Data

If the Payload Length field in the IPv6 header of a received Report indicates that there are additional octets of data present, beyond the last Multicast Address Record, MLDv2 implementations MUST include those octets in the computation to verify the received MLD Checksum, but MUST otherwise ignore those additional octets. When sending a Report, an MLDv2 implementation MUST NOT include additional octets beyond the last Multicast Address Record.

5.2.13. Multicast Address Record Types

There are a number of different types of Multicast Address Records that may be included in a Report message:

* A "Current State Record" is sent by a node in response to a Query received on an interface. It reports the current listening state of that interface, with respect to a single multicast address. The Record Type of a Current State Record may be one of the following two values:

1 - MODE_IS_INCLUDE - indicates that the interface has a filter mode of INCLUDE for the specified multicast address. The Source Address [i] fields in this Multicast Address Record contain the interface's source list for the specified multicast address. A MODE_IS_INCLUDE Record is never sent with an empty source list.

2 - MODE_IS_EXCLUDE - indicates that the interface has a filter mode of EXCLUDE for the specified multicast address. The Source Address [i] fields in this Multicast Address Record contain the interface's source list for the specified multicast address, if it is non-empty. An SSM-aware host SHOULD NOT send a MODE_IS_EXCLUDE record type for multicast addresses that fall within the SSM address range as they will be ignored by SSM-aware routers [RFC4604].

* A "Filter Mode Change Record" is sent by a node whenever a local invocation of IPv6MulticastListen causes a change of the filter mode (i.e., a change from INCLUDE to EXCLUDE, or from EXCLUDE to INCLUDE) of the interface-level state entry for a particular multicast address, whether the source list changes at the same time or not. The Record is included in a Report sent from the interface on which the change occurred. The Record Type of a Filter Mode Change Record may be one of the following two values:

  3 - CHANGE_TO_INCLUDE_MODE - indicates that the interface has changed to INCLUDE filter mode for the specified multicast address. The Source Address [i] fields in this Multicast Address Record contain the interface's new source list for the specified multicast address, if it is non-empty.

  4 - CHANGE_TO_EXCLUDE_MODE - indicates that the interface has changed to EXCLUDE filter mode for the specified multicast address. The Source Address [i] fields in this Multicast Address Record contain the interface's new source list for the specified multicast address, if it is non-empty. An SSM-aware host SHOULD NOT send a CHANGE_TO_EXCLUDE_MODE record type for multicast addresses that fall within the SSM address range.

* A "Source List Change Record" is sent by a node whenever a local invocation of IPv6MulticastListen causes a change of source list that is not coincident with a change of filter mode, of the interface-level state entry for a particular multicast address. The Record is included in a Report sent from the interface on which the change occurred. The Record Type of a Source List Change Record may be one of the following two values:

5 - ALLOW_NEW_SOURCES - indicates that the Source Address [i] fields in this Multicast Address Record contain a list of the additional sources that the node wishes to listen to, for packets sent to the specified multicast address. If the change was to an INCLUDE source list, these are the addresses that were added to the list; if the change was to an EXCLUDE source list, these are the addresses that were deleted from the list.

6 - BLOCK_OLD_SOURCES - indicates that the Source Address [i] fields in this Multicast Address Record contain a list of the sources that the node no longer wishes to listen to, for packets sent to the specified multicast address. If the change was to an INCLUDE source list, these are the addresses that were deleted from the list; if the change was to an EXCLUDE source list, these are the addresses that were added to the list.

If a change of source list results in both allowing new sources and blocking old sources, then two Multicast Address Records are sent for the same multicast address, one of type ALLOW_NEW_SOURCES and one of type BLOCK_OLD_SOURCES.

We use the term "State Change Record" to refer to either a Filter Mode Change Record or a Source List Change Record.

Multicast Address Records with an unrecognized Record Type value MUST be silently ignored, with the rest of the report being processed.

In the rest of this document, we use the following notation to describe the contents of a Multicast Address Record that pertains to a particular multicast address:

```
IS_IN ( x )  -  Type MODE_IS_INCLUDE, source addresses x
IS_EX ( x )  -  Type MODE_IS_EXCLUDE, source addresses x
TO_IN ( x )  -  Type CHANGE_TO_INCLUDE_MODE, source addresses x
TO_EX ( x )  -  Type CHANGE_TO_EXCLUDE_MODE, source addresses x
ALLOW ( x )  -  Type ALLOW_NEW_SOURCES, source addresses x
BLOCK ( x )  -  Type BLOCK_OLD_SOURCES, source addresses x
```

where x is either:

* a capital letter (e.g., "A") to represent the set of source addresses, or

* a set expression (e.g., "A+B"), where "A+B" means the union of sets A and B, "A*B" means the intersection of sets A and B, and "A-B" means the removal of all elements of set B from set A.

5.2.14. Source Addresses for Reports

An MLDv2 Report MUST be sent with a valid IPv6 link-local source address, or the unspecified address (::), if the sending interface has not acquired a valid link-local address yet. Sending reports with the unspecified address is allowed to support the use of IP multicast in the Neighbor Discovery Protocol [RFC4861]. For stateless autoconfiguration, as defined in [RFC4862], a node is required to join several IPv6 multicast groups, in order to perform Duplicate Address Detection (DAD). Prior to DAD, the only address the reporting node has for the sending interface is a tentative one, which cannot be used for communication. Thus, the unspecified address must be used.

On the other hand, routers MUST silently discard a message that is not sent with a valid link-local address, without taking any action on the contents of the packet. Thus, a Report is discarded if the router cannot identify the source address of the packet as belonging to a link connected to the interface on which the packet was received. A Report sent with the unspecified address is also discarded by the router. This enhances security, as unidentified reporting nodes cannot influence the state of the MLDv2 router(s). Nevertheless, the reporting node has modified its listening state for multicast addresses that are contained in the Multicast Address Records of the Report message. From now on, it will treat packets sent to those multicast addresses according to this new listening state. Once a valid link-local address is available, a node SHOULD generate new MLDv2 Report messages for all multicast addresses joined on the interface.

5.2.15. Destination Addresses for Reports

Version 2 Multicast Listener Reports are sent with an IP destination address of ff02::16, to which all MLDv2-capable multicast routers listen (see Section 11 for IANA considerations related to this special destination address). A node that operates in version 1 compatibility mode (see details in Section 8) sends version 1 Reports to the multicast address specified in the Multicast Address field of the Report. In addition, a node MUST accept and process any version 1 Report whose IP Destination Address field contains any of the IPv6 addresses (unicast or multicast) assigned to the interface on which the Report arrives. This might be useful, e.g., for debugging purposes.

5.2.16. Multicast Listener Report Size

If the set of Multicast Address Records required in a Report does not fit within the size limit of a single Report message (as determined by the MTU of the link on which it will be sent), the Multicast Address Records are sent in as many Report messages as needed to report the entire set.

If a single Multicast Address Record contains so many source addresses that it does not fit within the size limit of a single Report message, then:

* if its Type is not IS_EX or TO_EX, it is split into multiple Multicast Address Records; each such record contains a different subset of the source addresses, and is sent in a separate Report.

* if its Type is IS_EX or TO_EX, a single Multicast Address Record is sent, with as many source addresses as can fit; the remaining source addresses are not reported. Although the choice of which sources to report is arbitrary, it is preferable to report the same set of sources in each subsequent report, rather than reporting different sources each time.

6. Protocol Description for Multicast Address Listeners

MLD is an asymmetric protocol, as it specifies separate behaviors for multicast address listeners -- that is, hosts or routers that listen to multicast packets -- and multicast routers. This section describes the part of MLDv2 that applies to all multicast address listeners. (Note that a multicast router that is also a multicast address listener performs both parts of MLDv2; it receives and it responds to its own MLD messages, as well as to those of its neighbors.) The multicast router part of MLDv2 is described in Section 7.

A node performs the protocol described in this section over all interfaces on which multicast reception is supported, even if more than one of those interfaces are connected to the same link.

For interoperability with multicast routers that run the MLDv1 protocol, nodes maintain a Host Compatibility Mode variable for each interface on which multicast reception is supported. This section describes the behavior of multicast address listener nodes on interfaces for which Host Compatibility Mode = MLDv2. The algorithm for determining Host Compatibility Mode, and the behavior if its value is set to MLDv1, are described in Section 8.

The link-scope all-nodes multicast address, (ff02::1), is handled as a special case. On all nodes -- that is all hosts and routers, including multicast routers -- listening to packets destined to the all-nodes multicast address, from all sources, is permanently enabled on all interfaces on which multicast listening is supported. No MLD messages are ever sent regarding neither the link-scope all-nodes multicast address, nor any multicast address of scope 0 (reserved) or 1 (node-local). Multicast listeners MUST send MLD messages for all multicast addresses except for the link-scope all-nodes multicast address and any multicast addresses of scope less than 2.

There are three types of events that trigger MLDv2 protocol actions on an interface:

* a change of the per-interface listening state, caused by a local invocation of IPv6MulticastListen;

* the firing of a specific timer;

* the reception of a Query.

(Received MLD messages of types other than Query are silently ignored, except as required for interoperation with nodes that implement MLDv1.)

The following subsections describe the actions to be taken for each case. Timer and counter names appear in square brackets. Default values for those timers and counters are specified in Section 9.

6.1. Action on Change of Per-Interface State

An invocation of IPv6MulticastListen may cause the multicast listening state of an interface to change, according to the rules in Section 4.2. Each such change affects the per-interface entry for a single multicast address.

A change of per-interface state causes the node to immediately transmit a State Change Report from that interface. The type and contents of the Multicast Address Record(s) in that Report are determined by comparing the filter mode and source list for the affected multicast address before and after the change, according to Table 1. If no per-interface state existed for that multicast address before the change (i.e., the change consisted of creating a new per-interface record), or if no state exists after the change (i.e., the change consisted of deleting a per-interface record), then the "non-existent" state is considered to have an INCLUDE filter mode and an empty source list.

Old State	New State	State Change Record Sent
INCLUDE (A)	INCLUDE (B)	ALLOW (B-A), BLOCK (A-B)
EXCLUDE (A)	EXCLUDE (B)	ALLOW (A-B), BLOCK (B-A)
INCLUDE (A)	EXCLUDE (B)	TO_EX (B)
EXCLUDE (A)	INCLUDE (B)	TO_IN (B)

Table 1: State Change Record Transmission Logic

If the computed source list for either an ALLOW or a BLOCK State Change Record is empty, that record is omitted from the Report.

To cover the possibility of the State Change Report being missed by one or more multicast routers, [Robustness Variable] - 1 retransmissions are scheduled, through a Retransmission Timer, at intervals chosen at random from the range (0, [Unsolicited Report Interval]).

If more changes to the same per-interface state entry occur before all the retransmissions of the State Change Report for the first change have been completed, each such additional change triggers the immediate transmission of a new State Change Report.

The contents of the new Report are calculated as follows:

* As for the first Report, the per-interface state for the affected multicast address before and after the latest change is compared.

* The records that express the difference are built according to the table above. Nevertheless, these records are not transmitted in a separate message, but they are instead merged with the contents of the pending report, to create the new State Change Report. The rules for calculating this merged report are described below.

The transmission of the merged State Change Report terminates retransmissions of the earlier State Change Reports for the same multicast address, and becomes the first of [Robustness Variable] transmissions of the new State Change Reports. These transmissions are necessary in order to ensure that each instance of state change is transmitted at least [Robustness Variable] times.

Each time a source is included in the difference report calculated above, retransmission state for that source needs to be maintained until [Robustness Variable] State Change Reports have been sent by the node. This is done in order to ensure that a series of successive state changes do not break the protocol robustness. Sources in retransmission state can be kept in a per multicast address Retransmission List, with a Source Retransmission Counter associated to each source in the list. When a source is included in the list, its counter is set to [Robustness Variable]. Each time a State Change Report is sent the counter is decreased by one unit. When the counter reaches zero, the source is deleted from the Retransmission List for that multicast address.

If the per-interface listening change that triggers the new report is a filter mode change, then the next [Robustness Variable] State Change Reports will include a Filter Mode Change Record. This applies even if any number of source list changes occur in that period. The node has to maintain retransmission state for the multicast address until the [Robustness Variable] State Change Reports have been sent. This can be done through a per multicast address Filter Mode Retransmission Counter. When the filter mode changes, the counter is set to [Robustness Variable]. Each time a State Change Report is sent the counter is decreased by one unit. When the counter reaches zero, i.e., [Robustness Variable] State Change Reports with Filter Mode Change Records have been transmitted after the last filter mode change, and if source list changes have resulted in additional reports being scheduled, then the next State Change Report will include Source List Change Records.

Each time a per-interface listening state change triggers the Immediate transmission of a new State Change Report, its contents are determined as follows. If the report should contain a Filter Mode Change Record, i.e., the Filter Mode Retransmission Counter for that multicast address has a value higher than zero, then, if the current filter mode of the interface is INCLUDE, a TO_IN record is included in the report; otherwise a TO_EX record is included. If instead the report should contain Source List Change Records, i.e., the Filter Mode Retransmission Counter for that multicast address is zero, an ALLOW and a BLOCK record is included. The contents of these records are built according Table 2.

Record	Sources Included
TO_IN	All in the current per-interface state that must be forwarded
TO_EX	All in the current per-interface state that must be blocked
ALLOW	All with retransmission state (i.e., all sources from the Retransmission List) that must be forwarded
BLOCK	All with retransmission state that must be blocked

Table 2: Per-Interface State Change Report Contents

If the computed source list for either an ALLOW or a BLOCK record is empty, that record is omitted from the State Change Report.

Note: When the first State Change Report is sent, the non-existent pending report to merge with can be treated as a Source Change Report with empty ALLOW and BLOCK records (no sources have retransmission state).

The building of a scheduled State Change Report, triggered by the firing of a Retransmission Timer, instead of a per-interface listening state change, is described in Section 6.3.

6.2. Action on Reception of a Query

Upon reception of an MLD message that contains a Query, the node checks if the source address of the message is a valid link-local address, if the Hop Limit is set to 1, and if the Router Alert option is present in the Hop-By-Hop Options header of the IPv6 packet. If any of these checks fails, the packet is dropped.

If the validity of the MLD message is verified, the node starts to process the Query. Instead of responding immediately, the node delays its response by a random amount of time, bounded by the Maximum Response Delay value derived from the Maximum Response Code in the received Query message. A node may receive a variety of Queries on different interfaces and of different kinds (e.g., General Queries, Multicast Address Specific Queries, and Multicast Address and Source Specific Queries), each of which may require its own delayed response.

Before scheduling a response to a Query, the node must first consider previously scheduled pending responses and, in many cases, schedule a combined response. Therefore, for each of its interfaces on which it operates the listener part of the MLDv2 protocol, the node must be able to maintain the following state:

* an Interface Timer for scheduling responses to General Queries;

* a Multicast Address Timer for scheduling responses to Multicast Address (and Source) Specific Queries, for each multicast address the node has to report on;

* a per-multicast-address list of sources to be reported in response to a Multicast Address and Source Specific Query.

When a new valid General Query arrives on an interface, the node checks whether it has any per-interface listening state record to report on, or not. Similarly, when a new valid Multicast Address (and Source) Specific Query arrives on an interface, the node checks whether it has a per-interface listening state record that corresponds to the queried multicast address (and source), or not. If it does, a delay for a response is randomly selected in the range (0, [Maximum Response Delay]), where Maximum Response Delay is derived from the Maximum Response Code inserted in the received Query message. The following rules are then used to determine if a Report needs to be scheduled or not, and the type of Report to schedule. (The rules are considered in order and only the first matching rule is applied.)

1. If there is a pending response to a previous General Query scheduled sooner than the selected delay, no additional response needs to be scheduled.

2. If the received Query is a General Query, the Interface Timer is used to schedule a response to the General Query after the selected delay. Any previously pending response to a General Query is canceled.

3. If the received Query is a Multicast Address Specific Query or a Multicast Address and Source Specific Query and there is no pending response to a previous Query for this multicast address, then the Multicast Address Timer is used to schedule a report. If the received Query is a Multicast Address and Source Specific Query, the list of queried sources is recorded to be used when generating a response.

4. If there is already a pending response to a previous Query scheduled for this multicast address, and either the new Query is a Multicast Address Specific Query or the recorded source list associated with the multicast address is empty, then the multicast address source list is cleared and a single response is scheduled, using the Multicast Address Timer. The new response is scheduled to be sent at the earliest of the remaining time for the pending report and the selected delay.

5. If the received Query is a Multicast Address and Source Specific Query and there is a pending response for this multicast address with a non-empty source list, then the multicast address source list is augmented to contain the list of sources in the new Query, and a single response is scheduled using the Multicast Address Timer. The new response is scheduled to be sent at the earliest of the remaining time for the pending report and the selected delay.

6.3. Action on Timer Expiration

There are several timers that, upon expiration, trigger protocol actions on an MLDv2 Multicast Address Listener node. All these actions are related to pending reports scheduled by the node.

1. If the expired timer is the Interface Timer (i.e., there is a pending response to a General Query), then one Current State Record is sent for each multicast address for which the specified interface has listening state, as described in Section 4.2. The Current State Record carries the multicast address and its associated filter mode (MODE_IS_INCLUDE or MODE_IS_EXCLUDE) and Source list. Multiple Current State Records are packed into individual Report messages, to the extent possible.

   This naive algorithm may result in bursts of packets when a node listens to a large number of multicast addresses. Instead of using a single Interface Timer, implementations are recommended to spread transmission of such Report messages over the interval (0, [Maximum Response Delay]). Note that any such implementation MUST avoid the "ack-implosion" problem, i.e., MUST NOT send a Report immediately upon reception of a General Query.

2. If the expired timer is a Multicast Address Timer and the list of recorded sources for that multicast address is empty (i.e., there is a pending response to a Multicast Address Specific Query), then if, and only if, the interface has listening state for that multicast address, a single Current State Record is sent for that address. The Current State Record carries the multicast address and its associated filter mode (MODE_IS_INCLUDE or MODE_IS_EXCLUDE) and source list, if any.

3. If the expired timer is a Multicast Address Timer and the list of recorded sources for that multicast address is non-empty (i.e., there is a pending response to a Multicast Address and Source Specific Query), then if, and only if, the interface has listening state for that multicast address, the contents of the corresponding Current State Record are determined from the per-interface state and the pending response record, as specified in Table 3.

Per-Interface State	Set of Sources in the Pending Response Record	Current State Record
INCLUDE (A)	B	IS_IN (A*B)
EXCLUDE (A)	B	IS_IN (B-A)

Table 3: Determining Contents of Current State Record

If the resulting Current State Record has an empty set of source addresses, then no response is sent. After the required Report messages have been generated, the source lists associated with any reported multicast addresses are cleared.

4. If the expired timer is a Retransmission Timer for a multicast address (i.e., there is a pending State Change Report for that multicast address), the contents of the report are determined as follows. If the report should contain a Filter Mode Change Record, i.e., the Filter Mode Retransmission Counter for that multicast address has a value higher than zero, then, if the current filter mode of the interface is INCLUDE, a TO_IN record is included in the report; otherwise a TO_EX record is included. In both cases, the Filter Mode Retransmission Counter for that multicast address is decremented by one unit after the transmission of the report.

If instead the report should contain Source List Change Records, i.e., the Filter Mode Retransmission Counter for that multicast address is zero, an ALLOW and a BLOCK record is included. The contents of these records are built according to Table 4.

Record	Sources included
TO_IN	All in the current per-interface state that must be forwarded
TO_EX	All in the current per-interface state that must be blocked
ALLOW	All with retransmission state (i.e., all sources from the Retransmission List) that must be forwarded. For each included source, its Source Retransmission Counter is decreased with one unit after the transmission of the report. If the counter reaches zero, the source is deleted from the Retransmission List for that multicast address.
BLOCK	All with retransmission state (i.e., all sources from the Retransmission List) that must be blocked. For each included source, its Source Retransmission Counter is decreased with one unit after the transmission of the report. If the counter reaches zero, the source is deleted from the Retransmission List for that multicast address.

Table 4: Determining Contents of Source List Change Records

If the computed source list for either an ALLOW or a BLOCK record is empty, that record is omitted from the State Change Report.

7. Description of the Protocol for Multicast Routers

The purpose of MLD is to enable each multicast router to learn, for each of its directly attached links, which multicast addresses have listeners on that link. MLD version 2 adds the capability for a multicast router to also learn which sources have listeners among the neighboring nodes, for packets sent to any particular multicast address. The information gathered by MLD is provided to whichever multicast routing protocol is used by the router, in order to ensure that multicast packets are delivered to all links where there are interested listeners.

This section describes the part of MLDv2 that is performed by multicast routers. Multicast routers may themselves become multicast address listeners, and therefore also perform the multicast listener part of MLDv2, described in Section 6.

A multicast router performs the protocol described in this section over each of its directly attached links. If a multicast router has more than one interface to the same link, it only needs to operate this protocol over one of those interfaces.

For each interface over which the router operates the MLD protocol, the router must configure that interface to listen to all link-layer multicast addresses that can be generated by IPv6 multicasts. For example, an Ethernet-attached router must set its Ethernet address reception filter to accept all Ethernet multicast addresses that start with the hexadecimal value 3333 [RFC2464]; in the case of an Ethernet interface that does not support the filtering of such a multicast address range, it must be configured to accept ALL Ethernet multicast addresses, in order to meet the requirements of MLD.

On each interface over which this protocol is being run, the router MUST enable reception of the link-scope "all MLDv2-capable routers" multicast address from all sources, and MUST perform the multicast address listener part of MLDv2 for that address on that interface.

Multicast routers only need to know that at least one node on an attached link listens to packets for a particular multicast address from a particular source; a multicast router is not required to individually keep track of the interests of each neighboring node. (Nevertheless, see Appendix A.2 item 1 for discussion.)

MLDv2 is backward compatible with the MLDv1 protocol. For a detailed description of compatibility issues see Section 8.

7.1. Conditions for MLD Queries

The behavior of a router that implements the MLDv2 protocol depends on whether there are several multicast routers on the same subnet, or not. If it is the case, a querier election mechanism (described in Section 7.6.2) is used to elect a single multicast router to be in Querier state. All the multicast routers on the subnet listen to the messages sent by multicast address listeners, and maintain the same multicast listening information state, so that they can quickly and correctly take over the querier functionality, should the present Querier fail. Nevertheless, it is only the Querier that sends periodical or triggered query messages on the subnet.

The Querier periodically sends General Queries to request Multicast Address Listener information from an attached link. These queries are used to build and refresh the Multicast Address Listener state of routers on attached links.

Nodes respond to these queries by reporting their Multicast Address Listening state (and set of sources they listen to) with Current State Multicast Address Records in MLDv2 Multicast Listener Reports.

As a listener of a multicast address, a node may express interest in listening or not listening to traffic from particular sources. As the desired listening state of a node changes, it reports these changes using Filter Mode Change Records or Source List Change Records. These records indicate an explicit state change in a multicast address at a node in either the Multicast Address Record's source list or its filter mode. When Multicast Address Listening is terminated at a node or traffic from a particular source is no longer desired, the Querier must query for other listeners of the multicast address or of the source before deleting the multicast address (or source) from its Multicast Address Listener state and pruning its traffic.

To enable all nodes on a link to respond to changes in multicast address listening, the Querier sends specific queries. A Multicast Address Specific Query is sent to verify that there are no nodes that listen to the specified multicast address or to "rebuild" the listening state for a particular multicast address. Multicast Address Specific Queries are sent when the Querier receives a State Change Record indicating that a node ceases to listen to a multicast address. They are also sent in order to enable a fast transition of a router from EXCLUDE to INCLUDE mode, in case a received State Change Record motivates this action.

A Multicast Address and Source Specific Query is used to verify that there are no nodes on a link which listen to traffic from a specific set of sources. Multicast Address and Source Specific Queries list sources for a particular multicast address which have been requested to no longer be forwarded. This query is sent by the Querier in order to learn if any node listens to packets sent to the specified multicast address, from the specified source addresses. Multicast Address and Source Specific Queries are only sent in response to State Change Records and never in response to Current State Records. Section 5.1.13 describes each query in more detail.

7.2. MLD State Maintained by Multicast Routers

Multicast routers that implement the MLDv2 protocol keep state per multicast address per attached link. This multicast address state consists of a filter mode, a list of sources, and various timers. For each attached link on which MLD runs, a multicast router records the listening state for that link. That state conceptually consists of a set of records of the form:

```
(IPv6 multicast address, Filter Timer,
 Router Filter Mode, (source records) )
```

Each source record is of the form:

```
(IPv6 source address, source timer)
```

If all sources for a multicast address are listened to, an empty source record list is kept with the Router Filter Mode set to EXCLUDE. This means that nodes on this link want all sources for this multicast address to be forwarded. This is the MLDv2 equivalent of an MLDv1 listening state.

