

Workgroup: netmod  
Internet-Draft:  
draft-ietf-netmod-acl-extensions-06  
Published: 30 January 2024  
Intended Status: Standards Track  
Expires: 2 August 2024  
Authors: O. G. D. Dios    S. Barguil    M. Boucadair    Q. Wu  
          Telefonica        Telefonica    Orange         Huawei  
**Extensions to the Access Control Lists (ACLs) YANG Model**

## Abstract

RFC 8519 defines a YANG data model for Access Control Lists (ACLs). This document discusses a set of extensions that fix many of the limitations of the ACL model as initially defined in RFC 8519.

The document also defines IANA-maintained modules for ICMP types and IPv6 extension headers.

## Discussion Venues

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

Discussion of this document takes place on the Network Modeling Working Group mailing list (netmod@ietf.org), which is archived at <https://mailarchive.ietf.org/arch/browse/netmod/>.

Source for this draft and an issue tracker can be found at <https://github.com/boucadair/enhanced-acl-netmod>.

## 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 August 2024.

## Copyright Notice

Copyright (c) 2024 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

- [1. Introduction](#)
  - [1.1. Editorial Note \(To be removed by RFC Editor\)](#)
- [2. Terminology](#)
- [3. Overall Structure of The Enhanced ACL Module](#)
  - [3.1. Tree Structure](#)
  - [3.2. Defined Sets](#)
  - [3.3. IPv6 Extension Headers](#)
  - [3.4. TCP Flags Handling](#)
  - [3.5. Fragments Handling](#)
  - [3.6. Payload-based Filtering](#)
  - [3.7. Match on MPLS Headers](#)
  - [3.8. VLAN Filtering](#)
  - [3.9. Instance Service Identifier \(I-SID\) Filtering](#)
  - [3.10. Additional Actions](#)
- [4. Enhanced ACL YANG Module](#)
- [5. Security Considerations](#)
- [6. IANA Considerations](#)
  - [6.1. URI Registrations](#)
  - [6.2. YANG Module Name Registrations](#)
  - [6.3. Considerations for IANA-Maintained Modules](#)
    - [6.3.1. ICMPv4 Types IANA Module](#)
    - [6.3.2. ICMPv6 Types IANA Module](#)
    - [6.3.3. IPv6 Extension Header Types IANA Module](#)
- [7. References](#)
  - [7.1. Normative References](#)
  - [7.2. Informative References](#)
- [Appendix A. ICMPv4 Types](#)
  - [A.1. XSLT Template to Generate The ICMPv4 Types IANA-Maintained Module](#)
  - [A.2. Initial Version of the The ICMPv4 Types IANA-Maintained Module](#)

## [Appendix B. ICMPv6 Types](#)

[B.1. XSLT Template to Generate The ICMPv6 Types IANA-Maintained Module](#)

[B.2. Initial Version of the The ICMPv6 Types IANA-Maintained Module](#)

## [Appendix C. IPv6 Extension Header Types](#)

[C.1. XSLT Template to Generate The IPv6 Extension Header Types IANA-Maintained Module](#)

[C.2. Initial Version of The IPv6 Extension Header Types IANA-Maintained Module](#)

## [Appendix D. Problem Statement & Gap Analysis](#)

[D.1. Suboptimal Configuration: Lack of Support for Lists of Prefixes](#)

[D.2. Manageability: Impossibility to Use Aliases or Defined Sets](#)

[D.3. Bind ACLs to Devices, Not Only Interfaces](#)

[D.4. Partial or Lack of IPv4/IPv6 Fragment Handling](#)

[D.5. Suboptimal TCP Flags Handling](#)

[D.6. Rate-Limit Action](#)

[D.7. Payload-based Filtering](#)

[D.8. Reuse the ACLs Content Across Several Devices](#)

[D.9. Match MPLS Headers](#)

## [Appendix E. Examples](#)

[E.1. TCP Flags Handling](#)

[E.2. Fragments Handling](#)

[E.3. VLAN Filtering](#)

[E.4. ISID Filtering](#)

[E.5. Rate-Limit](#)

## [Acknowledgements](#)

## [Authors' Addresses](#)

### **1. Introduction**

[[RFC8519](#)] defines Access Control Lists (ACLs) as a user-ordered set of filtering rules. The model targets the configuration of the filtering behavior of a device. However, the model structure, as defined in [[RFC8519](#)], suffers from a set of limitations. This document describes these limitations and specifies an enhanced ACL structure ([Section 4](#)). The motivation of such enhanced ACL structure is discussed in detail in [Appendix D](#).