7.2.1. Definition of Router Filter Mode

To reduce internal state, MLDv2 routers keep a filter mode per multicast address per attached link. This filter mode is used to summarize the total listening state of a multicast address to a minimum set such that all nodes' listening states are respected. The filter mode may change in response to the reception of particular types of Multicast Address Records or when certain timer conditions occur. In the following sections, we use the term "Router Filter Mode" to refer to the filter mode of a particular multicast address within a router. Section 7.4 describes the changes of the Router Filter Mode per Multicast Address Record received.

A router is in INCLUDE mode for a specific multicast address on a given interface if all the listeners on the link interested in that address are in INCLUDE mode. The router state is represented through the notation INCLUDE (A), where A is called the "Include List". The Include List is the set of sources that one or more listeners on the link have requested to receive. All the sources from the Include List will be forwarded by the router. Any other source that is not in the Include List will be blocked by the router.

A router is in EXCLUDE mode for a specific multicast address on a given interface if there is at least one listener in EXCLUDE mode interested in that address on the link. Conceptually, when a Multicast Address Record is received, the Router Filter Mode for that

multicast address is updated to cover all the requested sources using the least amount of state. As a rule, once a Multicast Address Record with a filter mode of EXCLUDE is received, the Router Filter Mode for that multicast address will be set to EXCLUDE. Nevertheless, if all nodes with a multicast address record having filter mode set to EXCLUDE cease reporting, it is desirable for the Router Filter Mode for that multicast address to transition back to INCLUDE mode. This transition occurs when the Filter Timer expires, and is explained in detail in Section 7.5.

When the router is in EXCLUDE mode, the router state is represented through the notation EXCLUDE (X,Y), where X is called the "Requested List" and Y is called the "Exclude List". All sources, except those from the Exclude List, will be forwarded by the router. The Requested List has no effect on forwarding. Nevertheless, it has to be maintained for several reasons, as explained in Section 7.2.3.

The exact handling of both the INCLUDE and EXCLUDE mode router state, according to the received reports, is presented in details in Section 7.4.1 and Section 7.4.2.

7.2.2. Definition of Filter Timers

The Filter Timer is only used when the router is in EXCLUDE mode for a specific multicast address, and it represents the time for the Router Filter Mode of the multicast address to expire and switch to INCLUDE mode. A Filter Timer is a decrementing timer with a lower bound of zero. One Filter Timer exists per multicast address record. Filter Timers are updated according to the types of Multicast Address Records received.

If a Filter Timer expires, with the Router Filter Mode for that multicast address being EXCLUDE, it means that there are no more listeners in EXCLUDE mode on the attached link. At this point, the router transitions to INCLUDE filter mode. Section 7.5 describes the actions taken when a Filter Timer expires while in EXCLUDE mode.

Table 5 summarizes the role of the Filter Timer. Section 7.4 describes the details of setting the Filter Timer per type of Multicast Address Record received.

Router Filter Mode	Filter Timer Value	Actions/Comments
INCLUDE	Not Used	All listeners in INCLUDE mode.

EXCLUDE	Timer > 0	At least one listener in EXCLUDE mode.
EXCLUDE	Timer == 0	No more listeners in EXCLUDE mode for the multicast address. If the Requested List is empty, delete Multicast Address Record. If not, switch to INCLUDE filter mode; the sources in the Requested List are moved to the Include List, and the Exclude List is deleted.

Table 5: Filter Timer Management

7.2.3. Definition of Source Timers

A Source Timer is a decrementing timer with a lower bound of zero. One Source Timer is kept per source record. Source timers are updated according to the type and filter mode of the Multicast Address Record received. Section 7.4 describes the setting of source timers per type of Multicast Address Records received.

In the following, abbreviations are used for several variables (all of which are described in detail in Section 9). The variable MALI stands for the Multicast Address Listening Interval, which is the time in which multicast address listening state will time out. The variable LLQT is the Last Listener Query Time, which is the total time the router should wait for a report, after the Querier has sent the first query. During this time, the Querier should send [Last Member Query Count]-1 retransmissions of the query. LLQT represents the "leave latency", or the difference between the transmission of a listener state change and the modification of the information passed to the routing protocol.

If the router is in INCLUDE filter mode, a source can be added to the current Include List if a listener in INCLUDE mode sends a Current State or a State Change Report which includes that source. Each source from the Include List is associated with a source timer that is updated whenever a listener in INCLUDE mode sends a report that confirms its interest in that specific source. If the timer of a source from the Include List expires, the source is deleted from the Include List. If there are no more source records left, the multicast address record is deleted from the router.

Besides this "soft leave" mechanism, there is also a "fast leave" scheme in MLDv2; it is also based on the use of source timers. When a node in INCLUDE mode expresses its desire to stop listening to a

specific source, all the multicast routers on the link lower their timer for that source to a small interval of LLQT milliseconds. The Querier then sends then a Multicast Address and Source Specific Query, to verify whether there are other listeners for that source on the link, or not. If a corresponding report is received before the timer expires, all the multicast routers on the link update their source timer. If not, the source is deleted from the Include List. The handling of the Include List, according to the received reports, is detailed in Section 7.4.1 and Section 7.4.2.

Source timers are treated differently when the Router Filter Mode for a multicast address is EXCLUDE. For sources from the Requested List the source timers have running values; these sources are forwarded by the router. For sources from the Exclude List the source timers are set to zero; these sources are blocked by the router. If the timer of a source from the Requested List expires, the source is moved to the Exclude List. The router informs then the routing protocol that there is no longer a listener on the link interested in traffic from this source.

The router has to maintain the Requested List for two reasons:

* To keep track of sources that listeners in INCLUDE mode listen to. This is necessary in order to assure a seamless transition of the router to INCLUDE mode, when there will be no listener in EXCLUDE mode left. This transition should not interrupt the flow of traffic to the listeners in INCLUDE mode still interested in that multicast address. Therefore, at the moment of the transition, the Requested List should represent the set of sources that nodes in INCLUDE mode have explicitly requested.

  When the router switches to INCLUDE mode, the sources in the Requested List are moved to the Include List, and the Exclude List is deleted. Before the switch, the Requested List can contain an inexact guess at the sources that listeners in INCLUDE mode listen to - might be too large or too small. These inexactitudes are due to the fact that the Requested List is also used for fast blocking purposes, as described below. If such a fast blocking is required, some sources may be deleted from the Requested List (as shown in Section 7.4.1 and Section 7.4.2) in order to reduce router state. Nevertheless, in each such case the Filter Timer is updated as well. Therefore, listeners in INCLUDE mode will have enough time, before an eventual switching, to reconfirm their interest in the eliminated source(s), and rebuild the Requested List accordingly. The protocol ensures that when a switch to INCLUDE mode occurs, the Requested List will be accurate. Details about the transition of the router to INCLUDE mode are presented in Appendix A.3.

* To allow a fast blocking of previously unblocked sources. If the router receives a report that contains such a request, the concerned sources are added to the Requested List. Their timers are set to a small interval of LLQT milliseconds, and a Multicast Address and Source Specific Query is sent by the Querier, to check whether there are nodes on the link still interested in those sources, or not. If no node confirms its interest in receiving a specific source, the timer of that source expires. Then, the source is moved from the Requested List to the Exclude List. From then on, the source will be blocked by the router.

The handling of the EXCLUDE mode router state, according to the received reports, is detailed in Section 7.4.1 and Section 7.4.2.

When the Router Filter Mode for a multicast address is EXCLUDE, source records are only deleted when the Filter Timer expires, or when newly received Multicast Address Records modify the source record list of the router.

7.3. MLDv2 Source Specific Forwarding Rules

When a multicast router receives a datagram from a source destined to a particular multicast address, a decision has to be made whether to forward the datagram on an attached link or not. The multicast routing protocol in use is in charge of this decision, and should use the MLDv2 information to ensure that all sources/multicast addresses that have listeners on a link are forwarded to that link. MLDv2 information does not override multicast routing information; for example, if the MLDv2 filter mode for a multicast address is EXCLUDE, a router may still forward packets for excluded sources to a transit link.

To summarize, the following table describes the forwarding suggestions made by MLDv2 to the routing protocol for traffic originating from a source destined to a multicast address. It also summarizes the actions taken upon the expiration of a source timer based on the Router Filter Mode of the multicast address.

Router Filter Mode	Source Timer Value	Action
INCLUDE	TIMER > 0	Suggest to forward traffic from source
INCLUDE	TIMER == 0	Suggest to stop forwarding traffic from source and remove source record. If there are no more source records, delete multicast address record
EXCLUDE	TIMER > 0	Suggest to forward traffic from source
EXCLUDE	TIMER == 0	Suggest to not forward traffic from source. Move the source from the Requested List to the Exclude List (DO NOT remove source record)
EXCLUDE	No Source Element	Suggest to forward traffic from all sources

Table 6

7.4. Action on Reception of Reports

Upon reception of an MLD message that contains a Report, the router checks if the source address of the message is a valid link-local address, if the Hop Limit is set to 1, and if the Router Alert option is present in the Hop-By-Hop Options header of the IPv6 packet. If any of these checks fails, the packet is dropped. If the validity of the MLD message is verified, the router starts to process the Report.

SSM-aware routers SHOULD ignore records that contain multicast addresses in the SSM address range if the record type is MODE_IS_EXCLUDE or CHANGE_TO_EXCLUDE_MODE. SSM-aware routers SHOULD ignore MLDv1 Report and DONE messages that contain multicast addresses in the SSM address range, SHOULD NOT use such Reports to establish IP forwarding state, and MAY log an error if it receives such a message.

7.4.1. Reception of Current State Records

When receiving Current State Records, a router updates both its Filter Timer and its source timers. In some circumstances, the reception of a type of multicast address record will cause the Router Filter Mode for that multicast address to change. Table 7 describes the actions, with respect to state and timers, that occur to a router's state upon reception of Current State Records.

If the router is in INCLUDE filter mode for a multicast address, we will use the notation INCLUDE (A), where A denotes the associated Include List. If the router is in EXCLUDE filter mode for a multicast address, we will use the notation EXCLUDE (X,Y), where X and Y denote the associated Requested List and Exclude List respectively.

Within the "Actions" section of the router state tables, we use the notation '(A)=J', which means that the set A of source records should have their source timers set to value J. 'Delete (A)' means that the set A of source records should be deleted. 'Filter Timer = J' means that the Filter Timer for the multicast address should be set to value J.

Router State	Report Received	New Router State	Actions
INCLUDE (A)	IS_IN (B)	INCLUDE (A+B)	(B)=MALI
INCLUDE (A)	IS_EX (B)	EXCLUDE (A*B, B-A)	(B-A)=0 Delete (A-B) Filter Timer=MALI
EXCLUDE (X,Y)	IS_IN (A)	EXCLUDE (X+A, Y-A)	(A)=MALI
EXCLUDE (X,Y)	IS_EX (A)	EXCLUDE (A-Y, Y*A)	(A-X-Y)=MALI Delete (X-A) Delete (Y-A) Filter Timer=MALI

Table 7: Actions for Received Current State Records

7.4.2. Reception of Filter Mode Change and Source List Change Records

When a change in the global state of a multicast address occurs in a node, the node sends either a Source List Change Record or a Filter Mode Change Record for that multicast address. As with Current State Records, routers must act upon these records and possibly change their own state to reflect the new listening state of the link.

The Querier must query sources or multicast addresses that are requested to be no longer forwarded. When a router queries or receives a query for a specific set of sources, it lowers its source timers for those sources to a small interval of Last Listener Query Time milliseconds. If multicast address records are received in response to the queries which express interest in listening the queried sources, the corresponding timers are updated.

Multicast Address Specific queries can also be used in order to enable a fast transition of a router from EXCLUDE to INCLUDE mode, in case a received Multicast Address Record motivates this action. The

Filter Timer for that multicast address is lowered to a small interval of Last Listener Query Time milliseconds. If any multicast address records that express EXCLUDE mode interest in the multicast address are received within this interval, the Filter Timer is updated and the suggestion to the routing protocol to forward the multicast address stands without any interruption. If not, the router will switch to INCLUDE filter mode for that multicast address.

During the query period (i.e., Last Listener Query Time milliseconds) the MLD component in the router continues to suggest to the routing protocol to forward traffic from the multicast addresses or sources that are queried. It is not until after Last Listener Query Time milliseconds without receiving a record that expresses interest in the queried multicast address or sources that the router may prune the multicast address or sources from the link.

Table 8 describes the changes in multicast address state and the action(s) taken when receiving either Filter Mode Change or Source List Change Records. Table 8 also describes the queries which are sent by the Querier when a particular report is received.

We use the following notation for describing the queries that are sent. We use the notation 'Q(MA)' to describe a Multicast Address Specific Query to the MA multicast address. We use the notation 'Q(MA,A)' to describe a Multicast Address and Source Specific Query to the MA multicast address with source list A. If source list A is null as a result of the action (e.g. A*B), then no query is sent as a result of the operation.

In order to maintain protocol robustness, queries defined in the Actions column of Table 8 need to be transmitted [Last Listener Query Count] times, once every [Last Listener Query Interval] period.

If while scheduling new queries, there are already pending queries to be retransmitted for the same multicast address, the new and pending queries have to be merged. In addition, received host reports for a multicast address with pending queries may affect the contents of those queries. Section 7.6.3 describes the process of building and maintaining the state of pending queries.

Router State	Report Received	New Router State	Actions
INCLUDE (A)	ALLOW (B)	INCLUDE(A+B)	(B)=MALI
INCLUDE (A)	BLOCK (B)	INCLUDE(A)	Send Q(MA,A*B)
INCLUDE (A)	TO_EX (B)	EXCLUDE(A*B,B-A)	(B-A)=0, Delete (A-B), Send Q(MA,A*B), Filter Timer=MALI
INCLUDE (A)	TO_IN (B)	INCLUDE(A+B)	(B)=MALI, Send Q(MA,A-B)
EXCLUDE (X,Y)	ALLOW (A)	EXCLUDE(X+A,Y-A)	(A)=MALI
EXCLUDE (X,Y)	BLOCK (A)	EXCLUDE(X+(A-Y),Y)	(A-X-Y)=Filter Timer, Send Q(MA,A-Y)
EXCLUDE (X,Y)	TO_EX (A)	EXCLUDE(A-Y,Y*A)	(A-X-Y)=Filter Timer, Delete (X-A), Delete (Y-A), Send Q(MA,A-Y), Filter Timer=MALI
EXCLUDE (X,Y)	TO_IN (A)	EXCLUDE(X+A,Y-A)	(A)=MALI, Send Q(MA,X-A), Send Q(MA)

Table 8: Multicast Router State Transitions

7.5. Switching Router Filter Modes

The Filter Timer is used as a mechanism for transitioning the Router Filter Mode from EXCLUDE to INCLUDE.

When a Filter Timer expires with a Router Filter Mode of EXCLUDE, a router assumes that there are no nodes with a filter mode of EXCLUDE present on the attached link. Thus, the router transitions to INCLUDE filter mode for the multicast address.

A router uses the sources from the Requested List as its state for the switch to a filter mode of INCLUDE. Sources from the Requested List are moved in the Include List, while sources from the Exclude List are deleted. For example, if a router's state for a multicast

address is EXCLUDE(X,Y) and the Filter Timer expires for that multicast address, the router switches to filter mode of INCLUDE with state INCLUDE(X). If at the moment of the switch the Requested List (X) is empty, the multicast address record is deleted from the router.

7.6. Action on Reception of Queries

Upon reception of an MLD message that contains a Query, the router checks if the source address of the message is a valid link-local address, if the Hop Limit is set to 1, and if the Router Alert option is present in the Hop-By-Hop Options header of the IPv6 packet. If any of these checks fails, the packet is dropped.

If the validity of the MLD message is verified, the router starts to process the Query.

7.6.1. Timer Updates

MLDv2 uses the Suppress Router-Side Processing flag to ensure robustness, as explained in Section 2.1. When a router sends or receives a query with a clear Suppress Router-Side Processing flag, it must update its timers to reflect the correct timeout values for the multicast address or sources being queried. The following table describes the timer actions when sending or receiving a Multicast Address Specific or Multicast Address and Source Specific Query with the Suppress Router-Side Processing flag not set.

Query	Action
Q(MA,A)	Source Timers for sources in A are lowered to LLQT
Q(MA)	Filter Timer is lowered to LLQT

When a router sends or receives a query with the Suppress Router-Side Processing flag set, it will not update its timers.

7.6.2. Querier Election

MLDv2 elects a single router per subnet to be in Querier state; all the other routers on the subnet should be in Non-Querier state. MLDv2 uses the same querier election mechanism as MLDv1, namely the IPv6 address. When a router starts operating on a subnet, by default it considers itself as being the Querier. Thus, it sends several General Queries separated by a small time interval (see Section 9.6 and Section 9.7 for details).

When a router receives a query with a lower IPv6 address than its own, it sets the Other Querier Present timer to Other Querier Present Timeout; if it was previously in Querier state, it switches to Non-Querier state and ceases to send queries on the link. After the Other Querier Present timer expires, it should re-enter the Querier state and begin sending General Queries.

All MLDv2 queries MUST be sent with the FE80::/64 link-local source address prefix. Therefore, for the purpose of MLDv2 querier election, an IPv6 address A is considered to be lower than an IPv6 address B if the interface ID represented by the last 64 bits of address A, in big-endian bit order, is lower than the interface ID represented by the last 64 bits of address B.

7.6.3. Building and Sending Specific Queries

7.6.3.1. Building and Sending Multicast Address Specific Queries

When a table action "Send Q(MA)" is encountered, the Filter Timer must be lowered to LLQT. The Querier must then immediately send a Multicast Address Specific query as well as schedule [Last Listener Query Count - 1] query retransmissions to be sent every [Last Listener Query Interval], over [Last Listener Query Time].

When transmitting a Multicast Address Specific Query, if the Filter Timer is larger than LLQT, the "Suppress Router-Side Processing" bit is set in the query message.

7.6.3.2. Building and Sending Multicast Address and Source Specific Queries

When a table action "Send Q(MA,X)" is encountered by the Querier in the table in Section 7.4.2, the following actions must be performed for each of the sources in X that send to multicast address MA, with source timer larger than LLQT:

* Lower source timer to LLQT;

* Add the sources to the Retransmission List;

* Set the Source Retransmission Counter for each source to [Last Listener Query Count].

The Querier must then immediately send a Multicast Address and Source Specific Query as well as schedule [Last Listener Query Count -1] query retransmissions to be sent every [Last Listener Query Interval], over [Last Listener Query Time]. The contents of these queries are calculated as follows.

When building a Multicast Address and Source Specific Query for a multicast address MA, two separate query messages are sent for the multicast address. The first one has the "Suppress Router-Side Processing" bit set and contains all the sources with retransmission state (i.e., sources from the Retransmission List of that multicast address), and timers greater than LLQT. The second has the "Suppress Router-Side Processing" bit clear and contains all the sources with retransmission state and timers lower or equal to LLQT. If either of the two calculated messages does not contain any sources, then its transmission is suppressed.

Note: If a Multicast Address Specific query is scheduled to be transmitted at the same time as a Multicast Address and Source specific query for the same multicast address, then transmission of the Multicast Address and Source Specific message with the "Suppress Router-Side Processing" bit set may be suppressed.

8. Interoperation with MLDv1

MLD version 2 hosts and routers interoperate with hosts and routers that have not yet been upgraded to MLDv2. This compatibility is maintained by hosts and routers taking appropriate actions depending on the versions of MLD operating on hosts and routers within a network.

8.1. Query Version Distinctions

The MLD version of a Multicast Listener Query message is determined as follows:

MLDv1 Query: length = 24 octets

MLDv2 Query: length >= 28 octets

Query messages that do not match any of the above conditions (e.g., a Query of length 26 octets) MUST be silently ignored.

8.2. Multicast Address Listener Behavior

8.2.1. In the Presence of MLDv1 Routers

In order to be compatible with MLDv1 routers, MLDv2 hosts MUST operate in version 1 compatibility mode. MLDv2 hosts MUST keep state per local interface regarding the compatibility mode of each attached link. A host’s compatibility mode is determined from the Host Compatibility Mode variable which can be in one of the two states: MLDv1 or MLDv2.

The Host Compatibility Mode of an interface is set to MLDv1 whenever an MLDv1 Multicast Address Listener Query is received on that interface. At the same time, the Older Version Querier Present timer for the interface is set to Older Version Querier Present Timeout seconds. The timer is re-set whenever a new MLDv1 Query is received on that interface. If the Older Version Querier Present timer expires, the host switches back to Host Compatibility Mode of MLDv2.

When Host Compatibility Mode is MLDv2, a host acts using the MLDv2 protocol on that interface. When Host Compatibility Mode is MLDv1, a host acts in MLDv1 compatibility mode, using only the MLDv1 protocol, on that interface.

An MLDv1 Querier will send General Queries with the Maximum Response Code set to the desired Maximum Response Delay, i.e., the full range of this field is linear and the exponential algorithm described in Section 5.1.3. is not used.

Whenever a host changes its compatibility mode, it cancels all its pending responses and retransmission timers.

An SSM-aware host that receives an MLDv1 General Query or MLDv1 Group Specific Query for a multicast address in the SSM address range SHOULD log an error. It is RECOMMENDED that implementions provide a configuration option to disable use of Host Compatibility Mode to allow networks to operate only in SSM mode. This configuration option SHOULD be disabled by default.

8.2.2. In the Presence of MLDv1 Multicast Address Listeners

An MLDv2 host may be placed on a link where there are MLDv1 hosts. A host MAY allow its MLDv2 Multicast Listener Report to be suppressed by a Version 1 Multicast Listener Report.

8.3. Multicast Router Behavior

8.3.1. In the Presence of MLDv1 Routers

MLDv2 routers may be placed on a network where there is at least one MLDv1 router. The following requirements apply:

* If an MLDv1 router is present on the link, the Querier MUST use the lowest version of MLD present on the network. This must be administratively assured. Routers that desire to be compatible with MLDv1 MUST have a configuration option to act in MLDv1 mode; if an MLDv1 router is present on the link, the system administrator must explicitly configure all MLDv2 routers to act in MLDv1 mode. When in MLDv1 mode, the Querier MUST send periodic

General Queries truncated at the Multicast Address field (i.e., 24 bytes long), and SHOULD also warn about receiving an MLDv2 Query (such warnings MUST be rate-limited). The Querier MUST also fill in the Maximum Response Delay in the Maximum Response Code field, i.e., the exponential algorithm described in Section 5.1.3 is not used.

* If a router is not explicitly configured to use MLDv1 and receives an MLDv1 General Query, it SHOULD log a warning. These warnings MUST be rate-limited.

* It is RECOMMENDED that implementions provide a configuration option to disable use of compatibility mode to allow networks to operate only in SSM mode. This configuration option SHOULD be disabled by default.

8.3.2. In the Presence of MLDv1 Multicast Address Listeners

MLDv2 routers may be placed on a network where there are hosts that have not yet been upgraded to MLDv2. In order to be compatible with MLDv1 hosts, MLDv2 routers MUST operate in version 1 compatibility mode. MLDv2 routers keep a compatibility mode per multicast address record. The compatibility mode of a multicast address is determined from the Multicast Address Compatibility Mode variable, which can be in one of the two following states: MLDv1 or MLDv2.

The Multicast Address Compatibility Mode of a multicast address record is set to MLDv1 whenever an MLDv1 Multicast Listener Report is received for that multicast address. At the same time, the Older Version Host Present timer for the multicast address is set to Older Version Host Present Timeout seconds. The timer is re-set whenever a new MLDv1 Report is received for that multicast address. If the Older Version Host Present timer expires, the router switches back to Multicast Address Compatibility Mode of MLDv2 for that multicast address.

Note that when a router switches back to MLDv2 Multicast Address Compatibility Mode for a multicast address, it takes some time to regain source-specific state information. Source-specific information will be learned during the next General Query, but sources that should be blocked will not be blocked until [Multicast Address Listening Interval] after that.

When Multicast Address Compatibility Mode is MLDv2, a router acts using the MLDv2 protocol for that multicast address. When Multicast Address Compatibility Mode is MLDv1, a router internally translates the following MLDv1 messages for that multicast address to their MLDv2 equivalents (Table 9).

MLDv1 Message	MLDv2 Equivalent
Report	IS_EX({})
Done	TO_IN({})

Table 9: MLD Message Translation

MLDv2 BLOCK messages are ignored, as are source-lists in TO_EX() messages (i.e., any TO_EX() message is treated as TO_EX({})). On the other hand, the Querier continues to send MLDv2 queries, regardless of its Multicast Address Compatibility Mode.

9. List of Timers, Counters, and their Default Values

Most of these timers are configurable. If non-default settings are used, they MUST be consistent among all nodes on a single link. Note that parentheses are used to group expressions to make the algebra clear.

9.1. Robustness Variable

The Robustness Variable allows tuning for the expected packet loss on a link. If a link is expected to be lossy, the value of the Robustness Variable may be increased. MLD is robust to [Robustness Variable] - 1 packet losses. The value of the Robustness Variable MUST NOT be zero, and SHOULD NOT be one. Default value: 2.

9.2. Query Interval

The Query Interval variable denotes the interval between General Queries sent by the Querier. Default value: 125 seconds.

By varying the [Query Interval], an administrator may tune the number of MLD messages on the link; larger values cause MLD Queries to be sent less often.

9.3. Query Response Interval

The Maximum Response Delay used to calculate the Maximum Response Code inserted into the periodic General Queries. Default value: 10000 (10 seconds)

By varying the [Query Response Interval], an administrator may tune the burstiness of MLD messages on the link; larger values make the traffic less bursty, as host responses are spread out over a larger interval. The number of seconds represented by the [Query Response Interval] must be less than the [Query Interval].

9.4. Multicast Address Listening Interval

The Multicast Address Listening Interval (MALI) is the amount of time that must pass before a multicast router decides there are no more listeners of a multicast address or a particular source on a link. This value MUST be ([Robustness Variable] times [Query Interval]) plus 2 times [Query Response Interval].

9.5. Other Querier Present Timeout

The Other Querier Present Timeout is the length of time that must pass before a multicast router decides that there is no longer another multicast router which should be the Querier. This value MUST be ([Robustness Variable] times ([Query Interval]) plus (one half of [Query Response Interval]).

9.6. Startup Query Interval

The Startup Query Interval is the interval between General Queries sent by a Querier on startup. Default value: 1/4 the [Query Interval].

9.7. Startup Query Count

The Startup Query Count is the number of Queries sent out on startup, separated by the Startup Query Interval. Default value: [Robustness Variable].

9.8. Last Listener Query Interval

The Last Listener Query Interval is the Maximum Response Delay used to calculate the Maximum Response Code inserted into Multicast Address Specific Queries sent in response to Version 1 Multicast Listener Done messages. It is also the Maximum Response Delay used to calculate the Maximum Response Code inserted into Multicast Address and Source Specific Query messages. Default value: 1000 (1 second).

Note that for values of LLQI greater than 32.768 seconds, a limited set of values can be represented, corresponding to sequential values of Maximum Response Code. When converting a configured time to a Maximum Response Code value, it is recommended to use the exact value if possible, or the next lower value if the requested value is not exactly representable.

This value may be tuned to modify the "leave latency" of the link. A reduced value results in reduced time to detect the departure of the last listener for a multicast address or source.

9.9. Last Listener Query Count

The Last Listener Query Count is the number of Multicast Address Specific Queries sent before the router assumes there are no local listeners. The Last Listener Query Count is also the number of Multicast Address and Source Specific Queries sent before the router assumes there are no listeners for a particular source. Default value: [Robustness Variable].

9.10. Last Listener Query Time

The Last Listener Query Time is the time value represented by the Last Listener Query Interval, multiplied by [Last Listener Query Count]. It is not a tunable value, but may be tuned by changing its components.

9.11. Unsolicited Report Interval

The Unsolicited Report Interval is the time between repetitions of a node’s initial report of interest in a multicast address. Default value: 1 second.

9.12. Older Version Querier Present Timeout

The Older Version Querier Present Timeout is the time-out for transitioning a host back to MLDv2 Host Compatibility Mode. When an MLDv1 query is received, MLDv2 hosts set their Older Version Querier Present Timer to [Older Version Querier Present Timeout].

This value MUST be ([Robustness Variable] times (the [Query Interval] in the last Query received)) plus ([Query Response Interval]).

9.13. Older Version Host Present Timeout

The Older Version Host Present Timeout is the time-out for transitioning a router back to MLDv2 Multicast Address Compatibility Mode for a specific multicast address. When an MLDv1 report is received for that multicast address, routers set their Older Version Host Present Timer to [Older Version Host Present Timeout].

This value MUST be ([Robustness Variable] times [Query Interval]) plus ([Query Response Interval]).

9.14. Configuring timers

This section is meant to provide advice to network administrators on how to tune these settings to their network. Ambitious router implementations might tune these settings dynamically based upon changing characteristics of the network.

9.14.1. Robustness Variable

The Robustness Variable tunes MLD to expected losses on a link. MLDv2 is robust to [Robustness Variable] - 1 packet losses, e.g., if the Robustness Variable is set to the default value of 2, MLDv2 is robust to a single packet loss but may operate imperfectly if more losses occur. On lossy links, the value of the Robustness Variable should be increased to allow for the expected level of packet loss. However, increasing the value of the Robustness Variable increases the leave latency of the link (the time between when the last listener stops listening to a source or multicast address and when the traffic stops flowing).

9.14.2. Query Interval

The overall level of periodic MLD traffic is inversely proportional to the Query Interval. A longer Query Interval results in a lower overall level of MLD traffic. The value of the Query Interval MUST be equal to or greater than the Maximum Response Delay used to calculate the Maximum Response Code inserted in General Query messages.

9.14.3. Maximum Response Delay

The burstiness of MLD traffic is inversely proportional to the Maximum Response Delay. A longer Maximum Response Delay will spread Report messages over a longer interval. However, a longer Maximum Response Delay in Multicast Address Specific and Multicast Address And Source Specific Queries extends the leave latency (the time between when the last listener stops listening to a source or

multicast address and when the traffic stops flowing.) The expected rate of Report messages can be calculated by dividing the expected number of Reporters by the Maximum Response Delay. The Maximum Response Delay may be dynamically calculated (shown in Table 10) per Query by using the expected number of Reporters for that Query.

Query Type	Expected Number of Reporters
General Query	All nodes on link
Multicast Address Specific Query	All nodes on the link that had expressed interest in the multicast address
Multicast Address and Source Specific Query	All nodes on the link that had expressed interest in the source and multicast address

Table 10: Maximum Response Delay Calculation

A router is not required to calculate these populations or tune the Maximum Response Delay dynamically; these are simply guidelines.

10. Security Considerations

We consider the ramifications of a forged message of each type. Note that before processing an MLD message, nodes verify if the source address of the message is a valid link-local address (or the unspecified address), if the Hop Limit is set to 1, and if the Router Alert option is present in the Hop-By-Hop Options header of the IPv6 packet. If any of these checks fails, the packet is dropped. This defends the MLDv2 nodes from acting on forged MLD messages originated off-link. Therefore, in the following we discuss only the effects of on-link forgery.

10.1. Query Message

A forged Query message from a machine with a lower IPv6 address than the current Querier will cause Querier duties to be assigned to the forger. If the forger then sends no more Query messages, other routers' Other Querier Present timer will time out and one will resume the role of Querier. During this time, if the forger ignores Multicast Listener Done Messages, traffic might flow to multicast addresses with no listeners for up to [Multicast Address Listener Interval].

A forged Version 1 Query message will put MLDv2 listeners on that link in MLDv1 Host Compatibility Mode. This scenario can be avoided by providing MLDv2 hosts with a configuration option to ignore Version 1 messages completely.

A DoS attack on a node could be staged through forged Multicast Address and Source Specific Queries. The attacker can find out about the listening state of a specific node with a general query. After that it could send a large number of Multicast Address and Source Specific Queries, each with a large source list and/or long Maximum Response Delay. The node will have to store and maintain the sources specified in all of those queries for as long as it takes to send the delayed response. This would consume both memory and CPU cycles in order to augment the recorded sources with the source lists included in the successive queries.

To protect against such a DoS attack, a node stack implementation could restrict the number of Multicast Address and Source Specific Queries per multicast address within this interval, and/or record only a limited number of sources.

10.2. Current State Report messages

A forged Report message may cause multicast routers to think there are listeners of a multicast address on a link when there are not. Nevertheless, since listening to a multicast address on a host is generally an unprivileged operation, a local user may trivially gain the same result without forging any messages.

A forged Version 1 Report Message may put a router into MLDv1 Multicast Address Compatibility Mode for a particular multicast address, meaning that the router will ignore MLDv2 source specific state messages. This can cause traffic to flow from unwanted sources for up to [Multicast Address Listener Interval]. This can be solved by providing routers with a configuration switch to ignore Version 1 messages completely. This breaks automatic compatibility with Version 1 hosts, so it should only be used in situations where source filtering is critical.

10.3. State Change Report messages

A forged State Change Report message will cause the Querier to send out Multicast Address Specific or Multicast Address and Source Specific Queries for the multicast address in question. This causes extra processing on each router and on each listener of the multicast address, but cannot cause loss of desired traffic.

11. IANA Considerations

IANA has assigned the IPv6 link-local multicast address ff02::16, called "all MLDv2-capable routers", as described in Section 5.2.15. Version 2 Multicast Listener Reports will be sent to this special address. The reference for this assignment should be changed to this document upon publication as an RFC.

In addition, IANA has assigned the ICMPv6 message type value of 143 for Version 2 Multicast Listener Report messages, as specified in Section 4. The reference for this assignment should be changed to this document upon publication as an RFC.

12. Contributors

Roland Vida, Luis Henrique Maciel Kosmalski Costa, Serge Fdida, Steve Deering, Bill Fenner, and Isidor Kouvelas are the authors of RFC 3810, which makes up the majority of the content in this document.

Anuj Budhiraja, Toerless Eckert, Olufemi Komolafe and Tim Winters have contributed valuable content to this version of the specification.

13. Acknowledgments

We would like to thank Hitoshi Asaeda, Randy Bush, Francis Dupont, Ted Hardie, Russ Housley, Konstantin Kabassanov, Erik Nordmark, Shinsuke Suzuki, Margaret Wasserman, Bert Wijnen, and Remi Zara for their valuable comments and suggestions on this document.

Stig Venaas, Hitoshi Asaeda, and Mike McBride have provided valuable feedback on this version of the specification and we thank them for their input.

14. References

14.1. Normative References

[I-D.ietf-pim-3228bis]
Haberman, B., "IANA Considerations for Internet Group Management Protocols", Work in Progress, Internet-Draft, draft-ietf-pim-3228bis-06, 13 June 2024, <https://datatracker.ietf.org/api/v1/doc/document/draft-ietf-pim-3228bis/>.

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

[RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, DOI 10.17487/RFC2464, December 1998, <https://www.rfc-editor.org/info/rfc2464>.

[RFC2710] Deering, S., Fenner, W., and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6", RFC 2710, DOI 10.17487/RFC2710, October 1999, <https://www.rfc-editor.org/info/rfc2710>.

[RFC2711] Partridge, C. and A. Jackson, "IPv6 Router Alert Option", RFC 2711, DOI 10.17487/RFC2711, October 1999, <https://www.rfc-editor.org/info/rfc2711>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[RFC4604] Holbrook, H., Cain, B., and B. Haberman, "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast", RFC 4604, DOI 10.17487/RFC4604, August 2006, <https://www.rfc-editor.org/info/rfc4604>.

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <https://www.rfc-editor.org/info/rfc4607>.

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

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

14.2. Informative References

[RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, DOI 10.17487/RFC3376, October 2002, <https://www.rfc-editor.org/info/rfc3376>.

[RFC3569] Bhattacharyya, S., Ed., "An Overview of Source-Specific Multicast (SSM)", RFC 3569, DOI 10.17487/RFC3569, July 2003, <https://www.rfc-editor.org/info/rfc3569>.

[RFC3678] Thaler, D., Fenner, B., and B. Quinn, "Socket Interface Extensions for Multicast Source Filters", RFC 3678, DOI 10.17487/RFC3678, January 2004, <https://www.rfc-editor.org/info/rfc3678>.

[RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <https://www.rfc-editor.org/info/rfc3810>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <https://www.rfc-editor.org/info/rfc4861>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <https://www.rfc-editor.org/info/rfc4862>.

Appendix A. Design Rationale

A.1. The Need for State Change Messages

MLDv2 specifies two types of Multicast Listener Reports: Current State and State Change. This section describes the rationale for the need for both these types of Reports.

Routers need to distinguish Multicast Listener Reports that were sent in response to Queries from those that were sent as a result of a change in the per-interface state. Multicast Listener Reports that are sent in response to Multicast Address Listener Queries are used mainly to refresh the existing state at the router; they typically do not cause transitions in state at the router. Multicast Listener Reports that are sent in response to changes in the per-interface state require the router to take some action in response to the received report (see Section 7.4).

The inability to distinguish between the two types of reports would force a router to treat all Multicast Listener Reports as potential changes in state and could result in increased processing at the router as well as an increase in MLD traffic on the link.

A.2. Host Suppression

In MLDv1, a host would not send a pending multicast listener report if a similar report was sent by another listener on the link. In MLDv2, the suppression of multicast listener reports has been removed. The following points explain this decision.

1. Routers may want to track per-host multicast listener status on an interface. This would allow routers to implement fast leaves (e.g., for layered multicast congestion control schemes), as well as track listener status for possible security or accounting purposes. The present specification does not require routers to implement per-host tracking. Nevertheless, the lack of host suppression in MLDv2 makes possible to implement either proprietary or future standard behavior of multicast routers that would support per-host tracking, while being fully interoperable with MLDv2 listeners and routers that implement the exact behavior described in this specification.

2. Multicast Listener Report suppression does not work well on bridged LANs. Many bridges and Layer2/Layer3 switches that implement MLD snooping do not forward MLD messages across LAN segments in order to prevent multicast listener report suppression.

3. By eliminating multicast listener report suppression, hosts have fewer messages to process; this leads to a simpler state machine implementation.

4. In MLDv2, a single multicast listener report now bundles multiple multicast address records to decrease the number of packets sent. In comparison, the previous version of MLD required that each multicast address be reported in a separate message.

A.3. Switching router filter modes from EXCLUDE to INCLUDE

If on a link there are nodes in both EXCLUDE and INCLUDE modes for a single multicast address, the router must be in EXCLUDE mode as well (see section 7.2.1). In EXCLUDE mode, a router forwards traffic from all sources except those in the Exclude List. If all nodes in EXCLUDE mode cease to exist or to listen, it would be desirable for the router to switch back to INCLUDE mode seamlessly, without interrupting the flow of traffic to existing listeners.

One of the ways to accomplish this is for routers to keep track of all sources that nodes that are in INCLUDE mode listen to, even though the router itself is in EXCLUDE mode. If the Filter Timer for a multicast address expires, it implies that there are no nodes in EXCLUDE mode on the link (otherwise a multicast listener report from that node would have refreshed the Filter Timer). The router can then switch to INCLUDE mode seamlessly; sources from the Requested List are moved to the Include List, while sources from the Exclude List are deleted.

Appendix B. Summary of Changes

B.1. MLDv1

The following is a summary of changes from MLDv1, specified in [RFC2710].

* MLDv2 introduces source filtering.

* The IP service interface of MLDv2 nodes is modified accordingly. It enables the specification of a filter mode and a source list.

* An MLDv2 node keeps per-socket and per-interface multicast listening states that include a filter mode and a source list for each multicast address. This enables packet filtering based on a socket's multicast reception state.

* MLDv2 state kept on routers includes a filter mode and a list of sources and source timers for each multicast address that has listeners on the link. MLDv1 routers kept only the list of multicast addresses.

* Queries include additional fields (Section 5.1).

* The S flag (Suppress Router-Side Processing) is included in queries in order to fix robustness issues.

* The Querier's Robustness Variable and Query Interval Code are included in Queries in order to synchronize all MLDv2 routers connected to the same link.

* A new Query type (Multicast Address and Source Specific Query) is introduced.

* The Maximum Response Delay is not directly included in the Query anymore. Instead, an exponential algorithm is used to calculate its value, based on the Maximum Response Code included in the Query. The maximum value is increased from 65535 milliseconds to about 140 minutes.

* Reports include Multicast Address Records. Information on the listening state for several different multicast addresses can be included in the same Report message.

* Reports are sent to the "all MLDv2-capable multicast routers" address, instead of the multicast address the host listens to, as in MLDv1. This facilitates the operation of layer-2 snooping switches.

* There is no "host suppression", as in MLDv1. All nodes send Report messages.

* Unsolicited Reports, announcing changes in receiver listening state, are sent [Robustness Variable] times. RFC 2710 is less explicit.

* There are no Done messages.

* Interoperability with MLDv1 systems is achieved by MLDv2 state operations.

* In order to ensure interoperability, hosts maintain a Host Compatibility Mode variable and an Older Version Querier Present timer per interface. Routers maintain a Multicast Address Compatibility Mode variable and an Older Version Host Present timer per multicast address.

B.2. Changes since RFC 3810

The following summarizes the changes made since [RFC3810].

* Added definition of Resv to address Erratum 4773.

* Added clarifying text on which multicast addresses require the sending of MLD messages to address Erratum 5977.

* Added text to clarify the Group Membership Interval timer changes from Erratum 6725.

* Changed Reserved field in messages to Flags field to facilitate use of an IANA-managed registry for future bit allocations.

Author's Address

Brian Haberman (editor)
Johns Hopkins University Applied Physics Lab
Email: brian@innovationslab.net

RIFT WG
Internet-Draft
Intended status: Standards Track
Expires: 23 January 2025

Z. Zhang
Y. Wei
ZTE Corporation
S. Ma
Google
X. Liu
Alef Edge
B. Rijsman
Individual

22 July 2024

YANG Data Model for Routing in Fat Trees (RIFT)
draft-ietf-rift-yang-16

Abstract

This document defines a YANG data model for the configuration and management of Routing in Fat Trees (RIFT) Protocol. The model is based on YANG 1.1 as defined in RFC7950 and conforms to the Network Management Datastore Architecture (NMDA) as described in RFC8342.

Status of This Memo

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

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

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

This Internet-Draft will expire on 23 January 2025.

Table of Contents

1. Introduction

RFC Ed.: Please replace all occurrences of 'I-D.ietf-rift-rift' with the actual RFC number of draft-ietf-rift-rift (and remove this note).

[I-D.ietf-rift-rift] introduces the protocol definition of RIFT. This document defines a YANG data model that can be used to configure and manage the RIFT protocol. This model imports and augments ietf-routing YANG model defined in [RFC8349].

1.1. Terminology

The terminology for describing YANG data models is found in [RFC6020] and [RFC7950], including:

* augment

* container

* choice

* data model
* data node
* grouping
* identity
* leaf
* leaf-list
* list
* module
* uses

The following terminologies and abbreviations are used in this document and the defined model:

The content is copied from [I-D.ietf-rift-rift] for reading convenience.

Clos/Fat Tree: It refers to a folded spine-and-leaf topology with possibly multiple Points of Delivery (PoDs) and one or multiple Top of Fabric (ToF) planes.

RIFT: Routing in Fat Trees [I-D.ietf-rift-rift].

LIE: "Link Information Element" are exchanged on all the system's links running RIFT to form _ThreeWay_ adjacencies and carry information used to perform Zero Touch Provisioning (ZTP) of levels.

PoD: "Point of Delivery" means a self-contained vertical slice or subset of a Clos or Fat Tree network containing normally only level 0 and level 1 nodes. A node in a PoD communicates with nodes in other PoDs via the ToF nodes. PoDs are numbered to distinguish them and PoD value 0 is used to denote "undefined" or "any" PoD.

ThreeWay Adjacency: A unique adjacency between two nodes over a point-to-point interface and exchange local configuration and necessary RIFT ZTP information. An adjacency is only advertised in Node TIEs and used for computations after it achieved _ThreeWay_ state, i.e. both routers reflected each other in LIEs including relevant security information. Nevertheless, LIEs before _ThreeWay_ state is reached may carry RIFT ZTP related information already.

TIE: "Topology Information Element" are exchanged between RIFT nodes to describe parts of a network such as links and address prefixes. A TIE has always a direction and a type. North TIEs (sometimes abbreviated as N-TIEs) are used when dealing with TIEs in the northbound representation and South-TIEs (sometimes abbreviated as S-TIEs) for the southbound equivalent. TIEs have different types such as node and prefix TIEs.

ToF: "Top of Fabric" is the set of nodes that provide inter-PoD communication and have no northbound adjacencies, i.e. are at the "very top" of the fabric. ToF nodes do not belong to any PoD and are assigned default PoD value to indicate the equivalent of "any" PoD.

1.2. Conventions Used in This Document

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

1.3. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340].

1.4. Prefixes in Data Node Names

In this document, names of data nodes, actions, and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the corresponding YANG module, as shown in Table 1.

Prefix	YANG module	Reference
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]
rt	ietf-routing	[RFC8349]
if	ietf-interfaces	[RFC8343]
rt-types	ietf-routing-types	[RFC8294]
iana-rt-types	iana-routing-types	[RFC8294]
key-chain	ietf-key-chain	[RFC8177]

Table 1

2. Design of the Data Model

2.1. Scope of Model

The model covers RIFT [I-D.ietf-rift-rift].

This model can be used to configure and manage the RIFT protocol. The operational state data and statistics can be retrieved by this model. The subscription and push mechanism defined in [RFC8639] and [RFC8641] can be implemented by the user to subscribe to notifications on the data nodes in this model.

The model contains all the basic configuration parameters to operate the protocol. Depending on the implementation choices, some systems may not allow some of the advanced parameters to be configurable. The occasionally implemented parameters are modeled as optional features in this model. This model can be extended, and it has been structured in a way that such extensions can be conveniently made.

The RIFT YANG module augments the /routing/control-plane-protocols/ control-plane-protocol path defined in the ietf-routing module. The ietf-rift model defines a single instance of RIFT. Multiple instances are instantiated as multiple control-plane protocols instances.

2.2. Specification

This model imports and augments ietf-routing YANG model defined in [RFC8349]. Both configuration branch and state branch of [RFC8349] are augmented. The configuration branch covers node base and policy configuration. The container "rift" is the top level container in this data model. The container is expected to enable RIFT protocol functionality.

The YANG data model defined in this document conforms to the Network Management Datastore Architecture (NMDA) [RFC8342]. The operational state data is combined with the associated configuration data in the same hierarchy [RFC8407].

2.3. Overview

The RIFT YANG module defined in this document has all the common building blocks for the RIFT protocol.