When managing ACLs, it is common for network operators to group match elements in pre-defined sets. The consolidation into group matches allows for reducing the number of rules, especially in large scale networks. If, for example, it is needed to find a match against 100 IP addresses (or prefixes), a single rule will suffice rather than creating individual Access Control Entries (ACEs) for each IP address (or prefix). In doing so, implementations would optimize the performance of matching lists vs multiple rules matching.

The enhanced ACL structure ([Section 4](#)) is also meant to facilitate the management of network operators. Instead of entering the IP address or port number literals, using user-named lists decouples the creation of the rule from the management of the sets. Hence, it is possible to remove/add entries to the list without redefining the (parent) ACL rule.

In addition, the notion of Access Control List (ACL) and defined sets is generalized so that it is not device-specific as per [[RFC8519](#)]. ACLs and defined sets may be defined at network/administrative domain level and associated to devices. This approach facilitates the reusability across multiple network elements. For example, managing the IP prefix sets from a network level makes it easier to maintain by the security groups.

Network operators maintain sets of IP prefixes that are related to each other, e.g., deny-lists or accept-lists that are associated with those provided by a VPN customer. These lists are maintained and manipulated by security expert teams.

Note that ACLs are used locally in devices but are triggered by other tools such as DDoS mitigation [[RFC9132](#)] or BGP Flow Spec [[RFC8955](#)] [[RFC8956](#)]. Therefore, supporting means to easily map to the filtering rules conveyed in messages triggered by these tools is valuable from a network operation standpoint.

The enhanced ACL module ([Section 4](#)) conforms to the Network Management Datastore Architecture (NMDA) defined in [[RFC8342](#)].

A set of examples to illustrate the use of the enhanced ACL module are provided in [Appendix E](#).

The document also defines IANA-maintained modules for ICMP types and IPv6 extension headers. The design of the modules adheres to the recommendations in [[I-D.ietf-netmod-rfc8407bis](#)]. The templates to generate the modules are available in [Appendix A.1](#), [Appendix B.1](#), and [Appendix C.1](#). Readers should refer to the IANA websites "IANA\_ICMPv4\_YANG\_URL", "IANA\_ICMPv6\_YANG\_URL", and "IANA\_IPV6\_YANG\_URL" to retrieve the latest version of these IANA-maintained modules.

### **1.1. Editorial Note (To be removed by RFC Editor)**

Note to the RFC Editor: This section is to be removed prior to publication.

This document contains placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed.

(1) Please apply the following replacements:

XXXX --> the assigned RFC number for this I-D 2022-10-24 --> the actual date of the publication of this document

(2) The modules are provided in [Appendix A.2](#), [Appendix B.2](#), and [Appendix C.2](#) for the users convenience before publication as RFC. Please remove these appendices from the final RFC.

(3) Please update the following references:

\*IANA\_ICMPv4\_YANG\_URL --> The URL to retrieve the latest version of the IANA-maintained ICMPv4 module.

\*IANA\_ICMPv6\_YANG\_URL --> The URL to retrieve the latest version of the IANA-maintained ICMPv6 module.

\*IANA\_IPV6\_YANG\_URL --> The URL to retrieve the latest version of the IPv6 Extension Header Types IANA module.

## 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.

The terminology for describing YANG modules is defined in [[RFC7950](#)]. The meaning of the symbols in the tree diagrams is defined in [[RFC8340](#)].

In addition to the terms defined in [[RFC8519](#)], this document makes use of the following term:

Defined set: :Refers to reusable description of one or multiple information elements (e.g., IP address, IP prefix, port number, or ICMP type).