The RIFT YANG module augments the /routing/control-plane-protocols/ control-plane-protocol path defined in the ietf-routing module. The ietf-rift model defines a single instance of RIFT. Multiple instances are instantiated as multiple control-plane protocols instances.

```
module: ietf-rift
  augment /rt:routing/rt:control-plane-protocols
            /rt:control-plane-protocol:
    +--rw rift* [name]
       +--rw name                          string
       +--ro node-level?                   level
       +--rw system-id                     system-id
       +--rw fabric-id?                    uint16
       +--rw pod?                          uint32
       +--rw fabric-prefix?                inet:ip-prefix
       +--rw fabric-prefix-advertise?      boolean
       +--rw configured-level?             level
       +--rw overload
       |  +--rw overload?                    boolean
       |  +--rw (timeout-type)?
       |     +--:(on-startup)
       |     |  +--rw on-startup-timeout?
       |     |           rt-types:timer-value-seconds16
       |     +--:(immediate)
       |        +--rw immediate-timeout?
       |                 rt-types:timer-value-seconds16
       +--ro proto-major-ver               uint8
       +--ro proto-minor-ver               uint16
```

```
        +--rw node-capabilities
        |  +--rw hierarchy-indications?   enumeration
        |  +--rw flood-reduction?         boolean
        +--rw maximum-nonce-delta?         uint8 {nonce-delta-adjust}?
        +--rw nonce-increasing-interval?   uint16
        +--rw adjusted-lifetime?
        |       rt-types:timer-value-seconds16
        +--rw rx-lie-multicast-addr
        |  +--rw ipv4?   inet:ipv4-address
        |  +--rw ipv6?   inet:ipv6-address
        +--rw tx-lie-multicast-addr
        |  +--rw ipv4?   inet:ipv4-address
        |  +--rw ipv6?   inet:ipv6-address
        +--rw lie-tx-port?                 inet:port-number
        +--rw global-link-capabilities
        |  +--rw bfd-capable?              boolean
        |  +--rw v4-forwarding-capable?    boolean
        |  +--rw mtu-size?                 uint32
        +--rw tide-generation-interval?
        |       rt-types:timer-value-seconds16
        +--rw tie-security* [security-type] {tie-security}?
        |  +--rw security-type             enumeration
        |  +--rw shared?                   boolean
        |  +--rw (auth-key-chain)?
        |     +--:(auth-key-chain)
        |     |  +--rw key-chain?          key-chain:key-chain-ref
        |     +--:(auth-key-explicit)
        |        +--rw key?                string
        |        +--rw crypto-algorithm?   identityref
        +--rw inner-security-key-id?       uint8
        +--rw algorithm-type?              enumeration
        +--ro spf-statistics
        |  +--ro spf-statistics* [spf-direction-type]
        |     +--ro spf-direction-type    enumeration
        |     +--ro start-time?           yang:date-and-time
        |     +--ro end-time?             yang:date-and-time
        |     +--ro triggering-tie
        |        +--ro tie-direction-type?     enumeration
        |        +--ro originator?             system-id
        |        +--ro tie-type?               enumeration
        |        +--ro tie-number?             uint32
        |        +--ro seq?                    uint64
        |        +--ro size?                   uint32
        |        +--ro origination-time?       ieee802-1as-timestamp
        |        +--ro origination-lifetime?   uint32
        |        +--ro remaining-lifetime?     uint32
        +--ro hal
        |  +--ro hal-value?    level
```

```
|  +--ro system-ids*   system-id
+--ro miscabled-links*              uint32
+--rw hop-limit?                    uint8
+--rw maximum-clock-delta?          ieee802-1as-timestamp
+--ro total-num-routes-north?       uint32
+--ro total-num-routes-sourth?      uint32
+--rw interfaces* [name]
|  +--ro link-id?                          uint32
|  +--rw name                              if:interface-ref
|  +--rw cost?                             uint32
|  +--rw rx-flood-port?                    inet:port-number
|  +--rw holdtime?
|  |       rt-types:timer-value-seconds16
|  +--rw address-families*
|  |       iana-rt-types:address-family
|  +--rw advertised-source-addr
|  |  +--rw ipv4?   inet:ipv4-address-no-zone
|  |  +--rw ipv6?   inet:ipv6-address-no-zone
|  +--ro link-direction-type?              enumeration
|  +--rw broadcast-capable?                boolean
|  +--rw allow-horizontal-link?            boolean
|  +--rw security {link-security}?
|  |  +--rw security-type?              enumeration
|  |  +--rw shared?                     boolean
|  |  +--rw (auth-key-chain)?
|  |     +--:(auth-key-chain)
|  |     |  +--rw key-chain?            key-chain:key-chain-ref
|  |     +--:(auth-key-explicit)
|  |        +--rw key?                  string
|  |        +--rw crypto-algorithm?     identityref
|  +--rw security-checking?                enumeration
|  +--ro was-the-last-lie-accepted?        boolean
|  +--ro last-lie-reject-reason?           string
|  +--ro advertised-in-lies
|  |  +--ro label?                            uint32
|  |  |       {label-switching}?
|  |  +--ro you-are-flood-repeater?           boolean
|  |  +--ro not-a-ztp-offer?                  boolean
|  |  +--ro you-are-sending-too-quickly?      boolean
|  +--rw link-capabilities
|  |  +--rw bfd-capable?             boolean
|  |  +--rw v4-forwarding-capable?   boolean
|  |  +--rw mtu-size?                uint32
|  +--ro state                             enumeration
|  +--ro number-of-flaps?                  uint32
|  +--ro last-state-change?                yang:date-and-time
|  +--ro last-up?                          yang:date-and-time
|  +--ro last-down?                        yang:date-and-time
```

```
|  +--ro interface-states-statistics
|  |  +--ro intf-states-startup-time?   uint64
|  |  +--ro num-of-nbrs-3way?           uint32
|  |  +--ro num-of-nbrs-down?           uint32
|  |  +--ro nbrs-down-reasons* [system-id]
|  |  |  +--ro system-id           system-id
|  |  |  +--ro last-down-reason?   string
|  |  +--ro num-local-level-change?     uint32
|  |  +--ro intf-lie-states
|  |     +--ro last-lie-sent-time?              uint64
|  |     +--ro last-lie-received-time?          uint64
|  |     +--ro num-lie-received?                uint32
|  |     +--ro num-lie-transmitted?             uint32
|  |     +--ro num-lie-drop-invalid-envelope?   uint32
|  |     +--ro num-lie-drop-invalid-nonce?      uint32
|  |     +--ro num-lie-corrupted?               uint32
|  +--ro flood-repeater-statistics
|  |  +--ro flood-repeater?                       system-id
|  |  +--ro num-flood-repeater-changes?           uint32
|  |  +--ro last-flood-repeater-change-reason?   string
|  +--ro neighbors* [system-id]
|     +--ro node-level?                   level
|     +--ro system-id                     system-id
|     +--ro fabric-id?                    uint16
|     +--ro pod?                          uint32
|     +--ro proto-major-ver?              uint8
|     +--ro proto-minor-ver?              uint16
|     +--ro sent-offer
|     |  +--ro level?             level
|     |  +--ro not-a-ztp-offer?   boolean
|     +--ro received-offer
|     |  +--ro level?                          level
|     |  +--ro not-a-ztp-offer?                boolean
|     |  +--ro best?                           boolean
|     |  +--ro removed-from-consideration?   boolean
|     |  +--ro removal-reason?                 string
|     +--ro received-source-addr
|     |  +--ro ipv4?   inet:ipv4-address-no-zone
|     |  +--ro ipv6?   inet:ipv6-address-no-zone
|     +--ro link-id-pair* [remote-id]
|     |  +--ro local-id?           uint32
|     |  +--ro remote-id           uint32
|     |  +--ro if-index?           uint32
|     |  +--ro if-name?            if:interface-ref
|     |  +--ro address-families*
|     |           iana-rt-types:address-family
|     +--ro cost?                         uint32
|     +--ro bandwidth?                    uint32
```

```
|      +--ro received-link-capabilities
|      |  +--ro bfd-capable?             boolean
|      |  +--ro v4-forwarding-capable?   boolean
|      |  +--ro mtu-size?                uint32
|      +--ro received-in-lies
|      |  +--ro label?                          uint32
|      |  |       {label-switching}?
|      |  +--ro you-are-flood-repeater?         boolean
|      |  +--ro not-a-ztp-offer?                boolean
|      |  +--ro you-are-sending-too-quickly?    boolean
|      +--ro nbr-flood-port?                inet:port-number
|      +--ro tx-flood-port?                 inet:port-number
|      +--ro bfd-state?                     enumeration
|      +--ro outer-security-key-id?         uint8
|      +--ro local-nonce?                   uint16
|      +--ro remote-nonce?                  uint16
|      +--ro tie-state-statistics
|         +--ro transmit-queue?                    uint32
|         +--ro last-queued-tie
|         |  +--ro tie-direction-type?     enumeration
|         |  +--ro originator?             system-id
|         |  +--ro tie-type?               enumeration
|         |  +--ro tie-number?             uint32
|         |  +--ro seq?                    uint64
|         |  +--ro size?                   uint32
|         |  +--ro origination-time?
|         |  |       ieee802-1as-timestamp
|         |  +--ro origination-lifetime?   uint32
|         |  +--ro remaining-lifetime?     uint32
|         |  +--ro reason-queued?          string
|         +--ro num-received-ties?                 uint32
|         +--ro num-transmitted-ties?              uint32
|         +--ro num-retransmitted-ties?            uint32
|         +--ro num-flood-reduced-ties?            uint32
|         +--ro num-received-tides?                uint32
|         +--ro num-transmitted-tides?             uint32
|         +--ro num-received-tires?                uint32
|         +--ro num-transmitted-tires?             uint32
|         +--ro num-request-locally?               uint32
|         +--ro num-request-remotely?              uint32
|         +--ro num-same-older-ties-received?      uint32
|         +--ro num-seq-mismatch-pkts-received?    uint32
|         +--ro last-sent-tie
|         |  +--ro tie-direction-type?     enumeration
|         |  +--ro originator?             system-id
|         |  +--ro tie-type?               enumeration
|         |  +--ro tie-number?             uint32
|         |  +--ro seq?                    uint64
```

```
|          |  +--ro size?                    uint32
|          |  +--ro origination-time?
|          |  |       ieee802-1as-timestamp
|          |  +--ro origination-lifetime?   uint32
|          |  +--ro remaining-lifetime?     uint32
|          |  +--ro last-tie-sent-time?     yang:date-and-time
|          +--ro last-recv-tie
|          |  +--ro tie-direction-type?     enumeration
|          |  +--ro originator?             system-id
|          |  +--ro tie-type?               enumeration
|          |  +--ro tie-number?             uint32
|          |  +--ro seq?                    uint64
|          |  +--ro size?                   uint32
|          |  +--ro origination-time?
|          |  |       ieee802-1as-timestamp
|          |  +--ro origination-lifetime?   uint32
|          |  +--ro remaining-lifetime?     uint32
|          |  +--ro last-tie-recv-time?     yang:date-and-time
|          +--ro largest-tie
|          |  +--ro largest-tie-sent
|          |  |  +--ro tie-direction-type?     enumeration
|          |  |  +--ro originator?             system-id
|          |  |  +--ro tie-type?               enumeration
|          |  |  +--ro tie-number?             uint32
|          |  |  +--ro seq?                    uint64
|          |  |  +--ro size?                   uint32
|          |  |  +--ro origination-time?
|          |  |  |       ieee802-1as-timestamp
|          |  |  +--ro origination-lifetime?   uint32
|          |  |  +--ro remaining-lifetime?     uint32
|          |  +--ro largest-tide-sent
|          |  |  +--ro tie-direction-type?     enumeration
|          |  |  +--ro originator?             system-id
|          |  |  +--ro tie-type?               enumeration
|          |  |  +--ro tie-number?             uint32
|          |  |  +--ro seq?                    uint64
|          |  |  +--ro size?                   uint32
|          |  |  +--ro origination-time?
|          |  |  |       ieee802-1as-timestamp
|          |  |  +--ro origination-lifetime?   uint32
|          |  |  +--ro remaining-lifetime?     uint32
|          |  +--ro largest-tire-sent
|          |     +--ro tie-direction-type?     enumeration
|          |     +--ro originator?             system-id
|          |     +--ro tie-type?               enumeration
|          |     +--ro tie-number?             uint32
|          |     +--ro seq?                    uint64
|          |     +--ro size?                   uint32
```

```
|          |     +--ro origination-time?
|          |     |       ieee802-1as-timestamp
|          |     +--ro origination-lifetime?   uint32
|          |     +--ro remaining-lifetime?     uint32
|          +--ro num-tie-dropped
|             +--ro num-tie-outer-envelope?   uint32
|             +--ro num-tie-inner-envelope?   uint32
|             +--ro num-tie-nonce?            uint32
|
+--ro database
   +--ro ties*
           [tie-direction-type originator tie-type tie-number]
      +--ro tie-direction-type      enumeration
      +--ro originator              system-id
      +--ro tie-type                enumeration
      +--ro tie-number              uint32
      +--ro seq?                    uint64
      +--ro size?                   uint32
      +--ro origination-time?       ieee802-1as-timestamp
      +--ro origination-lifetime?   uint32
      +--ro remaining-lifetime?     uint32
      +--ro node
      |  +--ro level?                   level
      |  +--ro neighbors* [system-id]
      |  |  +--ro node-level?                   level
      |  |  +--ro system-id                     system-id
      |  |  +--ro fabric-id?                    uint16
      |  |  +--ro pod?                          uint32
      |  |  +--ro link-id-pair* [remote-id]
      |  |  |  +--ro local-id?           uint32
      |  |  |  +--ro remote-id           uint32
      |  |  |  +--ro if-index?           uint32
      |  |  |  +--ro if-name?            if:interface-ref
      |  |  |  +--ro address-families*
      |  |  |           iana-rt-types:address-family
      |  |  +--ro cost?                         uint32
      |  |  +--ro bandwidth?                    uint32
      |  |  +--ro received-link-capabilities
      |  |     +--ro bfd-capable?             boolean
      |  |     +--ro v4-forwarding-capable?   boolean
      |  |     +--ro mtu-size?                uint32
      |  +--ro proto-minor-ver?         uint16
      |  +--ro flood-reduction?         boolean
      |  +--ro hierarchy-indications
      |  |  +--ro hierarchy-indications?   enumeration
      |  +--ro overload-flag?           boolean
      |  +--ro name?                    string
      |  +--ro pod?                     uint32
      |  +--ro startup-time?            uint64
```

```
               |  +--ro miscabled-links*         uint32
               |  +--ro same-plane-tofs*         system-id
               |  +--ro fabric-id?               uint32
               +--ro prefixes
               |  +--ro prefixes* [prefix]
               |     +--ro prefix               inet:ip-prefix
               |     +--ro tie-type?            enumeration
               |     +--ro metric?              uint32
               |     +--ro tags*                uint64
               |     +--ro monotonic-clock
               |     |  +--ro prefix-sequence-type
               |     |     +--ro timestamp
               |     |     |       ieee802-1as-timestamp
               |     |     +--ro transaction-id?   uint8
               |     +--ro loopback?            boolean
               |     +--ro directly-attached?   boolean
               |     +--ro from-link?           uint32
               |     +--ro label?               uint32
               +--ro key-value
                  +--ro key?     binary
                  +--ro value?   binary

notifications:
  +---n error-set
     +--ro tie-level-error
     |  +--ro rift* [name]
     |     +--ro name    string
     |     +--ro ties* [originator]
     |        +--ro tie-direction-type?     enumeration
     |        +--ro originator              system-id
     |        +--ro tie-type?               enumeration
     |        +--ro tie-number?             uint32
     |        +--ro seq?                    uint64
     |        +--ro size?                   uint32
     |        +--ro origination-time?       ieee802-1as-timestamp
     |        +--ro origination-lifetime?   uint32
     |        +--ro remaining-lifetime?     uint32
     +--ro neighbor-error
        +--ro rift* [name]
           +--ro name         string
           +--ro neighbors* [system-id]
              +--ro node-level?                   level
              +--ro system-id                     system-id
              +--ro fabric-id?                    uint16
              +--ro pod?                          uint32
              +--ro proto-major-ver?              uint8
              +--ro proto-minor-ver?              uint16
              +--ro sent-offer
```

```
|  +--ro level?             level
|  +--ro not-a-ztp-offer?   boolean
+--ro received-offer
|  +--ro level?                          level
|  +--ro not-a-ztp-offer?                boolean
|  +--ro best?                           boolean
|  +--ro removed-from-consideration?     boolean
|  +--ro removal-reason?                 string
+--ro received-source-addr
|  +--ro ipv4?   inet:ipv4-address-no-zone
|  +--ro ipv6?   inet:ipv6-address-no-zone
+--ro link-id-pair* [remote-id]
|  +--ro local-id?           uint32
|  +--ro remote-id           uint32
|  +--ro if-index?           uint32
|  +--ro if-name?            if:interface-ref
|  +--ro address-families*
|           iana-rt-types:address-family
+--ro cost?                          uint32
+--ro bandwidth?                     uint32
+--ro received-link-capabilities
|  +--ro bfd-capable?              boolean
|  +--ro v4-forwarding-capable?    boolean
|  +--ro mtu-size?                 uint32
+--ro received-in-lies
|  +--ro label?                           uint32
|  |       {label-switching}?
|  +--ro you-are-flood-repeater?          boolean
|  +--ro not-a-ztp-offer?                 boolean
|  +--ro you-are-sending-too-quickly?     boolean
+--ro nbr-flood-port?                inet:port-number
+--ro tx-flood-port?                 inet:port-number
+--ro bfd-state?                     enumeration
+--ro outer-security-key-id?         uint8
```

2.4. RIFT configuration

The configuration data nodes cover node configuration attributes. RIFT configurations require node base information configurations. Some features can be used to enhance protocol, such as BFD [RFC5881], flooding-reducing, community attribute.

2.5. RIFT State

The state data nodes include node, neighbor, database and kv-store information.

2.6. Notifications

Unexpected TIE and neighbor's layer error should be notified.

3. RIFT YANG model

This module references [I-D.ietf-rift-rift], [RFC5881], [RFC6991], [RFC8177], [RFC8294], [RFC8343], [RFC8349], [RFC8505], [IEEE8021AS].

<CODE BEGINS> file "ietf-rift@2024-07-23.yang"

```
module ietf-rift {

  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-rift";
  prefix rift;

  import ietf-inet-types {
    prefix "inet";
    reference
      "RFC 6991: Common YANG Data Types";
  }

  import ietf-yang-types {
    prefix "yang";
    reference
      "RFC 6991: Common YANG Data Types";
  }

  import ietf-routing {
    prefix "rt";
    reference
      "RFC 8349: A YANG Data Model for Routing Management
       (NMDA Version)";
  }

  import ietf-interfaces {
    prefix "if";
    reference
      "RFC 8343: A YANG Data Model for Interface Management";
  }

  import ietf-routing-types {
    prefix "rt-types";
    reference
      "RFC 8294: Common YANG Data Types for the Routing Area";
  }
```

```
  import iana-routing-types {
    prefix "iana-rt-types";
    reference
      "RFC 8294: Common YANG Data Types for the Routing Area";
  }

  import ietf-key-chain {
    prefix "key-chain";
    reference
      "RFC 8177: YANG Data Model for Key Chains";
  }

  organization
    "IETF RIFT (Routing In Fat Trees) Working Group";

  contact
    "WG Web:   <https://datatracker.ietf.org/wg/rift/>
     WG List:  <mailto:rift@ietf.org>

     Editor:   Zheng Zhang
               <mailto:zhang.zheng@zte.com.cn>

     Editor:   Yuehua Wei
               <mailto:wei.yuehua@zte.com.cn>

     Editor:   Shaowen Ma
               <mailto:mashaowen@gmail.com>

     Editor:   Xufeng Liu
               <mailto:xufeng.liu.ietf@gmail.com>

     Editor:   Bruno Rijsman
               <mailto:brunorijsman@gmail.com>";

  // RFC Ed.: replace XXXX with actual RFC number and remove
  // this note

  description
    "This YANG module defines the generic configuration and
     operational state for the RIFT protocol common to all
     vendor implementations.  It is intended that the module
     will be extended by vendors to define vendor-specific
     RIFT configuration parameters and policies --
     for example, route maps or route policies.

     This YANG data model conforms to the Network Management
     Datastore Architecture (NMDA) as described in RFC 8342.
```

```
      The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
      'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
      'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
      are to be interpreted as described in BCP 14 (RFC 2119)
      (RFC 8174) when, and only when, they appear in all
      capitals, as shown here.

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

      Redistribution and use in source and binary forms, with or
      without modification, is permitted pursuant to, and subject
      to the license terms contained in, the Revised BSD License
      set forth in Section 4.c of the IETF Trust's Legal
      Provisions Relating to IETF Documents
      (https://trustee.ietf.org/license-info).

      This version of this YANG module is part of RFC XXXX; see
      the RFC itself for full legal notices.";

    revision 2024-07-23 {
      description
        "Initial revision.";
      reference
        "RFCXXXX: YANG Data Model for Routing in Fat Trees
        (RIFT).";
    }

    /*
     * Features
     */

    feature nonce-delta-adjust {
      description
        "Support weak nonce delta adjusting which is used in
         security.";
      reference
        "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
         Section 6.9.";
    }

    feature label-switching {
      description
        "Support label switching for instance distinguishing.";
      reference
        "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
         Section 6.8.8";
    }
```

```
    feature tie-security {
      description
        "Support security function for the TIE exchange.";
      reference
        "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
         Section 6.9.3.";
    }

    feature link-security {
      description
        "Support security function of link.";
      reference
        "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
         Section 6.9.";
    }

    typedef system-id {
      type string {
        pattern
  '[0-9A-Fa-f]{4}\.[0-9A-Fa-f]{4}\.[0-9A-Fa-f]{4}\.[0-9A-Fa-f]{4}';
      }
      description
        "This type defines RIFT system id using pattern,
         the system id looks like: 0021.2FFF.FEB5.6E10";
    }

    typedef level {
      type uint8 {
        range "0 .. 24";
      }
      default "0";
      description
        "The value of node level.
        Clos and Fat Tree networks are topologically partially
        ordered graphs and 'level' denotes the set of nodes at
        the same height in such a network.
        Nodes at the top level (i.e., ToF) are at the level with
        the highest value and count down to the nodes
        at the bottom level (i.e., leaf) with the lowest value.
        In RIFT, Level 0 always indicates that a node is a leaf,
        but does not have to be level 0.
        Level values can be configured manually or automatically
        derived.";
      reference
        "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
         Section 6.7.";
    }
```

```
  typedef ieee802-1as-timestamp {
    type uint64;
    units "seconds";
    description
      "Timestamp per IEEE802.1AS. It is advertised with prefix
       to achieve mobility.";
    reference
      "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees. Section 6.8.4.
       IEEE8021AS: Timing and Synchronization for Time-Sensitive
                   Applications in Bridged Local Area Networks";
  }

  /*
   * Identity
   */
  identity rift {
    base rt:routing-protocol;
    description
      "Identity for the RIFT routing protocol.";
    reference
      "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees";
  }

  /*
   * Groupings
   */

  grouping address-families {
    leaf-list address-families {
      type iana-rt-types:address-family;
      description
        "Indication which address families are up on the
         interface.";
    }
    description
      "Containing address families on the interface.";
  }

  grouping hierarchy-indications {
    leaf hierarchy-indications {
      type enumeration {
        enum "leaf-only" {
          description
            "The node will never leave the
             'bottom of the hierarchy'.
             When this value is set, the 'configured-level'
             is the minimum level value.";
        }
```

```
            enum "leaf-only-and-leaf-2-leaf-procedures" {
              description
                "This means leaf to leaf.
                 When this value is set, the 'configured-level'
                 is the minimum level value.";
            }
            enum "top-of-fabric" {
              description
                "The node is 'top of fabric'.
                 When this value is set, the 'configured-level'
                 is the maximum level value.";
            }
          }
          description
            "The hierarchy indications of this node.";
        }
        description
          "Flags indicating node configuration in case of ZTP";
      }

      grouping node-capability {
        leaf proto-minor-ver {
          type uint16;
          description
            "Represents the minor protocol encoding schema
             version of this node.";
        }
        leaf flood-reduction {
          type boolean;
          description
            "If the value is set to 'true', it means that
             this node enables the flood reduction function.";
        }
        container hierarchy-indications {
          config false;
          description
            "The hierarchy-indications of the node.";
          uses hierarchy-indications;
        }
        description
          "The supported capabilities of this node.";
      }

      grouping tie-type {
        leaf tie-type {
          type enumeration {
            enum "illegal" {
              description
```

```
            "The illegal TIE.";
          }
          enum "min-tie-type" {
            description
              "The minimum TIE.";
          }
          enum "node" {
            description
              "The node TIE.";
          }
          enum "prefix" {
            description
              "The prefix TIE.";
          }
          enum "positive-disaggregation-prefix" {
            description
              "The positive disaggregation prefix TIE.";
          }
          enum "negative-disaggregation-prefix" {
            description
              "The negative disaggregation prefix TIE.";
          }
          enum "pgp-prefix" {
            description
              "The policy guide prefix TIE.";
          }
          enum "key-value" {
            description
              "The key value TIE.";
          }
          enum "external-prefix" {
            description
              "The external prefix TIE.";
          }
          enum "positive-external-disaggregation-prefix" {
            description
              "The positive external disaggregation prefix TIE.";
          }
          enum "max-tie-type" {
            description
              "The maximum TIE.";
          }
        }
        description
          "The types of TIE.";
      }
      description
        "The types of TIE";
```

```
    }

    grouping prefix-attribute {

      leaf metric {
        type uint32;
        description
          "The metric of this prefix.";
      }
      leaf-list tags {
        type uint64;
        description
          "The tags of this prefix.";
      }
      container monotonic-clock {
        container prefix-sequence-type {
          leaf timestamp {
            type ieee802-1as-timestamp;
            mandatory true;
            description
              "The timestamp per 802.1AS can be advertised
               with the desired prefix North TIEs.";
          }
          leaf transaction-id {
            type uint8;
            description
              "As per RFC 8505, a sequence number called a
               Transaction ID (TID) with a prefix can be
               advertised.";
            reference
              "RFC 8505: Registration Extensions for IPv6 over
               Low-Power Wireless Personal Area Network (6LoWPAN)
               Neighbor Discovery";
          }
          description
            "The prefix sequence attribute which can be advertised
             for mobility.";
          reference
            "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
             Section 6.8.4.";
        }
        description
          "The monotonic clock for mobile addresses.";
      }
      leaf loopback {
        type boolean;
        description
          "If the value is set to 'true', it
```

```
            indicates if the interface is a node loopback.
            The node's loopback address can be injected into
            North and South Prefix TIEs for node reachability.";
          reference
            "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
            Section 6.4.";
        }
        leaf directly-attached {
          type boolean;
          description
            "If the value is set to 'true', it indicates that the
            prefix is directly attached, i.e. should be routed to
            even if the node is in overload.";
        }
        leaf from-link {
          type uint32;
          description
            "In case of locally originated prefixes,
            i.e. interface addresses this can describe which
            link the address belongs to.";
        }
        leaf label {
          type uint32;
          description
            "Per prefix significant label.";
          reference
            "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees";
        }
        description
          "The attributes of the prefix.";
      }

      grouping security {
        leaf security-type {
          type enumeration {
            enum public {
              description
                "When using PKI  (Public Key Infrastructure),
                the public and shared key can be used to verify
                the original packet exchanged with the neighbor.";
            }
            enum private {
              description
                "When using PKI (Public Key Infrastructure),
                the private key can be used by the Security
                fingerprint originating node to create the signature.";
            }
          }
```

```
        description
          "The security type.";
        reference
          "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
           Section 6.9.";
      }
      leaf shared {
        type boolean;
          description
            "When using PKI (Public Key Infrastructure),
             if the key is shared.";
          reference
          "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
           Section 6.9.";
      }

      choice auth-key-chain {
        description
          "Key chain or explicit key parameter specification";
        case auth-key-chain {
          leaf key-chain {
            type key-chain:key-chain-ref;
            description
              "key-chain name.";
            reference
              "RFC 8177: YANG Data Model for Key Chains";
          }
        }
        case auth-key-explicit {
          leaf key {
            type string;
            description
              "Authentication key. The length of the key may be
               dependent on the cryptographic algorithm.";
          }
          leaf crypto-algorithm {
            type identityref {
              base key-chain:crypto-algorithm;
            }
            description
              "Cryptographic algorithm associated with key.";
            reference
              "RFC 8177: YANG Data Model for Key Chains";
          }
        }
      }
      description
        "The security parameters.";
```

```
    }

    grouping base-node-info {
      leaf node-level {
        type level;
        config false;
        description
          "The level of this node.";
      }
      leaf system-id {
        type system-id;
        mandatory true;
        description
          "Each node is identified via a system-id which is 64
           bits wide.";
      }
      leaf fabric-id {
        type uint16;
        description
          "The optional id of the fabric.";
      }
      leaf pod {
        type uint32 {
          range "1..max";
        }
        description
          "The identifier of the Point of Delivery (PoD).
           A PoD is the self-contained vertical slice of a
           Clos or Fat Tree network containing normally only leaf
           nodes (level 0) and their immediate northbound
           neighbors. It communicates with nodes
           in other PoDs via the spine. Making this leaf
           unspecified indicates that the PoD is 'undefined'.";
      }
      description
        "The base information of a node.";
    } // base-node-info

    grouping link-capabilities {
      leaf bfd-capable {
        type boolean;
        default "true";
        description
          "If this value is set to 'true', it means that
           BFD function is enabled on the neighbor.";
        reference
          "RFC 5881: Bidirectional Forwarding Detection (BFD)
           for IPv4 and IPv6 (Single Hop)";
```

```
      }
      leaf v4-forwarding-capable {
        type boolean;
        default "true";
        description
          "If this value is set to 'true', it means that
           the neighbor supports v4 forwarding.";
      }
      leaf mtu-size {
        type uint32;
        default "1400";
        description
          "MTU of the link.";
      }
      description
        "The features of neighbor.";
    } // link-capabilities

    grouping addresses {
      leaf ipv4 {
        type inet:ipv4-address-no-zone;
        description
          "IPv4 address to be used.";
      }
      leaf ipv6 {
        type inet:ipv6-address-no-zone;
        description
          "IPv6 address to be used.";
      }
      description
        "IPv4 and/or IPv6 address to be used.";
    }

    grouping lie-elements {
      leaf label {
        if-feature label-switching;
        type uint32;
        description
          "A locally significant, downstream assigned by
           the neighbor, interface specific label which may
           be advertised in its LIEs.";
        reference
          "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
           Section 6.8.8.";
      }
      leaf you-are-flood-repeater {
        type boolean;
        description
```

```
          "If the neighbor on this link is flooding repeater.
           When this value is set to 'true', the value can be
           carried in exchanged packet.";
        reference
          "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
           Section 6.3.9.";
      }
      leaf not-a-ztp-offer {
        type boolean;
        description
          "When this value is set to 'true', the flag can be
           carried in the LIE packet. When the value received
           in the LIE from neighbor, it indicates the level on
           the LIE MUST NOT be used to derive a ZTP level by
           the receiving node.";
        reference
          "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
           Section 6.7.";
      }
      leaf you-are-sending-too-quickly {
        type boolean;
        description
          "Can be optionally set to indicate to neighbor that
           packet losses are seen on reception based on packet
           numbers or the rate is too high. The receiver SHOULD
           temporarily slow down flooding rates. When this value
           is set to 'true', the flag can be carried in packet.";
      }
      description
        "The elements set in the LIEs.";
    } // lie-elements

    grouping link-id-pair {
      leaf local-id {
        type uint32;
        description
          "The local-id of link connect to this neighbor.";
      }
      leaf remote-id {
        type uint32;
        description
          "The remote-id to reach this neighbor.";
      }
      leaf if-index {
        type uint32;
        description
          "The local index of this interface.";
      }
```

```
      leaf if-name {
        type if:interface-ref;
        description
          "The name of this interface.";
      }
      uses address-families;
      description
        "A pair of local and remote link-id to identify a link
         between two nodes.";
    } // link-id-pair

    grouping neighbor-node {
      list link-id-pair {
        key "remote-id";
        uses link-id-pair;
        description
          "The Multiple parallel links to this neighbor.";
      }
      leaf cost {
        type uint32;
        description
          "The cost value advertised by the neighbor.";
      }
      leaf bandwidth {
        type uint32;
        units "bits";
        description
          "Total bandwith to the neighbor, this will be
           normally sum of the bandwidths of all the
           parallel links.";
      }
      container received-link-capabilities {
        uses link-capabilities;
        description
          "The link capabilities advertised by the neighbor.";
      }
      description
        "The neighbor information indicated in node TIE.";
    } // neighbor-node

    grouping neighbor {
      leaf proto-major-ver {
        type uint8;
        description
          "Represents protocol encoding schema major version of
           this neighbor.";
      }
      leaf proto-minor-ver {
```

```
        type uint16;
        description
          "Represents protocol encoding schema minor version of
           this neighbor.";
      }
      container sent-offer {
        leaf level {
          type level;
          description
            "The level value.";
        }
        leaf not-a-ztp-offer {
          type boolean;
          description
            "If the value is set to 'true', it indicates the
             level on the LIE MUST NOT be used to derive a
             ZTP level by the neighbor.";
        }
        description
          "The level sent to the neighbor in case the neighbor
           needs to be offered.";
      }
      container received-offer {
        leaf level {
          type level;
          description
            "The level value.";
        }
        leaf not-a-ztp-offer {
          type boolean;
          description
            "If the value is set to 'true', it indicates the
             level on the received LIE MUST NOT be used to
             derive a ZTP level.";
        }
        leaf best {
          type boolean;
          description
            "If the value is set to 'true', it means that
             the level is the best level received from all
             the neighbors.";
        }
        leaf removed-from-consideration {
          type boolean;
          description
            "If the value is set to 'true', it means that
             the level value is not considered to be used.";
        }
```

```
        leaf removal-reason {
          when "../removed-from-consideration='true'" {
             description
               "The level value is not considered to be used.";
           }
          type string;
          description
            "The reason why this value is not considered to
             be used.";
        }
        description
          "The level offered to the interface from the neighbor.
           And if the level value is considered to be used.";
      }
      container received-source-addr {
        uses addresses;
        description
          "The source address of LIE and TIE packets from
           the neighbor.";
      } // received-offer
      uses neighbor-node;
      container received-in-lies {
        uses lie-elements;
        description
          "The attributes received from this neighbor.";
      }
      leaf nbr-flood-port {
        type inet:port-number;
        default "915";
        description
          "The UDP port which is used by the neighbor to flood
           TIEs.";
      }
      leaf tx-flood-port {
        type inet:port-number;
        default "915";
        description
          "The UDP port which is used by the node to flood
           TIEs to the neighbor.";
      }
      leaf bfd-state {
        type enumeration {
          enum up {
            description
              "The link is protected by established BFD session.";
          }
          enum down {
            description
```

```
              "The link is not protected by established BFD session.";
          }
        }
        description
          "The link is protected by established BFD session or not.";
      }
      leaf outer-security-key-id {
        type uint8;
        description
          "The received security key id from the neighbor.";
        reference
          "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
           Section 6.9.3.";
      }
      description
        "The neighbor information.";
    } // neighbor

    grouping link-direction-type {
      leaf link-direction-type {
        type enumeration {
          enum illegal {
            description
              "Illegal direction.";
          }
          enum south {
            description
              "A link to a node one level down.";
          }
          enum north {
            description
              "A link to a node one level up.";
          }
          enum east-west {
            description
              "A link to a node in the same level.";
          }
          enum max {
            description
              "The max value of direction.";
          }
        }
        config false;
        description
          "The type of a link.";
      }
      description
        "The type of a link.";
```

```
  } // link-direction-type

  grouping tie-direction-type {
    leaf tie-direction-type {
      type enumeration {
        enum illegal {
          description
            "Illegal direction.";
        }
        enum south {
          description
            "The direction to a node one level down.";
        }
        enum north {
          description
            "The direction to a node one level up.";
        }
        enum max {
          description
            "The max value of direction.";
        }
      }
      config false;
      description
        "The direction type of a TIE.";
    }
    description
      "The direction type of a TIE.";
  } // tie-direction-type

  grouping spf-direction-type {
    leaf spf-direction-type {
      type enumeration {
        enum n-spf {
          description
            "A reachability calculation that is progressing
            northbound, as example SPF that is using South
            Node TIEs only. Normally it progresses a single
            hop only and installs default routes.";
        }
        enum s-spf {
          description
            "A reachability calculation that is progressing
             southbound, as example SPF that is using North
             Node TIEs only.";
        }
      }
      config false;
```

```
        description
          "The direction type of a SPF calculation.";
      }
      description
        "The direction type of a SPF calculation.";
    } // spf-direction-type

    grouping tie-header {
      uses tie-direction-type;
      leaf originator {
        type system-id;
        description
          "The originator's system-id of this TIE.";
      }

      uses tie-type;

      leaf tie-number {
        type uint32;
        description
          "The number of this TIE";
      }

      leaf seq {
        type uint64;
        description
          "The sequence number of a TIE.";
        reference
          "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
          Section 6.3.1.";
      }
      leaf size {
        type uint32;
        description
          "The size of this TIE.";
      }
      leaf origination-time {
        type ieee802-1as-timestamp;
        description
          "Absolute timestamp when the TIE was generated.
           This can be used on fabrics with synchronized
           clock to prevent lifetime modification attacks.";
      }
      leaf origination-lifetime {
        type uint32;
        units seconds;
        description
          "Original lifetime when the TIE was generated.
```

```
            This can be used on fabrics with synchronized clock
            to prevent lifetime modification attacks.";
        }
        leaf remaining-lifetime {
          type uint32;
          units seconds;
          description
            "The remaining lifetime of the TIE.";
        }

        description
          "TIEs are exchanged between RIFT nodes to describe parts
           of a network such as links and address prefixes.
           This is the TIE header information.";
      } // tie-header

      /*
       * Data nodes
       */
      augment "/rt:routing/rt:control-plane-protocols"
              + "/rt:control-plane-protocol" {
        when "derived-from-or-self(rt:type, 'rift:rift')" {
          description
            "This augment is only valid when routing protocol
             instance type is 'RIFT'.";
        }
        description
          "RIFT ( Routing in Fat Trees ) YANG model.";

        list rift {
          key "name";
          leaf name {
            type string;
            description
              "The RIFT instance's name.";
          }

          uses base-node-info;
          leaf fabric-prefix {
            type inet:ip-prefix;
            description
              "The configured fabric prefix.";
          }
          leaf fabric-prefix-advertise {
            type boolean;
            description
              "Whether the fabric-prefix can be advertised or not.
               If the value is set to 'true', it means that
```

```
           the fabric-prefix can be advertised to neighbors.";
        }
        leaf configured-level {
          type level;
          description
            "The configured level value of this node.";
        }
        container overload {
          description
            "If the overload in TIEs can be set
             and the timeout value with according type.";
          leaf overload {
            type boolean;
            description
              "If the value is set to 'true', it means that
               the overload bit in TIEs can be set.";
          }
          choice timeout-type {
            description
              "The value of timeout timer for overloading.
               This makes sense when overload is set to 'TRUE'.";
            case on-startup {
              leaf on-startup-timeout {
                type rt-types:timer-value-seconds16;
                description
                  "Node goes into overload until this timer
                   expires when starting up.";
              }
            }
            case immediate {
              leaf immediate-timeout {
                type rt-types:timer-value-seconds16;
                description
                  "Set overload and remove after the timeout
                   expired.";
              }
            }
          }
        }

        leaf proto-major-ver {
          type uint8;
          config false;
          mandatory true;
          description
            "Represents protocol encoding schema major version.";
        }
        leaf proto-minor-ver {
```

```
          type uint16;
          config false;
          mandatory true;
          description
            "Represents protocol encoding schema minor version.";
        }

        container node-capabilities {
          uses hierarchy-indications;
          leaf flood-reduction {
            type boolean;
            description
              "If the node supports flood reduction function.
               If this value is set to 'true',  it means that
               the flood reduction function is enabled.";
            reference
              "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
               Section 6.3.8.";
          }
          description
            "The node's capabilities.";
        }
        leaf maximum-nonce-delta {
          if-feature nonce-delta-adjust;
          type uint8 {
            range "1..5";
          }
          description
            "The configurable valid nonce delta value used for
             security. It is used as vulnerability window.
             If the nonces in received packet exceeds the range
             indicated by this value, the packet MUST be
             discarded.";
          reference
            "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
            Section 6.9.4.";
        }

        leaf nonce-increasing-interval {
          type uint16;
          units seconds;
          description
            "The configurable nonce increasing interval.";
        }

        leaf adjusted-lifetime {
          type rt-types:timer-value-seconds16;
          units seconds;
```

```
          description
            "The adjusted lifetime may affect the TIE stability.
             Be careful to change this parameter.
             This should be prohibited less than 2*purge-lifetime.";
        }
        container rx-lie-multicast-addr {
          leaf ipv4 {
            type inet:ipv4-address;
            default "224.0.0.121";
            description
              "The configurable LIE receiving IPv4 multicast
               address.
               Different multicast addresses can be used for
               receiving and sending.";
          }
          leaf ipv6 {
            type inet:ipv6-address;
            default "ff02::a1f7";
            description
              "The configurable LIE receiving IPv6 multicast
               address.
               Different multicast addresses can be used for
               receiving and sending.";
          }
          description
            "The configurable LIE receiving IPv4/IPv6 multicast
             address.
             Different multicast addresses can be used for
             receiving and sending.";
        }
        container tx-lie-multicast-addr {
          leaf ipv4 {
            type inet:ipv4-address;
            description
              "The configurable LIE sending IPv4 multicast
               address.
               Different multicast addresses can be used for
               receiving and sending.";
          }
          leaf ipv6 {
            type inet:ipv6-address;
            description
              "The configurable LIE sending IPv6 multicast
               address.
               Different multicast addresses can be used for
               receiving and sending.";
          }
          description
```

```
            "The configurable LIE sending IPv4/IPv6 multicast
             address.
             Different multicast addresses can be used for
             receiving and sending.";
          }
          leaf lie-tx-port {
            type inet:port-number;
            default "914";
            description
              "The UDP port of LIE packet sending. The default port
               number is 914. The value can be set to other value
               associated with different RIFT instance.";
          }

          container global-link-capabilities {
            uses link-capabilities;
            description
              "The node default link capabilities. It can be
               overwrite by the configuration underneath interface
               and neighbor.";
          }

          leaf tide-generation-interval {
            type rt-types:timer-value-seconds16;
            units seconds;
            description
              "The TIDE generation interval.";
          }

          list tie-security {
            if-feature tie-security;
            key "security-type";
            uses security;
            description
              "The security function used for the TIE exchange.";
            reference
              "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
               Section 6.9.3.";
          }

          leaf inner-security-key-id {
            type uint8;
            description
              "The inner security key id for received packet checking.";
            reference
              "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
             Section 6.9.3.";
          }
```

```
        leaf algorithm-type {
          type enumeration {
            enum spf {
              description
                "The algorithm is SPF.";
            }
            enum all-path {
              description
                "The algorithm is all-path.";
            }
          }
          description
            "The possible algorithm types.";
        }
        container spf-statistics {
          config false;
          list spf-statistics {
            key "spf-direction-type";
            description
              "The statistics of SPF calculation.";
            uses spf-direction-type;

            leaf start-time {
              type yang:date-and-time;
              description
                "The last SPF calculation start time.";
            }
            leaf end-time {
              type yang:date-and-time;
              description
                "The last SPF calculation end time.";
            }
            container triggering-tie {
              uses tie-header;
              description
                "The TIE that triggered the SPF.";
            }
          }
          description
            "The statistics of SPF calculation.";
        }

        container hal {
          config false;
          leaf hal-value {
            type level;
            description
              "The highest defined level value seen from all
```

```
            valid level offers received.";
        }
        leaf-list system-ids{
          type system-id;
          description
            "The node's system-id of the offered level comes
             from.";
        }
        description
          "The highest defined level and the offered nodes set.";
      }

      leaf-list miscabled-links {
        type uint32;
        config false;
        description
          "List of miscabled links.";
      }

      leaf hop-limit {
        type uint8 {
          range "1 | 255";
        }
        default "1";
        description
          "The IPv4 TTL or IPv6 HL used for LIE and TIE
           sending/receiving.";
      }

      leaf maximum-clock-delta {
        type ieee802-1as-timestamp;
        description
          "The maximum drift for the timestamp comparing.";
        reference
          "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
           Section 6.8.4.";
      }

      leaf total-num-routes-north {
        type uint32;
        config false;
        description
          "The total number of north routes.";
      }
      leaf total-num-routes-sourth {
        type uint32;
        config false;
        description
```

```
            "The total number of sourth routes.";
          }

          list interfaces {
            key "name";
            leaf link-id {
              type uint32;
              config false;
              description
                "The local id of this interface.";
            }
            leaf name {
              type if:interface-ref;
              description
                "The interface's name.";
            }
            leaf cost {
              type uint32;
              description
                "The cost from this interface to the neighbor.";
            }
            leaf rx-flood-port {
              type inet:port-number;
              default "915";
              description
                "The UDP port which is used to receive flooded
                 TIEs. The default port number is 915. The value can
                 be set to other value associated with different
                 RIFT instance.";
            }
            leaf holdtime {
              type rt-types:timer-value-seconds16;
              units seconds;
              default "3";
              description
                "The holding time of LIE.";
            }

            uses address-families;
            container advertised-source-addr {
              uses addresses;
              description
                "The address used in the advertised LIE and TIE
                 packets.";
            }

            uses link-direction-type;
```

```
            leaf broadcast-capable {
              type boolean;
              description
                "If LIE can be received by broadcast address.";
              reference
                "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
                 Section 6.2.";
            }

            leaf allow-horizontal-link {
              type boolean;
              description
                "If this link allow horizontal link adjacency.";
            }

            container security {
              if-feature link-security;
              uses security;
              description
                "The security function used for this interface.";
              reference
                "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
                 Section 6.9.3.";
            }

            leaf security-checking {
              type enumeration {
                enum "no-checking" {
                  description
                    "The security envelope does not be checked.";
                }
                enum "permissive" {
                  description
                    "The security envelope checking is permissive.";
                }
                enum "loose" {
                  description
                    "The security envelope checking is loose.";
                }
                enum "strict" {
                  description
                    "The security envelope checking is strict.";
                }
              }
              description
                "The possible security checking types.
                 Only one type can be set at the same time.";
            }
```

```
          leaf was-the-last-lie-accepted {
            type boolean;
            config false;
            description
              "If the value is set to 'true', it means that
               the most recently received LIE was accepted.
               If the LIE was rejected, the neighbor error
               notifications should be used to find the reason.";
          }
          leaf last-lie-reject-reason {
            type string;
            config false;
            description
              "Description for the reject reason of the last LIE.";
          }
          container advertised-in-lies {
            config false;
            uses lie-elements;
            description
              "The attributes advertised in the LIEs from
               this interface.";
          }
          container link-capabilities {
            uses link-capabilities;
            description
              "The interface's link capabilities.";
          }
          leaf state {
            type enumeration {
              enum "one-way" {
                description
                  "The initial state.";
              }
              enum "two-way" {
                description
                  "Valid LIE received but not a Three Way LIE.";
              }
              enum "three-way" {
                description
                  "Valid Three Way LIE received.";
              }
              enum "multiple-neighbors-wait" {
                description
                  "More than two neighbors found in the same link.";
              }
            }
            config false;
            mandatory true;
```

```
            description
              "The states of LIE finite state machine.";
            reference
              "I-D.ietf-rift-rift: RIFT: Routing in Fat Trees.
               Section 6.2.1.";
          }
          leaf number-of-flaps {
            type uint32;
            config false;
            description
              "The number of interface state flaps.";
          }
          leaf last-state-change {
            type yang:date-and-time;
            config false;
            description
              "Time duration in the current state.";
          }
          leaf last-up {
            type yang:date-and-time;
            config false;
            description
              "The last time of up.";
          }
          leaf last-down {
            type yang:date-and-time;
            config false;
            description
              "The last time of down.";
          }

          container interface-states-statistics {
            config false;
            leaf intf-states-startup-time {
              type uint64;
              description
                "The states and statistics record startup time
                 of the interface.";
            }
            leaf num-of-nbrs-3way {
              type uint32;
              description
               "The number of neighbors which state is in 3-way.";
            }
            leaf num-of-nbrs-down {
              type uint32;
              description
               "The number of neighbors which state changed to down.";
```

```
            }
            list nbrs-down-reasons {
              key "system-id";
              leaf system-id {
                type system-id;
                description
                  "The system-id of neighbor.";
              }
              leaf last-down-reason {
                type string;
                description
                  "The last down reason of the neighbor.";
              }
              description
                "The down neighbors and reasons.";
            }

            leaf num-local-level-change {
              type uint32;
              description
                "The number of local level changes.";
            }

            container intf-lie-states {
              leaf last-lie-sent-time {
                type uint64;
                description
                  "The time of the last LIE sent.";
              }
              leaf last-lie-received-time {
                type uint64;
                description
                  "The time of the last LIE received.";
              }
              leaf num-lie-received {
                type uint32;
                description
                 "The number of received LIEs.";
              }
              leaf num-lie-transmitted {
                type uint32;
                description
                  "The number of transmitted LIEs.";
              }
              leaf num-lie-drop-invalid-envelope {
                type uint32;
                description
                  "The number of dropped LIEs due to
```

```
                invalid outer envelope.";
            }
            leaf num-lie-drop-invalid-nonce {
              type uint32;
              description
                "The number of dropped LIEs due to
                 invalid nonce.";
            }
            leaf num-lie-corrupted {
              type uint32;
              description
                "The number of corrupted LIEs received.";
            }
            description
              "The LIE's statistics of this interface.";
          }
          description
            "The states and statistics of this interface.";
        }

        container flood-repeater-statistics {
          config false;
          leaf flood-repeater {
            type system-id;
            description
              "The system-id of the current flood repeater.
               If this leaf has no value, that means the neighbor
               is not flood repeater.";
          }
          leaf num-flood-repeater-changes {
            type uint32;
            description
              "The number of flood repeater changes.";
          }
          leaf last-flood-repeater-change-reason {
            type string;
            description
              "The reason of the last flood repeater change.";
          }
          description
            "The flood repeater statistics.";
        }

        list neighbors {
          key "system-id";
          config false;
          uses base-node-info;
          uses neighbor;
```

```
            leaf local-nonce {
              type uint16;
              description
                "The exchanged local nonce with this neighbor.";
            }
            leaf remote-nonce {
              type uint16;
              description
                "The exchanged remote nonce to this neighbor.";
            }

            container tie-state-statistics {
              leaf transmit-queue {
                type uint32;
                description
                  "The length of TIE transmit queue.";
              }
              container last-queued-tie {
                uses tie-header;
                leaf reason-queued {
                  type string;
                  description
                    "The queued reason of the last queued TIE.";
                }
                description
                  "The last queued TIE for transmit.";
              }

              leaf num-received-ties {
                type uint32;
                description
                  "The number of TIEs received.";
              }
              leaf num-transmitted-ties {
                type uint32;
                description
                  "The number of TIEs transmitted.";
              }
              leaf num-retransmitted-ties {
                type uint32;
                description
                  "The number of TIEs retransmitted.";
              }
              leaf num-flood-reduced-ties {
                type uint32;
                description
                  "The number of TIEs that were flood reduced.";
              }
```

```
            leaf num-received-tides {
              type uint32;
              description
                "The number of TIDEs received.";
            }
            leaf num-transmitted-tides {
              type uint32;
              description
                "The number of TIDEs transmitted.";
            }
            leaf num-received-tires {
              type uint32;
              description
                "The number of TIREs received.";
            }
            leaf num-transmitted-tires {
              type uint32;
              description
                "The number of TIREs transmitted.";
            }
            leaf num-request-locally {
              type uint32;
              description
                "The number of TIEs requested locally.";
            }
            leaf num-request-remotely {
              type uint32;
              description
                "The number of TIEs requested by the neighbor.";
            }
            leaf num-same-older-ties-received {
              type uint32;
              description
                "The number of times of the same or older TIE
                 has been received.";
            }
            leaf num-seq-mismatch-pkts-received {
              type uint32;
              description
                "The number of packets with sequence number
                 mismatches.";
            }

            container last-sent-tie {
              uses tie-header;
              leaf last-tie-sent-time {
                type yang:date-and-time;
                description
```

```
          "The time of the last TIE sent.";
      }
      description
        "The information of the last sent TIE.";
    }

    container last-recv-tie {
      uses tie-header;
      leaf last-tie-recv-time {
        type yang:date-and-time;
        description
          "The time of the last TIE received.";
      }
      description
        "The information of the last received TIE.";
    }

    container largest-tie {
      container largest-tie-sent {
        uses tie-header;
        description
          "The largest TIE sent.";
      }
      container largest-tide-sent {
        uses tie-header;
        description
          "The largest TIDE sent.";
      }
      container largest-tire-sent {
        uses tie-header;
        description
          "The largest TIRE sent.";
      }
      description
        "The largest sent TIE, TIDE and TIRE.";
    }

    container num-tie-dropped {
      leaf num-tie-outer-envelope {
        type uint32;
        description
          "The total number of TIEs dropped due to
           invalid outer envelope.";
      }
      leaf num-tie-inner-envelope {
        type uint32;
        description
          "The total number of TIEs dropped due to
```

```
                    invalid inner envelope.";
                }
                leaf num-tie-nonce {
                  type uint32;
                  description
                    "The total number of TIEs dropped due to
                     invalid nonce.";
                }

                description
                  "The total number of TIEs dropped due to
                   security reasons.";
              }

              description
                "The states and statistics of TIE, TIDE, TIRE
                 exchanging with this neighbor.";
            }
            description
              "The neighbor's information.";
          }

          description
            "The interface information on this node.";
        } // list interface

        container database {
          config false;
          list ties {
            key "tie-direction-type originator tie-type tie-number";
            description
              "A list of TIEs (Topology Information Elements).";
            uses tie-header;

            container node {
              leaf level {
                type level;
                config false;
                description
                  "The level of this node.";
              }
              list neighbors {
                key "system-id";
                uses base-node-info;
                uses neighbor-node;
                description
                  "The node TIE information of a neighbor.";
              }
```

```
            uses node-capability;
            leaf overload-flag {
              type boolean;
              description
                "If the value is set to 'true', it means that
                 the overload bit in TIEs is set.";
            }
            leaf name {
              type string;
              description
                "The name of this node. It won't be used as the
                 key of node, just used for description.";
            }
            leaf pod {
              type uint32;
              description
                "Point of Delivery. The self-contained vertical
                 slice of a Clos or Fat Tree network containing
                 normally only level 0 and level 1 nodes. It
                 communicates with nodes in other PoDs via the
                 spine. We number PoDs to distinguish them and
                 use PoD #0 to denote 'undefined' PoD.";
            }
            leaf startup-time {
              type uint64;
              description
                "Startup time of the node.";
            }
            leaf-list miscabled-links {
              type uint32;
              config false;
                description
                  "List of miscabled links.";
            }
            leaf-list same-plane-tofs {
              type system-id;
              config false;
                description
                  "ToFs in the same plane. Only carried by ToF.
                   Multiple Node TIEs can carry disjoint sets of
                   ToFs which MUST be joined to form a single
                   set.";
            }
            leaf fabric-id {
              type uint32;
              config false;
                description
                  "The optional ID of the Fabric configured.";
```

```
              }
              description
                "The node element information in this TIE.";
            } // node

            container prefixes {
              description
                "The prefix element information in this TIE.";
              list prefixes {
                key "prefix";
                leaf prefix {
                  type inet:ip-prefix;
                  description
                    "The prefix information.";
                }
                uses tie-type;
                uses prefix-attribute;
                description
                  "The prefix set information.";
              }
            }

            container key-value {
              leaf key {
                type binary;
                description
                  "The type of key value combination.";
              }
              leaf value {
                type binary;
                description
                  "The value of key value combination.";
              }
              description
                "The information used to distinguish a Key/Value
                 pair. When the type of kv is set to 'node',
                 node-element is making sense. When the type of
                 kv is set to other values except 'node',
                 prefix-info is making sense.";
            } // kv-store
          } // ties
          description
            "The TIEs information in database.";
        }// container database
        description
          "RIFT configuration and state data.";
      }//rift
    }//augment
```

```
     /*
      * Notifications
      */
     notification error-set {
       description
         "The errors notification of RIFT.";
       container tie-level-error {
         description
           "The TIE errors notification of RIFT.";

         list rift {
           key "name";
           leaf name {
             type string;
             description
               "The RIFT instance's name.";
           }
           list ties {
             key "originator";
             uses tie-header;
             description
               "The level is undefined in the LIEs.";
           }
           description
             "The TIE errors set.";
         }
       }
       container neighbor-error {
         description
           "The neighbor errors notification of RIFT.";
         list rift {
           key "name";
           leaf name {
             type string;
             description
               "The RIFT instance's name.";
           }
           list neighbors {
             key "system-id";
             uses base-node-info;
             uses neighbor;
             description
               "The information of a neighbor.";
           }
           description
             "The neighbor errors set.";
         }
       }
```

```
    }
  }
  <CODE ENDS>
```

4. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. Writable data node represent configuration of each instance, node, interface, etc. These correspond to the following schema node:

* /rift

Modifying the configuration may cause all the RIFT neighborship to be rebuilt. For example, the configuration changing of configured-level or system-id will lead to all the neighbor connections of this node rebuilt. The incorrect modification of authentication, except for the neighbor connection broken, will lead to the permanent connection broken. The modification of interface will lead to the neighbor state changing. In general, unauthorized modification of most RIFT configurations will pose their own set of security risks and the "Security Considerations" in the respective reference RFCs should be consulted.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

* /rift

* /rift/tie-security

* /rift/interface

* /rift/neighbor

* /rift/database

The exposure of the database will expose the detailed topology of the network. Network operators may consider their topologies to be sensitive confidential data.

For RIFT authentication, configuration is supported via the specification of key-chains [RFC8177] or the direct specification of key and authentication algorithm. Hence, authentication configuration inherits the security considerations of [RFC8177]. This includes the considerations with respect to the local storage and handling of authentication keys.

The actual authentication key data (whether locally specified or part of a key chain) is sensitive and needs to be kept secret from unauthorized parties; compromise of the key data would allow an attacker to forge RIFT packet that would be accepted as authentic, potentially compromising the entire domain.

5. IANA Considerations

RFC Ed.: Please replace all occurrences of 'XXXX' with the actual RFC number (and remove this note).

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-rift

Registrant Contact: The IESG

XML: N/A, the requested URI is an XML namespace.

This document also requests one new YANG module name in the YANG Module Names registry [RFC6020] with the following suggestion:

name: ietf-rift

namespace: urn:ietf:params:xml:ns:yang:ietf-rift

prefix: rift

reference: RFC XXXX

6. Acknowledgement

The authors would like to thank Tony Przygienda, Jordan Head, Benchong Xu (xu.benchong@zte.com.cn), Tom Petch for their review, valuable comments and suggestions.

7. References

7.1. Normative References

[I-D.ietf-rift-rift]
Przygienda, T., Head, J., Sharma, A., Thubert, P., Rijsman, B., and D. Afanasiev, "RIFT: Routing in Fat Trees", Work in Progress, Internet-Draft, draft-ietf-rift-rift-24, 23 May 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-rift-rift-24>.

[IEEE8021AS]
"IEEE Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks", <https://ieeexplore.ieee.org/document/5741898/>.

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

[RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, DOI 10.17487/RFC5881, June 2010, <https://www.rfc-editor.org/info/rfc5881>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

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

[RFC8177] Lindem, A., Ed., Qu, Y., Yeung, D., Chen, I., and J. Zhang, "YANG Data Model for Key Chains", RFC 8177, DOI 10.17487/RFC8177, June 2017, <https://www.rfc-editor.org/info/rfc8177>.

[RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, DOI 10.17487/RFC8294, December 2017, <https://www.rfc-editor.org/info/rfc8294>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

[RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <https://www.rfc-editor.org/info/rfc8349>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8505] Thubert, P., Ed., Nordmark, E., Chakrabarti, S., and C. Perkins, "Registration Extensions for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Neighbor Discovery", RFC 8505, DOI 10.17487/RFC8505, November 2018, <https://www.rfc-editor.org/info/rfc8505>.

7.2. Informative References

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", BCP 216, RFC 8407, DOI 10.17487/RFC8407, October 2018, <https://www.rfc-editor.org/info/rfc8407>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

[RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/info/rfc8641>.

Authors' Addresses

Zheng Zhang
ZTE Corporation
Email: zhang.zheng@zte.com.cn

Yuehua Wei
ZTE Corporation
Email: wei.yuehua@zte.com.cn

Shaowen Ma
Google
Email: mashaowen@gmail.com

Xufeng Liu
Alef Edge
Email: xufeng.liu.ietf@gmail.com

Bruno Rijsman
Individual
Email: brunorijsman@gmail.com

TEAS Working Group
Internet Draft
Category: Informational

Haomian Zheng
Yi Lin
Huawei Technologies
Yang Zhao
China Mobile
Yunbin Xu
CAICT
Dieter Beller
Nokia

Expires: January 25, 2025
July 24, 2024

Interworking of GMPLS Control and Centralized Controller Systems

draft-ietf-teas-gmpls-controller-inter-work-15

Abstract

Generalized Multi-Protocol Label Switching (GMPLS) control allows each network element (NE) to perform local resource discovery, routing and signaling in a distributed manner.

The advancement of software-defined transport networking technology enables a group of NEs to be managed through centralized controller hierarchies. This helps to tackle challenges arising from multiple domains, vendors, and technologies. An example of such a centralized architecture is the Abstraction and Control of Traffic Engineered Networks (ACTN) controller hierarchy, as described in RFC 8453.

Both the distributed and centralized control planes have their respective advantages and should complement each other in the system, rather than competing. This document outlines how the GMPLS distributed control plane can work together with a centralized controller system in a transport 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."

This Internet-Draft will expire on January 25, 2025.

Table of Contents

1. Introduction

Generalized Multi-Protocol Label Switching (GMPLS) [RFC3945] extends MPLS to support different classes of interfaces and switching capabilities such as Time-Division Multiplex Capable (TDM), Lambda Switch Capable (LSC), and Fiber-Switch Capable (FSC). Each network element (NE) running a GMPLS control plane collects network information from other NEs and supports service provisioning through signaling in a distributed manner. A more generic description of Traffic-engineering networking information exchange can be found in [RFC7926].

On the other hand, Software-Defined Networking (SDN) technologies have been introduced to control the transport network centrally. Centralized controllers can collect network information from each node and provision services on corresponding nodes. One example is the Abstraction and Control of Traffic Engineered Networks (ACTN) [RFC8453], which defines a hierarchical architecture with Provisioning Network Controller (PNC), Multi-domain Service Coordinator (MDSC) and Customer Network Controller (CNC) as centralized controllers for different network abstraction levels. A Path Computation Element (PCE) based approach has been proposed as Application-Based Network Operations (ABNO) in [RFC7491].

GMPLS can be used to control Network Elements (NEs) in such centralized controller architectures. A centralized controller may support GMPLS-enabled domains and communicate with a GMPLS-enabled domain where the GMPLS control plane handles service provisioning from ingress to egress. In this scenario, the centralized controller sends a request to the entry node and does not need to configure all

NEs along the path within the domain from ingress to egress, thus leveraging the GMPLS control plane. This document describes how the GMPLS control plane interworks with a centralized controller system in a transport network.

2. Abbreviations

The following abbreviations are used in this document.

ABNO	Application-Based Network Operations
ACTN	Abstraction and Control of Traffic Engineered Networks
APS	Automatic Protection Switching
BRPC	Backward-Recursive PCE-Based Computation
CNC	Customer Network Controller
CSPF	Constraint-based Shortest Path First
DoS	Denial-of-Service
E2E	End-to-end
ERO	Explicit Route Object
FA	Forwarding Adjacency
FRR	Fast Reroute
FSC	Fiber-Switch Capable
GMPLS	Generalized Multi-Protocol Label Switching
H-PCE	Hierarchical PCE
IDS	Intrusion Detection System
IGP	Interior Gateway Protocol
IoC	Indicators of Compromise
IPS	Intrusion Prevention System
IS-IS	Intermediate System to Intermediate System protocol
LMP	Link Management Protocol
LSC	Lambda Switch Capable
LSP	Label Switched Path
LSP-DB	LSP Database
MD	Multi-domain
MDSC	Multi-domain Service Coordinator
MITM	Man-In-The-Middle
ML	Multi-layer
MPI	MDSC to PNC Interface
NE	Network element
NETCONF	Network Configuration Protocol
NMS	Network Management System
OSPF	Open Shortest Path First protocol
PCC	Path Computation Client
PCE	Path Computation Element
PCEP	Path Computation Element communication Protocol
PCEP-LS	Link-state PCEP
PLR	Point of Local Repair
PNC	Provisioning Network Controller
RSVP	Resource Reservation Protocol
SBI	Southbound Interface
SDN	Software-Defined Networking

TDM	Time-Division Multiplex Capable
TE	Traffic Engineering
TED	Traffic Engineering Database
TLS	Transport Layer Security
VNTM	Virtual Network Topology Manager

3. Overview

This section provides an overview of the GMPLS control plane, centralized controller systems and their interactions in transport networks.

A transport network [RFC5654] is a server-layer network designed to provide connectivity services for client-layer connectivity. This setup allows client traffic to be carried seamlessly across the server-layer network resources.

3.1. Overview of GMPLS Control Plane

GMPLS separates the control plane and the data plane to support time-division, wavelength, and spatial switching, which are significant in transport networks. For the NE level control in GMPLS, each node runs a GMPLS control plane instance. Functionalities such as service provisioning, protection, and restoration can be performed via GMPLS communication among multiple NEs. At the same time, the GMPLS control plane instance can also collect information about node and link resources in the network to construct the network topology and compute routing paths for serving service requests.

Several protocols have been designed for the GMPLS control plane [RFC3945], including link management [RFC4204], signaling [RFC3471], and routing [RFC4202] protocols. The GMPLS control plane instances applying these protocols communicate with each other to exchange resource information and establish Label Switched Paths (LSPs). In this way, GMPLS control plane instances in different nodes in the network have the same view of the network topology and provision services based on local policies.

3.2. Overview of Centralized Controller System

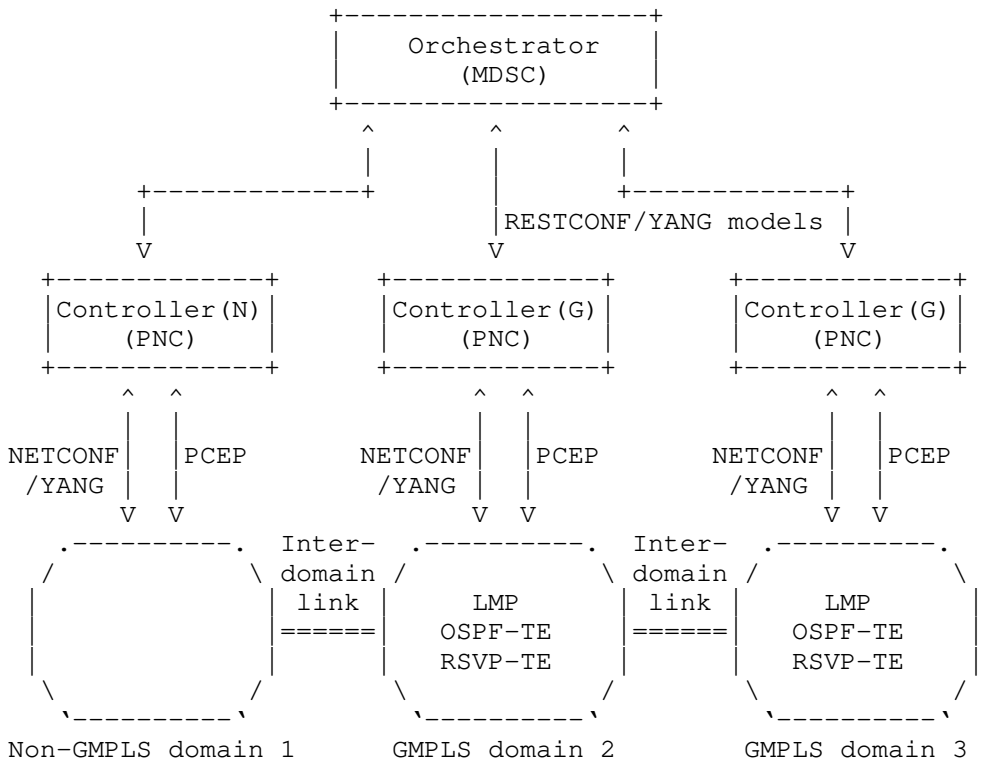
With the development of SDN technologies, a centralized controller architecture has been introduced to transport networks. One example architecture can be found in ACTN [RFC8453]. In such systems, a controller is aware of the network topology and is responsible for provisioning incoming service requests.

Multiple hierarchies of controllers are designed at different levels to implement different functions. This kind of architecture enables multi-vendor, multi-domain, and multi-technology control. For

example, a higher-level controller coordinates several lower-level controllers controlling different domains, for topology collection and service provisioning. Vendor-specific features can be abstracted between controllers, and a standard API (e.g., generated from RESTCONF [RFC8040] / YANG [RFC7950]) may be used.

3.3. GMPLS Control Interworking with a Centralized Controller System

Besides GMPLS and the interactions among the controller hierarchies, it is also necessary for the controllers to communicate with the network elements. Within each domain, GMPLS control can be applied to each NE. The bottom-level centralized controller can act as an NE to collect network information and initiate LSPs. Figure 1 shows an example of GMPLS interworking with centralized controllers (ACTN terminologies are used in the figure).



Controller(N): A domain controller controlling a non-GMPLS domain
Controller(G): A domain controller controlling a GMPLS domain

Figure 1: Example of GMPLS/non-GMPLS interworking with Controllers

Figure 1 shows the scenario with two GMPLS domains and one non-GMPLS domain. This system supports the interworking among non-GMPLS domain, GMPLS domain and the controller hierarchies.

For domain 1, the network elements were not enabled with GMPLS so the control is purely from the controller, via Network Configuration Protocol (NETCONF) [RFC6241] / YANG and/or PCE Communication Protocol (PCEP) [RFC5440].

For domains 2 and 3:

- Each domain has the GMPLS control plane enabled at the physical network level. The Provisioning Network Controller (PNC) can exploit GMPLS capabilities implemented in the domain to listen to the IGP routing protocol messages (OSPF LSAs, for example) that the GMPLS control plane instances are disseminating into the network and thus learn the network topology. For path computation in the domain with PNC implementing a PCE, Path Computation Clients (PCCs) (e.g. NEs, other controller/PCE) use PCEP to ask the PNC for a path and get replies. The Multi-Domain Service Coordinator (MDSC) communicates with PNCs using, for example REST/RESTCONF based on YANG data models. As a PNC has learned its domain topology, it can report the topology to the MDSC. When a service arrives, the MDSC computes the path and coordinates PNCs to establish the corresponding LSP segment;

- Alternatively, the NETCONF protocol can be used to retrieve topology information utilizing the [RFC8795] YANG model and the technology-specific YANG model augmentations required for the specific network technology. The PNC can retrieve topology information from any NE (the GMPLS control plane instance of each NE in the domain has the same topological view), construct the topology of the domain, and export an abstract view to the MDSC. Based on the topology retrieved from multiple PNCs, the MDSC can create a topology graph of the multi-domain network, and can use it for path computation. To set up a service, the MDSC can exploit the [TE-Tunnel] YANG model together with the technology-specific YANG model augmentations.

This document focuses on the interworking between GMPLS and the centralized controller system, including:

- The interworking between the GMPLS domains and the centralized controllers (including the orchestrator, if it exists) controlling the GMPLS domains;

- The interworking between a non-GMPLS domain (which is controlled by a centralized controller system) and a GMPLS domain, through the controller hierarchy architecture.

For convenience, this document uses the following terminologies for the controller and the orchestrator:

- Controller(G): A domain controller controlling a GMPLS domain (the controller(G) of the GMPLS domains 2 and 3 in Figure 1);
- Controller(N): A domain controller controlling a non-GMPLS domain (the controller(N) of the non-GMPLS domain 1 in Figure 1);
- H-Controller(G): A domain controller controlling the higher-layer GMPLS domain, in the context of multi-layer networks;
- L-Controller(G): A domain controller controlling the lower-layer GMPLS domain, in the context of multi-layer networks;
- H-Controller(N): A domain controller controlling the higher-layer non-GMPLS domain, in the context of multi-layer networks;
- L-Controller(N): A domain controller controlling the lower-layer non-GMPLS domain, in the context of multi-layer networks;
- Orchestrator(MD): An orchestrator used to orchestrate the multi-domain networks;
- Orchestrator(ML): An orchestrator used to orchestrate the multi-layer networks.

4. Discovery Options

In GMPLS control, the link connectivity must be verified between each pair of nodes. In this way, link resources, which are fundamental resources in the network, are discovered by both ends of the link.

4.1. LMP

Link management protocol (LMP) [RFC4204] runs between nodes and manages TE links. In addition to the setup and maintenance of control channels, LMP can be used to verify the data link connectivity and correlate the link properties.

5. Routing Options

In GMPLS control, link state information is flooded within the network as defined in [RFC4202]. Each node in the network can build the network topology according to the flooded link state information. Routing protocols such as OSPF-TE [RFC4203] and ISIS-TE [RFC5307] have been extended to support different interfaces in GMPLS.

In a centralized controller system, the centralized controller can be placed in the GMPLS network and passively receives the IGP information flooded in the network. In this way, the centralized controller can construct and update the network topology.

5.1. OSPF-TE

OSPF-TE is introduced for TE networks in [RFC3630]. OSPF extensions have been defined in [RFC4203] to enable the capability of link state information for the GMPLS network. Based on this work, OSPF has been extended to support technology-specific routing. The routing protocol for Optical Transport Network (OTN), Wavelength Switched Optical Network (WSON) and optical flexi-grid networks are defined in [RFC7138], [RFC7688] and [RFC8363], respectively.

5.2. ISIS-TE

ISIS-TE is introduced for TE networks in [RFC5305] and is extended to support GMPLS routing functions [RFC5307], and has been updated to [RFC7074] to support the latest GMPLS switching capability and Types fields.

5.3. NETCONF/RESTCONF

NETCONF [RFC6241] and RESTCONF [RFC8040] protocols are originally used for network configuration. These protocols can also utilize topology-related YANG models, such as [RFC8345] and [RFC8795]. These protocols provide a powerful mechanism for notification of topology changes to the client.

6. Path Computation

6.1. Controller-based Path Computation

Once a controller learns the network topology, it can utilize the available resources to serve service requests by performing path computation. Due to abstraction, the controllers may not have sufficient information to compute the optimal path. In this case, the controller can interact with other controllers by sending, for example, YANG-based Path Computation requests [PAT-COMP] or PCEP, to compute a set of potential optimal paths and then, based on its constraints, policy, and specific knowledge (e.g. cost of access link) can choose the more feasible path for end-to-end (E2E) service path setup.

Path computation is one of the key objectives in various types of controllers. In the given architecture, it is possible for different components that have the capability to compute the path.

6.2. Constraint-based Path Computing in GMPLS Control

In GMPLS control, a routing path may be computed by the ingress node ([RFC3473]) based on the ingress node Traffic Engineering Database (TED). In this case, constraint-based path computation is performed according to the local policy of the ingress node.

6.3. Path Computation Element (PCE)

The PCE was first introduced in [RFC4655] as a functional component that offers services for computing paths within a network. In [RFC5440], path computation is achieved using the TED, which maintains a view of the link resources in the network. The introduction of PCE has significantly improved the quality of network planning and offline computation. However, there is a potential risk that the computed path may be infeasible when there is a diversity requirement, as stateless PCE lacks knowledge about previously computed paths.

To address this issue, stateful PCE has been proposed in [RFC8231]. Besides the TED, an additional LSP Database (LSP-DB) is introduced to archive each LSP computed by the PCE. This way, PCE can easily determine the relationship between the computing path and former computed paths. In this approach, PCE provides computed paths to PCC, and then PCC decides which path is deployed and when to be established.

With PCE-Initiated LSPs [RFC8281], PCE can trigger the PCC to perform setup, maintenance, and teardown of the PCE-initiated LSP under the stateful PCE model. This would allow a dynamic network that is centrally controlled and deployed.

In a centralized controller system, the PCE can be implemented within the centralized controller. The centralized controller then calculates paths based on its local policies. Alternatively, the PCE can be located outside of the centralized controller. In this scenario, the centralized controller functions as a PCC and sends a path computation request to the PCE using the PCEP. A reference architecture for this can be found in [RFC7491].

7. Signaling Options

Signaling mechanisms are used to set up LSPs in GMPLS control. Messages are sent hop by hop between the ingress node and the egress node of the LSP to allocate labels. Once the labels are allocated along the path, the LSP setup is accomplished. Signaling protocols such as Resource Reservation Protocol - Traffic Engineering (RSVP-TE) [RFC3473] have been extended to support different interfaces in GMPLS.

7.1. RSVP-TE

RSVP-TE is introduced in [RFC3209] and extended to support GMPLS signaling in [RFC3473]. Several label formats are defined for a generalized label request, a generalized label, a suggested label and label sets. Based on [RFC3473], RSVP-TE has been extended to support technology-specific signaling. The RSVP-TE extensions for OTN, WSON, and optical flexi-grid network are defined in [RFC7139], [RFC7689], and [RFC7792], respectively.

8. Interworking Scenarios

8.1. Topology Collection & Synchronization

Topology information is necessary on both network elements and controllers. The topology on a network element is usually raw information, while the topology used by the controller can be either raw, reduced, or abstracted. Three different abstraction methods have been described in [RFC8453], and different controllers can select the corresponding method depending on the application.

When there are changes in the network topology, the impacted network elements need to report changes to all the other network elements, together with the controller, to sync up the topology information. The inter-NE synchronization can be achieved via protocols mentioned in Sections 4 and 5. The topology synchronization between NEs and controllers can either be achieved by routing protocols OSPF-TE/PCEP-LS in [PCEP-LS] or NETCONF protocol notifications with YANG model.

8.2. Multi-domain Service Provisioning

Service provisioning can be deployed based on the topology information on controllers and network elements. Many methods have been specified for single-domain service provisioning, such as the PCEP and RSVP-TE methods.

Multi-domain service provisioning would require coordination among the controller hierarchies. Given the service request, the end-to-end delivery procedure may include interactions at any level (i.e. interface) in the hierarchy of the controllers (e.g. MPI and SBI for ACTN). The computation for a cross-domain path is usually completed by controllers who have a global view of the topologies. Then the configuration is decomposed into lower-level controllers, to configure the network elements to set up the path.

A combination of centralized and distributed protocols may be necessary to interact between network elements and controllers. Several methods can be used to create the inter-domain path:

1) With end-to-end RSVP-TE session:

In this method, all the domains need to support the RSVP-TE protocol and thus need to be GMPLS domains. The Controller(G) of the source domain triggers the source node to create the end-to-end RSVP-TE session, and the assignment and distribution of the labels on the inter-domain links are done by the border nodes of each domain, using RSVP-TE protocol. Therefore, this method requires the interworking of RSVP-TE protocols between different domains.

There are two possible methods:

1.1) One single end-to-end RSVP-TE session

In this method, an end-to-end RSVP-TE session from the source node to the destination node will be used to create the inter-domain path. A typical example would be the PCE Initiation scenario, in which a PCE message (PCInitiate) is sent from the controller(G) to the source node, triggering an RSVP procedure along the path. Similarly, the interaction between the controller and the source node of the source domain can be achieved by using the NETCONF protocol with corresponding YANG models, and then it can be completed by running RSVP among the network elements.

1.2) LSP Stitching

The LSP stitching method defined in [RFC5150] can also create the E2E LSP. I.e., when the source node receives an end-to-end path creation request (e.g., using PCEP or NETCONF protocol), the source node starts an end-to-end RSVP-TE session along the endpoints of each LSP segment (refers to S-LSP in [RFC5150]) of each domain, to assign the labels on the inter-domain links between each pair of neighbor S-LSPs, and stitch the end-to-end LSP to each S-LSP. See Figure 2 as an example. Note that the S-LSP in each domain can be either created by its Controller(G) in advance, or created dynamically triggered by the end-to-end RSVP-TE session.

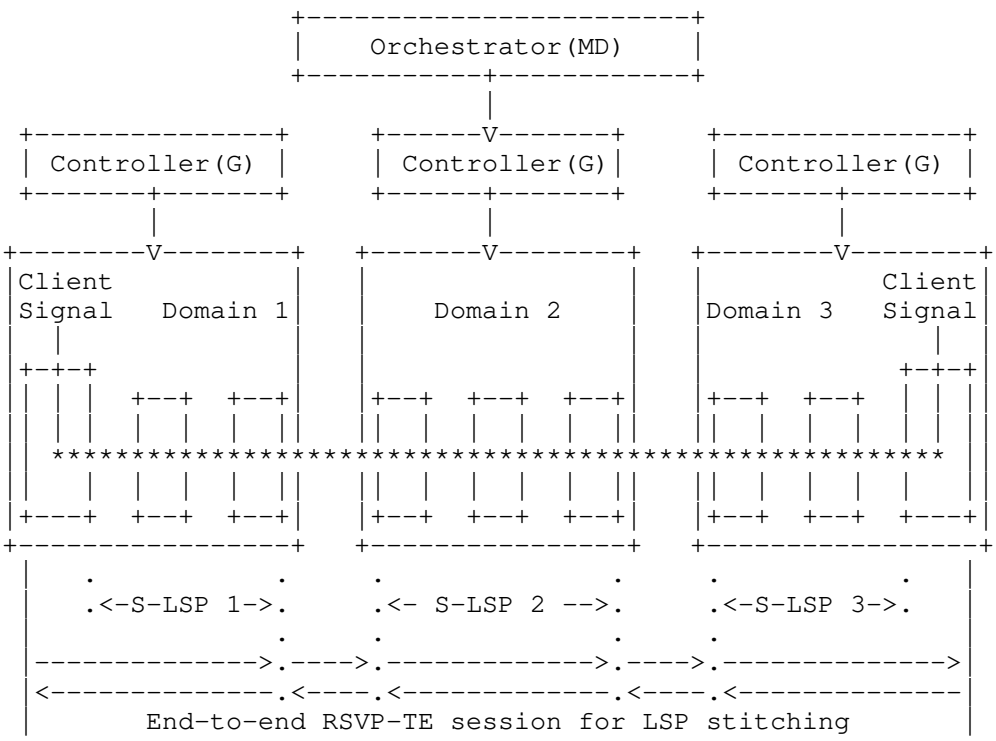


Figure 2: LSP stitching

2) Without end-to-end RSVP-TE session:

In this method, each domain can be a GMPLS domain or a non-GMPLS domain. Each controller (may be a Controller(G) or a Controller(N)) is responsible for creating the path segment within its domain. The border node does not need to communicate with other border nodes in other domains for the distribution of labels on inter-domain links, so end-to-end RSVP-TE session through multiple domains is not required, and the interworking of RSVP-TE protocol between different domains is not needed.

Note that path segments in the source domain and the destination domain are "asymmetrical" segments, because the configuration of client signal mapping into server layer tunnel is needed at only one end of the segment, while configuration of server layer cross-connect is needed at the other end of the segment. See the example in Figure 3.

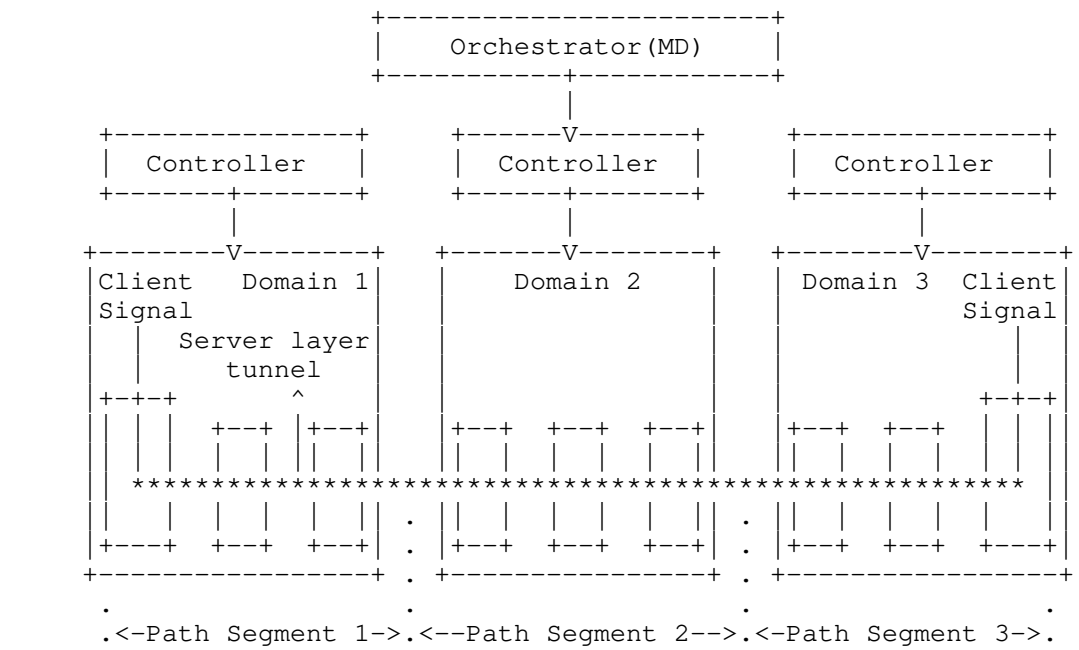


Figure 3: Example of asymmetrical path segment

The PCEP / GMPLS protocols should support the creation of such asymmetrical segments.

Note also that mechanisms to assign the labels in the inter-domain links also need to be considered. There are two possible methods:

2.1) Inter-domain labels assigned by NEs:

The concept of Stitching Label that allows stitching local path segments was introduced in [RFC5150] and [sPCE-ID], in order to form the inter-domain path crossing several different domains. It also describes the Backward-Recursive PCE-Based Computation (BRPC) and Hierarchical Path Computation Element (H-PCE) PCInitiate procedure, i.e., the ingress node of each downstream domain assigns the stitching label for the inter-domain link between the downstream domain and its upstream neighbor domain, and this stitching label will be passed to the upstream neighbor domain by PCE protocol, which will be used for the path segment creation in the upstream neighbor domain.

2.2) Inter-domain labels assigned by controller:

If the resources of inter-domain links are managed by the orchestrator(MD), each domain controller can provide to the orchestrator(MD) the list of available labels (e.g. timeslots if OTN

is the scenario) using the IETF Topology YANG model and related technology specific extension. Once the orchestrator(MD) has computed the E2E path, RSVP-TE or PCEP can be used in the different domains to set up the related segment tunnel consisting of label inter-domain information, e.g. for PCEP, the label Explicit Route Object (ERO) can be included in the PCInitiate message to indicate the inter-domain labels, so that each border node of each domain can configure the correct cross-connect within itself.

8.3. Multi-layer Service Provisioning

GMPLS can interwork with centralized controller systems in multi-layer networks.

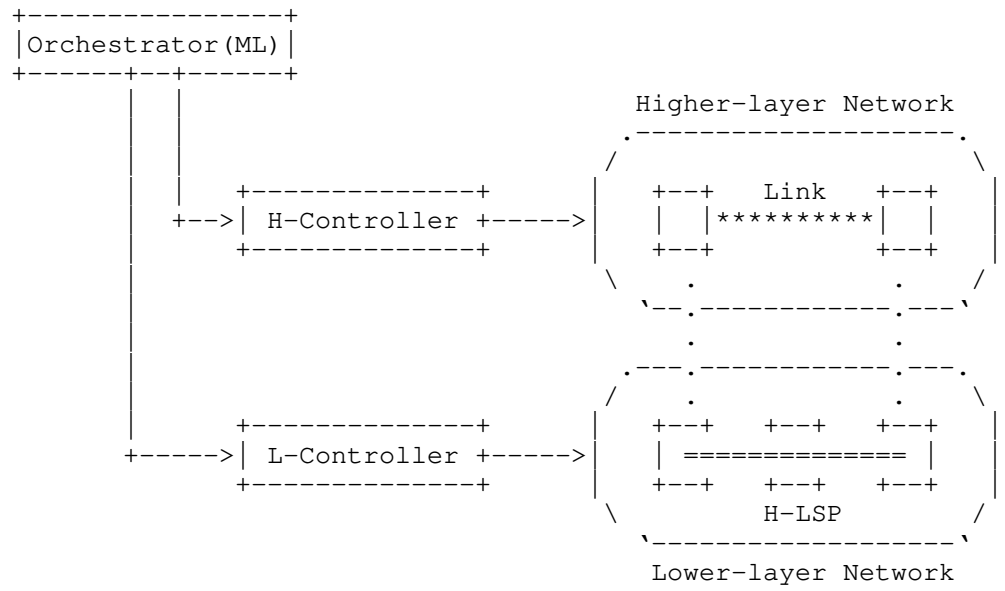


Figure 4: GMPLS-controller interworking in multi-layer networks

An example with two layers of network is shown in Figure 4. In this example, the GMPLS control plane is enabled in at least one layer network (otherwise, it is out of the scope of this document) and interworks with the controller of its domain (H-Controller and L-Controller, respectively). The Orchestrator(ML) is used to coordinate the control of the multi-layer network.

8.3.1. Multi-layer Path Computation

[RFC5623] describes three inter-layer path computation models and four inter-layer path control models:

- 3 Path computation:

- Single PCE path computation model
- Multiple PCE path computation with inter-PCE communication model
- Multiple PCE path computation without inter-PCE communication model

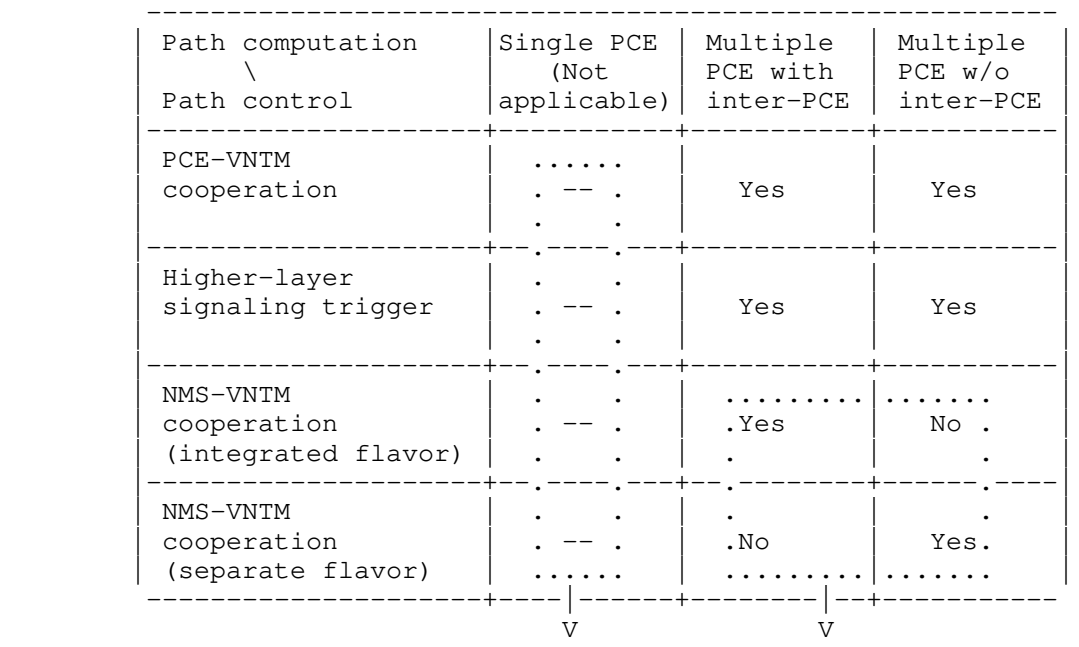
- 4 Path control:
  - PCE-Virtual Network Topology Manager (PCE-VNTM) cooperation model
  - Higher-layer signaling trigger model
  - Network Management System-VNTM (NMS-VNTM) cooperation model (integrated flavor)
  - NMS-VNTM cooperation model (separate flavor)

Section 4.2.4 of [RFC5623] also provides all the possible combinations of inter-layer path computation and inter-layer path control models.

To apply [RFC5623] in multi-layer network with GMPLS-controller interworking, the H-Controller and the L-Controller can act as the PCE Hi and PCE Lo respectively, and typically, the Orchestrator(ML) can act as a VNTM because it has the abstracted view of both the higher-layer and lower-layer networks.

Table 1 shows all possible combinations of path computation and path control models in multi-layer network with GMPLS-controller interworking:

Table 1: Combinations of path computation and path control models



Note that:

- Since there is one PCE in each layer network, the path computation model "Single PCE path computation" is not applicable.

- For the other two path computation models "Multiple PCE with inter-PCE" and "Multiple PCE w/o inter-PCE", the possible combinations are the same as defined in [RFC5623]. More specifically:

  o The path control models "NMS-VNTM cooperation (integrated flavor)" and "NMS-VNTM cooperation (separate flavor)" are the typical models to be used in multi-layer network with GMPLS-controller interworking. This is because in these two models, the path computation is triggered by the NMS or VNTM. And in the centralized controller system, the path computation requests are typically from the Orchestrator(ML) (acts as VNTM).

  o For the other two path control models "PCE-VNTM cooperation" and "Higher-layer signaling trigger", the path computation is triggered by the NEs, i.e., NE performs PCC functions. These

two models are still possible to be used, although they are not the main methods.

8.3.2. Cross-layer Path Creation

In a multi-layer network, a lower-layer LSP in the lower-layer network can be created, which will construct a new link in the higher-layer network. Such lower-layer LSP is called Hierarchical LSP, or H-LSP for short, see [RFC6107].

The new link constructed by the H-LSP can then be used by the higher-layer network to create new LSPs.

As described in [RFC5212], two methods are introduced to create the H-LSP: the static (pre-provisioned) method and the dynamic (triggered) method.

1) Static (pre-provisioned) method

In this method, the H-LSP in the lower-layer network is created in advance. After that, the higher-layer network can create LSPs using the resource of the link constructed by the H-LSP.

The Orchestrator(ML) is responsible to decide the creation of H-LSP in the lower-layer network if it acts as a VNTM. It then requests the L-Controller to create the H-LSP via, for example, MPI interface under the ACTN architecture. See Section 3.3.2 of [TE-Tunnel].

If the lower-layer network is a GMPLS domain, the L-Controller(G) can trigger the GMPLS control plane to create the H-LSP. As a typical example, the PCInitiate message can be used for the communication between the L-Controller and the source node of the H-LSP. And the source node of the H-LSP can trigger the RSVP-TE signaling procedure to create the H-LSP, as described in [RFC6107].

If the lower-layer network is a non-GMPLS domain, other methods may be used by the L-Controller(N) to create the H-LSP, which is out of scope of this document.

2) Dynamic (triggered) method

In this method, the signaling of LSP creation in the higher-layer network will trigger the creation of H-LSP in the lower-layer network dynamically, if it is necessary. Therefore, both the higher-layer and lower-layer networks need to support the RSVP-TE protocol and thus need to be GMPLS domains.

In this case, after the cross-layer path is computed, the Orchestrator(ML) requests the H-Controller(G) for the cross-layer

LSP creation. As a typical example, the MPI interface under the ACTN architecture could be used.

The H-Controller(G) can trigger the GMPLS control plane to create the LSP in the higher-layer network. As a typical example, the PCInitiate message can be used for the communication between the H-Controller(G) and the source node of the Higher-layer LSP, as described in Section 4.3 of [RFC8282]. At least two sets of ERO information should be included to indicate the routes of higher-layer LSP and lower-layer H-LSP.

The source node of the Higher-layer LSP follows the procedure defined in Section 4 of [RFC6001], to trigger the GMPLS control plane in both higher-layer network and lower-layer network to create the higher-layer LSP and the lower-layer H-LSP.

On success, the source node of the H-LSP should report the information of the H-LSP to the L-Controller(G) via, for example, PCRpt message.

8.3.3. Link Discovery

If the higher-layer network and the lower-layer network are under the same GMPLS control plane instance, the H-LSP can be a Forwarding Adjacency LSP (FA-LSP). Then the information of the link constructed by this FA-LSP, called Forwarding Adjacency (FA), can be advertised in the routing instance, so that the H-Controller can be aware of this new FA. [RFC4206] and the following updates to it (including [RFC6001] and [RFC6107]) describe the detailed extensions to support advertisement of an FA.

If the higher-layer network and the lower-layer network are under separate GMPLS control plane instances, or one of the layer networks is a non-GMPLS domain, after an H-LSP is created in the lower-layer network, the link discovery procedure will be triggered in the higher-layer network to discover the information of the link constructed by the H-LSP. LMP protocol defined in [RFC4204] can be used if the higher-layer network supports GMPLS. The information of this new link will be advertised to the H-Controller.

8.4. Recovery

The GMPLS recovery functions are described in [RFC4426]. Span protection, end-to-end protection and restoration, are discussed with different protection schemes and message exchange requirements. Related RSVP-TE extensions to support end-to-end recovery is described in [RFC4872]. The extensions in [RFC4872] include protection, restoration, preemption, and rerouting mechanisms for an end-to-end LSP. Besides end-to-end recovery, a GMPLS segment recovery mechanism is defined in [RFC4873], which also intends to be

compatible with Fast Reroute (FRR) (see [RFC4090] which defines RSVP-TE extensions for the FRR mechanism, and [RFC8271] which described the updates of GMPLS RSVP-TE protocol for FRR of GMPLS TE-LSPs).

8.4.1. Span Protection

Span protection refers to the protection of the link between two neighboring switches. The main protocol requirements include:

- Link management: Link property correlation on the link protection type;

- Routing: announcement of the link protection type;

- Signaling: indication of link protection requirement for that LSP.

GMPLS already supports the above requirements, and there are no new requirements in the scenario of interworking between GMPLS and centralized controller system.

8.4.2. LSP Protection

The LSP protection includes end-to-end and segment LSP protection. For both cases:

- In the provisioning phase:

  In both single-domain and multi-domain scenarios, the disjoint path computation can be done by the centralized controller system, as it has the global topology and resource view. And the path creation can be done by the procedure described in Section 8.2.

- In the protection switchover phase:

  In both single-domain and multi-domain scenarios, the existing standards provide the distributed way to trigger the protection switchover. For example, data plane Automatic Protection Switching (APS) mechanism described in [G.808.1], [RFC7271] and [RFC8234], or GMPLS Notify mechanism described in [RFC4872] and [RFC4873]. In the scenario of interworking between GMPLS and centralized controller system, using these distributed mechanisms rather than centralized mechanism (i.e., the controller triggers the protection switchover) can significantly shorten the protection switching time.

8.4.3. Single-domain LSP Restoration

- Pre-planned LSP protection (including shared-mesh restoration):

In pre-planned protection, the protecting LSP is established only in the control plane in the provisioning phase, and will be activated in the data plane once failure occurs.

In the scenario of interworking between GMPLS and centralized controller system, the route of protecting LSP can be computed by the centralized controller system. This takes the advantage of making better use of network resource, especially for the resource sharing in shared-mesh restoration.

- Full LSP rerouting:

In full LSP rerouting, the normal traffic will be switched to an alternate LSP that is fully established only after failure occurrence.

As described in [RFC4872] and [RFC4873], the alternate route can be computed on demand when failure occurrence, or pre-computed and stored before failure occurrence.

In a fully distributed scenario, the pre-computation method offers faster restoration time, but has the risk that the pre-computed alternate route may become out of date due to the changes of the network.

In the scenario of interworking between GMPLS and centralized controller system, the pre-computation of the alternate route could take place in the centralized controller (and may be stored in the controller or the head-end node of the LSP). In this way, any changes in the network can trigger the refreshment of the alternate route by the centralized controller. This makes sure that the alternate route will not become out of date.

8.4.4. Multi-domain LSP Restoration

A working LSP may traverse multiple domains, each of which may or may not support GMPLS distributed control plane.

If all the domains support GMPLS, both the end-to-end rerouting method and the domain segment rerouting method could be used.

If only some domains support GMPLS, the domain segment rerouting method could be used in those GMPLS domains. For other domains which do not support GMPLS, other mechanisms may be used to protect the LSP segments, which are out of scope of this document.

1) End-to-end rerouting:

In this scenario, a failure on the working LSP inside any domain or on the inter-domain links will trigger the end-to-end restoration.

In both pre-planned and full LSP rerouting, the end-to-end protecting LSP could be computed by the centralized controller system, and could be created by the procedure described in Section 8.2. Note that the end-to-end protecting LSP may traverse different domains from the working LSP, depending on the result of multi-domain path computation for the protecting LSP.

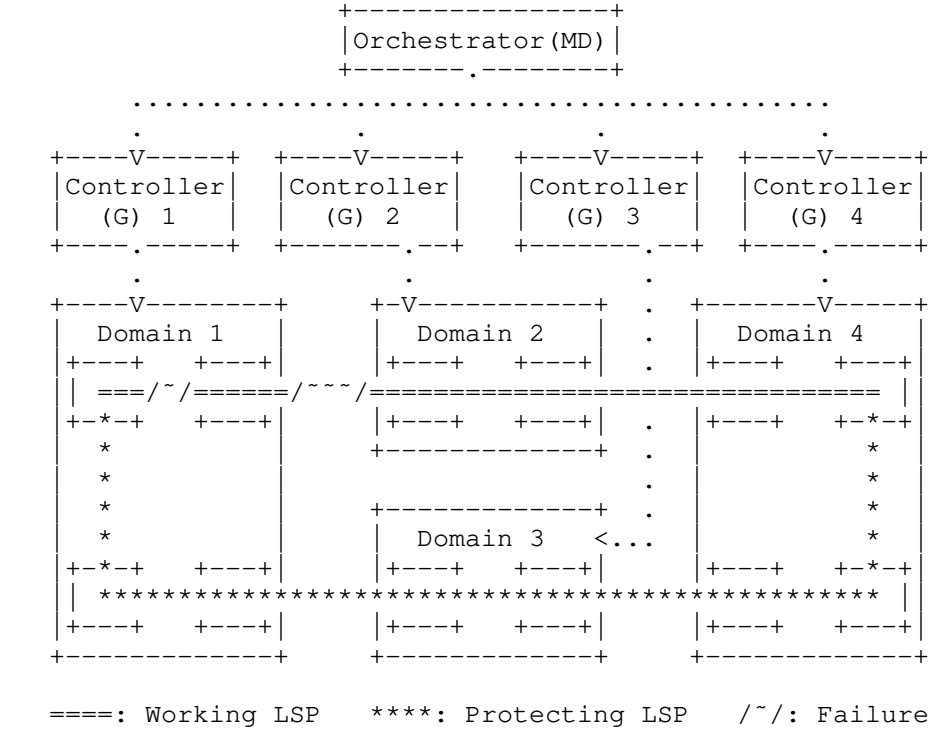


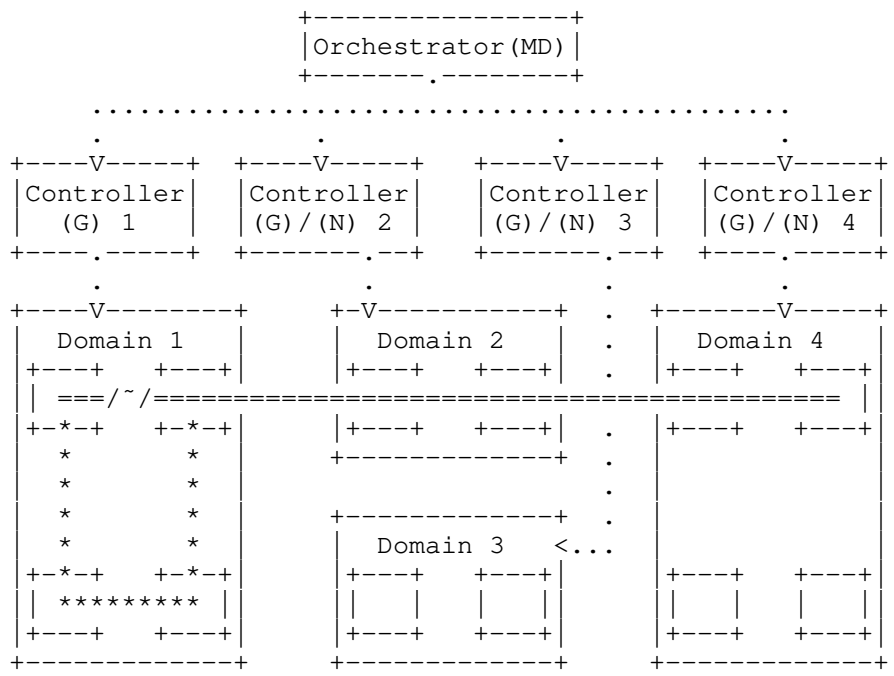
Figure 5: End-to-end restoration

2) Domain segment rerouting:

2.1) Intra-domain rerouting:

If failure occurs on the working LSP segment in a GMPLS domain, the segment rerouting ([RFC4873]) could be used for the working LSP segment in that GMPLS domain. Figure 6 shows an example of intra-domain rerouting.

The intra-domain rerouting of a non-GMPLS domain is out of scope of this document.



====: Working LSP ****: Rerouting LSP segment /~/: Failure

Figure 6: Intra-domain segment rerouting

2.2) Inter-domain rerouting:

If intra-domain segment rerouting failed (e.g., due to lack of resource in that domain), or if failure occurs on the working LSP on an inter-domain link, the centralized controller system may coordinate with other domain(s), to find an alternative path or path segment to bypass the failure, and then trigger the inter-domain rerouting procedure. Note that the rerouting path or path segment may traverse different domains from the working LSP.

The domains involved in the inter-domain rerouting procedure need to be GMPLS domains, which support the RSVP-TE signaling for the creation of rerouting LSP segment.

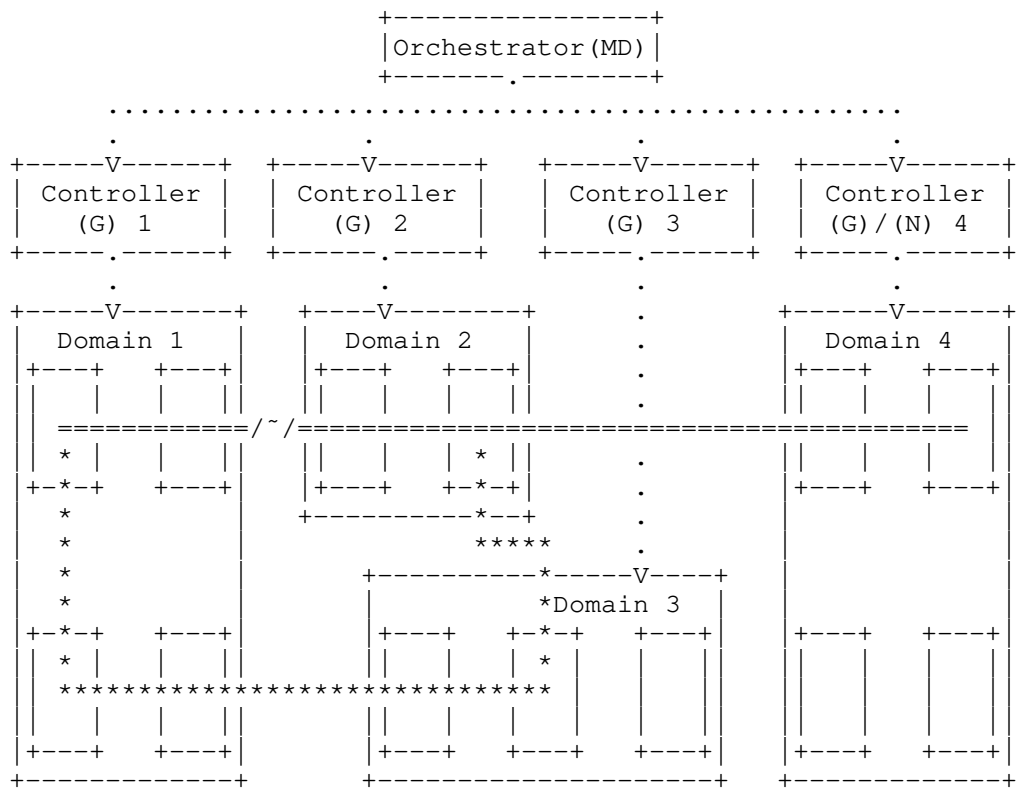
For inter-domain rerouting, the interaction between GMPLS and centralized controller system is needed:

- Report of the result of intra-domain segment rerouting to its Controller(G), and then to the Orchestrator(MD). The former one could be supported by the PCRpt message in [RFC8231], while the latter one could be supported by the MPI interface of ACTN.

- Report of inter-domain link failure to the two Controllers (e.g., Controller(G) 1 and Controller(G) 2 in Figure 7) by which the two ends of the inter-domain link are controlled respectively, and then to the Orchestrator(MD). The former one could be done as described in Section 8.1 of this document, while the latter one could be supported by the MPI interface of ACTN.

- Computation of rerouting path or path segment crossing multi-domains by the centralized controller system (see [PAT-COMP]);

- Creation of rerouting LSP segment in each related domain. The Orchestrator(MD) can send the LSP segment rerouting request to the source Controller(G) (e.g., Controller(G) 1 in Figure 7) via MPI interface, and then the Controller(G) can trigger the creation of rerouting LSP segment through multiple GMPLS domains using GMPLS rerouting signaling. Note that the rerouting LSP segment may traverse a new domain which the working LSP does not traverse (e.g., Domain 3 in Figure 7).



====: Working LSP ****: Rerouting LSP segment /~/: Failure

Figure 7: Inter-domain segment rerouting

8.4.5. Fast Reroute

[RFC4090] defines two methods of fast reroute, the one-to-one backup method and the facility backup method. For both methods:

1) Path computation of protecting LSP:

In Section 6.2 of [RFC4090], the protecting LSP (detour LSP in one-to-one backup, or bypass tunnel in facility backup) could be computed by the Point of Local Repair (PLR) using, for example, Constraint-based Shortest Path First (CSPF) computation. In the scenario of interworking between GMPLS and centralized controller system, the protecting LSP could also be computed by the centralized controller system, as it has the global view of the network topology, resource and information of LSPs.

2) Protecting LSP creation:

In the scenario of interworking between GMPLS and centralized controller system, the Protecting LSP could still be created by the RSVP-TE signaling protocol as described in [RFC4090] and [RFC8271].

In addition, if the protecting LSP is computed by the centralized controller system, the Secondary Explicit Route Object defined in [RFC4873] could be used to explicitly indicate the route of the protecting LSP.

3) Failure detection and traffic switchover:

If a PLR detects that failure occurs, it may significantly shorten the protection switching time by using the distributed mechanisms described in [RFC4090] to switch the traffic to the related detour LSP or bypass tunnel, rather than in a centralized way.

8.5. Controller Reliability

The reliability of the controller is crucial due to its important role in the network. It is essential that if the controller is shut down or disconnected from the network, all currently provisioned services in the network continue to function and carry traffic. In addition, protection switching to pre-established paths should also work. It is desirable to have protection mechanisms, such as redundancy, to maintain full operational control even if one instance of the controller fails. This can be achieved through controller backup or functionality backup. There are several controller backup or federation mechanisms in the literature. It is also more reliable to have function backup in the network element to guarantee performance in the network.

9. Manageability Considerations

Each network entity, including controllers and network elements, should be managed properly and with the relevant trust and security policies applied (see Section 10 of this document), as they will interact with other entities. The manageability considerations in controller hierarchies and network elements still apply, respectively. The overall manageability of the protocols applied in the network should also be a key consideration.

The responsibility of each entity should be clarified. The control of function and policy among different controllers should be consistent via a proper negotiation process.

10. Security Considerations

This document outlines the interworking between GMPLS and controller hierarchies. The security requirements specific to both systems remain applicable. Protocols referenced herein possess security considerations, which must be adhered to, with their core specifications and identified risks detailed earlier in this document.

Security is a critical aspect in both GMPLS and controller-based networks. Ensuring robust security mechanisms in these environments is paramount to safeguard against potential threats and vulnerabilities. Below are expanded security considerations and some relevant IETF RFC references.

- Authentication and Authorization: It is essential to implement strong authentication and authorization mechanisms to control access to the controller from multiple network elements. This ensures that only authorized devices and users can interact with the controller, preventing unauthorized access that could lead to network disruptions or data breaches. Transport Layer Security (TLS) Protocol [RFC8446] and Enrollment over Secure Transport [RFC7030] provide guidelines on secure communication and certificate-based authentication that can be leveraged for these purposes.

- Controller Security: The controller's security is crucial as it serves as the central control point for the network elements. The controller must be protected against various attacks, such as Denial-of-Service (DoS), Man-In-The-Middle (MITM), and unauthorized access. Security mechanisms should include regular security audits, application of security patches, and firewalls and Intrusion Detection/Prevention Systems (IDS/IPS).

- Data Transport Security: Security mechanisms on the controller should also safeguard the underlying network elements against

unauthorized usage of data transport resources. This includes encryption of data in transit to prevent eavesdropping and tampering, as well as ensuring data integrity and confidentiality.

- Secure Protocol Implementation: Protocols used within the GMPLS and controller frameworks must be implemented with security in mind. Known vulnerabilities should be addressed, and secure versions of protocols should be used wherever possible.

Finally, robust network security often depends on Indicators of Compromise (IoCs) to detect, trace, and prevent malicious activities in networks or endpoints. These are described in [RFC9424] along with the fundamentals, opportunities, operational limitations, and recommendations for IoC use.

11. IANA Considerations

This document requires no IANA actions.

12. References

12.1. Normative References

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[RFC3473] Berger, L., Ed., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions", RFC 3473, January 2003.

[RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, September 2003.

[RFC3945] Mannie, E., Ed., "Generalized Multi-Protocol Label Switching (GMPLS) Architecture", RFC 3945, October 2004.

[RFC4090] Pan, P., Ed., Swallow, G., Ed., and A. Atlas, Ed., "Fast Reroute Extensions to RSVP-TE for LSP Tunnels", RFC 4090, May 2005.

[RFC4203] Kompella, K., Ed. and Y. Rekhter, Ed., "OSPF Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)", RFC 4203, October 2005.

[RFC4206] Kompella, K. and Rekhter Y., "Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE)", RFC 4206, October 2005.

[RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, August 2006.

[RFC4872] Lang, J., Ed., Rekhter, Y., Ed., and D. Papadimitriou, Ed., "RSVP-TE Extensions in Support of End-to-End Generalized Multi-Protocol Label Switching (GMPLS) Recovery", RFC 4872, May 2007.

[RFC4873] Berger, L., Bryskin, I., Papadimitriou, D., and A. Farrel, "GMPLS Segment Recovery", RFC 4873, May 2007.

[RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic Engineering", RFC 5305, October 2008.

[RFC5307] Kompella, K., Ed. and Y. Rekhter, Ed., "IS-IS Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)", RFC 5307, October 2008.

[RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, March 2009.

[RFC6001] Papadimitriou D., Vigoureux M., Shiomoto K., Brungard D. and Le Roux JL., "Generalized MPLS (GMPLS) Protocol Extensions for Multi-Layer and Multi-Region Networks (MLN/MRN)", RFC 6001, October 2010.

[RFC6107] Shiomoto K. and Farrel A., "Procedures for Dynamically Signaled Hierarchical Label Switched Paths", RFC 6107, February 2011.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder J., Bierman A., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC7030] Pritikin, M., Yee, P. and Harkins, D., "Enrollment over Secure Transport", RFC7030, October 2013.

[RFC7074] Berger, L. and J. Meuric, "Revised Definition of the GMPLS Switching Capability and Type Fields", RFC 7074, November 2013.

[RFC7491] King, D., Farrel, A., "A PCE-Based Architecture for Application-Based Network Operations", RFC7491, March 2015.

[RFC7926] Farrel, A., Drake, J., Bitar, N., Swallow, G., Ceccarelli, D. and Zhang, X., "Problem Statement and Architecture for Information Exchange between Interconnected Traffic-Engineered Networks", RFC7926, July 2016.

[RFC7950] Bjorklund, M., "The YANG 1.1 Data Modeling Language", RFC7950, August 2016.

[RFC8040] Bierman, A., Bjorklund, M., Watsen, K., "RESTCONF Protocol", RFC 8040, January 2017.

[RFC8271] Taillon M., Saad T., Gandhi R., Ali Z. and Bhatia M., "Updates to the Resource Reservation Protocol for Fast Reroute of Traffic Engineering GMPLS Label Switched Paths", RFC 8271, October 2017.

[RFC8282] Oki E., Takeda T., Farrel A. and Zhang F., "Extensions to the Path Computation Element Communication Protocol (PCEP) for Inter-Layer MPLS and GMPLS Traffic Engineering", RFC 8282, December 2017.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC8446, August 2018.

[RFC8453] Ceccarelli, D. and Y. Lee, "Framework for Abstraction and Control of Traffic Engineered Networks", RFC 8453, August 2018.

[RFC8795] Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., Gonzalez De Dios, O., "YANG Data Model for Traffic Engineering (TE) Topologies", RFC8795, August 2020.

[RFC9424] Paine, K., Whitehouse, O., Sellwood, J. and Shaw, A., "Indicators of Compromise (IoCs) and Their Role in Attack Defence", RFC9424, August 2023.

12.2. Informative References

[RFC3471] Berger, L., Ed., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description", RFC 3471, January 2003.

[RFC4202] Kompella, K., Ed. and Y. Rekhter, Ed., "Routing Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)", RFC 4202, October 2005.

[RFC4204] Lang, J., Ed., "Link Management Protocol (LMP)", RFC 4204, October 2005.

[RFC4426] Lang, J., Ed., Rajagopalan, B., Ed., and D. Papadimitriou, Ed., "Generalized Multi-Protocol Label witching (GMPLS) Recovery Functional Specification", RFC 4426, March 2006.

[RFC5150] Ayyangar, A., Kompella, K., Vasseur, J.P., Farrel, A., "Label Switched Path Stitching with Generalized Multiprotocol Label Switching Traffic Engineering (GMPLS TE)", RFC 5150, February, 2008.

[RFC5212] Shiomoto K., Papadimitriou D., Le Roux JL., Vigoureux M. and Brungard D., "Requirements for GMPLS-Based Multi-Region and Multi-Layer Networks (MRN/MLN)", RFC 5212, July 2008.

[RFC5623] Oki E., Takeda T., Le Roux JL. and Farrel A., "Framework for PCE-Based Inter-Layer MPLS and GMPLS Traffic Engineering", RFC 5623, September 2009.

[RFC5654] Niven-Jenkins B., Ed., Brungard D., Ed., Betts M., Ed., Sprecher N., Ueno S., "Requirements of an MPLS Transport Profile", RFC 5654, September 2009.

[RFC7138] Ceccarelli, D., Ed., Zhang, F., Belotti, S., Rao, R., and J. Drake, "Traffic Engineering Extensions to OSPF for GMPLS Control of Evolving G.709 Optical Transport Networks", RFC 7138, March 2014.

[RFC7139] Zhang, F., Ed., Zhang, G., Belotti, S., Ceccarelli, D., and K. Pithewan, "GMPLS Signaling Extensions for Control of Evolving G.709 Optical Transport Networks", RFC 7139, March 2014.

[RFC7271] Ryoo, J., Ed., Gray, E., Ed., van Helvoort, H., D'Alessandro, A., Cheung, T., and Osborne, E., "MPLS Transport Profile (MPLS-TP) Linear Protection to Match the Operational Expectations of Synchronous Digital Hierarchy, Optical Transport Network, and Ethernet Transport Network Operators", RFC 7271, June 2014.

[RFC7688] Lee, Y., Ed. and G. Bernstein, Ed., "GMPLS OSPF Enhancement for Signal and Network Element Compatibility for Wavelength Switched Optical Networks", RFC 7688, November 2015.

[RFC7689] Bernstein, G., Ed., Xu, S., Lee, Y., Ed., Martinelli, G., and H. Harai, "Signaling Extensions for Wavelength Switched Optical Networks", RFC 7689, November 2015.

[RFC7792] Zhang, F., Zhang, X., Farrel, A., Gonzalez de Dios, O., and D. Ceccarelli, "RSVP-TE Signaling Extensions in Support of Flexi-Grid Dense Wavelength Division Multiplexing (DWDM) Networks", RFC 7792, March 2016.

[RFC8231] Crabbe, E., Minei, I., Medved, J., and R. Varga, "Path Computation Element Communication Protocol (PCEP) Extensions for Stateful PCE", RFC 8231, September 2017.

[RFC8234] Ryoo, J., Cheung, T., van Helvoort, H., Busi, I. and Wen G., "Updates to MPLS Transport Profile (MPLS-TP) Linear Protection in Automatic Protection Switching (APS) Mode", RFC 8234, August 2017.

[RFC8281] Crabbe, E., Minei, I., Sivabalan, S., and R. Varga, "PCEP Extensions for PCE-initiated LSP Setup in a Stateful PCE Model", RFC 8281, October 2017.

[RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., Liu, X., "A YANG Data Model for Network Topologies", RFC 8345, March 2018.

[RFC8363] Zhang, X., Zheng, H., Casellas, R., Dios, O., and D. Ceccarelli, "GMPLS OSPF-TE Extensions in support of Flexi-grid DWDM networks", RFC8363, February 2017.

[PAT-COMP] Busi, I., Belotti, S., Lopez, V., Gonzalez de Dios, O., Sharma, A., Shi, Y., Vilalta, R., Setheraman, K., "Yang model for requesting Path Computation", draft-ietf-teas-yang-path-computation, work in progress.

[PCEP-LS] Dhody, D., Lee, Y., Ceccarelli, D., "PCEP Extensions for Distribution of Link-State and TE Information", draft-dhodylee-pce-pcep-ls, work in progress.

[TE-Tunnel] Saad, T. et al., "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te, work in progress.

[sPCE-ID] Dugeon, O. et al., "PCEP Extension for Stateful Inter-Domain Tunnels", draft-ietf-pce-stateful-interdomain, work in progress.

[G.808.1] ITU-T, "Generic protection switching - Linear trail and subnetwork protection", G.808.1, May 2014.

13. Contributors

Xianlong Luo
Huawei Technologies
G1, Huawei Xiliu Beipo Village, Songshan Lake
Dongguan
Guangdong, 523808 China
Email: luoxianlong@huawei.com

Sergio Belotti
Nokia
Email: sergio.belotti@nokia.com

14. Authors' Addresses

Haomian Zheng
Huawei Technologies
H1, Huawei Xiliu Beipo Village, Songshan Lake
Dongguan
Guangdong, 523808 China
Email: zhenghaomian@huawei.com

Yunbin Xu
CAICT
Email: xuyunbin@caict.ac.cn

Yang Zhao
China Mobile
Email: zhaoyangyjy@chinamobile.com

Dieter Beller
Nokia
Email: Dieter.Beller@nokia.com

Yi Lin
Huawei Technologies
H1, Huawei Xiliu Beipo Village, Songshan Lake
Dongguan
Guangdong, 523808 China
Email: yi.lin@huawei.com

Acknowledgements

The authors would like to thank Jim Guichard, Area Director of IETF Routing Area, Vishnu Pavan Beeram, Chair of TEAS WG, Jia He and Stewart Bryant, rtgdir reviewers, Thomas Fossati, Gen-ART reviewer, Yingzhen Qu, opsdir reviewer, David Mandelberg, secdir reviewer, and David Dong, IANA Services Sr. Specialist, for their reviews and comments on this document.