## 3. Overall Structure of The Enhanced ACL Module

### 3.1. Tree Structure

[Figure 1](#) shows the full tree of the enhanced ACL module ([Section 4](#)):

```

module: ietf-acl-enh
  +--rw defined-sets
  | +--rw ipv4-prefix-sets
  | | +--rw prefix-set* [name]
  | |   +--rw name          string
  | |   +--rw description?  string
  | |   +--rw prefix*       inet:ipv4-prefix
  | +--rw ipv6-prefix-sets
  | | +--rw prefix-set* [name]
  | |   +--rw name          string
  | |   +--rw description?  string
  | |   +--rw prefix*       inet:ipv6-prefix
  | +--rw port-sets
  | | +--rw port-set* [name]
  | |   +--rw name          string
  | |   +--rw port* [id]
  | |     +--rw id              string
  | |     +--rw (port)?
  | |       +--:(port-range-or-operator)
  | |         +--rw port-range-or-operator
  | |         +---u packet-fields:port-range-or-operator
  | +--rw protocol-sets
  | | +--rw protocol-set* [name]
  | |   +--rw name          string
  | |   +--rw protocol*     union
  | +--rw icmpv4-type-sets
  | | +--rw icmpv4-type-set* [name]
  | |   +--rw name          string
  | |   +--rw types* [type]
  | |     +---u icmpv4-header-fields
  | +--rw icmpv6-type-sets
  |   +--rw icmpv6-type-set* [name]
  |     +--rw name          string
  |     +--rw types* [type]
  |       +---u icmpv6-header-fields
+--rw aliases
  +--rw alias* [name]
  +--rw name          string
  +---u alias

augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches:
  +--rw (payload)?
  | +--:(prefix-pattern)
  |   +--rw prefix-pattern {match-on-payload}?
  |   +---u payload
+--rw (alias)?
  | +--:(alias-name)
  |   +--rw alias-name*          alias-ref
+--rw (mpls)?

```

```

    +--:(mpls-values)
      +--rw mpls-values {match-on-mpls}?
        +---u mpls-match-parameters-config
augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches/acl:l2:
  +--rw vlan-filter {match-on-vlan-filter}?
    | +--rw frame-type?          string
    | +--rw (vlan-type)?
    |   +--:(range)
    |     | +--rw lower-vlan    uint16
    |     | +--rw upper-vlan   uint16
    |     +--:(operator)
    |       +--rw operator?    packet-fields:operator
    |       +--rw vlan*        uint16
  +--rw isid-filter {match-on-isid-filter}?
    +--rw (isid-type)?
      +--:(range)
      | +--rw lower-isid      uint16
      | +--rw upper-isid     uint16
      +--:(operator)
        +--rw operator?      packet-fields:operator
        +--rw isid*          uint16
augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches/acl:l3
  /acl:ipv4/acl:ipv4:
  +--rw ipv4-fragment
    | +---u fragment-fields
  +--rw source-ipv4-prefix-list?    ipv4-prefix-set-ref
  +--rw destination-ipv4-prefix-list?  ipv4-prefix-set-ref
  +--rw next-header-set?              protocol-set-ref
augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches/acl:l3
  /acl:ipv6/acl:ipv6:
  +--rw ipv6-fragment
    | +---u fragment-fields
  +--rw source-ipv6-prefix-list?    ipv6-prefix-set-ref
  +--rw destination-ipv6-prefix-list?  ipv6-prefix-set-ref
  +--rw protocol-set?                protocol-set-ref
  +--rw extension-header?
    iana-ipv6-ext-types:ipv6-extension-header-type
augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches/acl:l4
  /acl:tcp/acl:tcp:
  +--rw flags-bitmask
    | +---u tcp-flags
  +--rw source-tcp-port-set?        port-set-ref
  +--rw destination-tcp-port-set?   port-set-ref
augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches/acl:l4
  /acl:udp/acl:udp:
  +--rw source-udp-port-set?        port-set-ref
  +--rw destination-udp-port-set?   port-set-ref
augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:matches/acl:l4
  /acl:icmp/acl:icmp:

```

```
+-rw icmpv4-set?    icmpv4-type-set-ref
+-rw icmpv6-set?    icmpv6-type-set-ref
augment /acl:acls/acl:acl/acl:aces/acl:ace/acl:actions:
+--u acl-complementary-actions
+-rw rate-limit?    decimal64
```

Figure 1: Enhanced ACL Tree Structure

[Figure 2](#) shows the reusable groupings that are defined in the enhanced ACL module:



```

grouping tcp-flags:
  +-- (mode)?
    +---:(explicit)
      | +-- operator?          operator
      | +-- explicit-tcp-flag* identityref
    +---:(builtin)
      +-- bitmask?           uint16
grouping fragment-fields:
  +-- operator?  operator
  +-- type?      fragment-type
grouping mpls-match-parameters-config:
  +-- traffic-class?      uint8
  +-- label-position?     identityref
  +-- upper-label-range?  rt-types:mpls-label
  +-- lower-label-range?  rt-types:mpls-label
  +-- label-block-name?   string
  +-- ttl-value?          uint8
grouping payload:
  +-- offset?            identityref
  +-- offset-end?        uint64
  +-- operator?          operator
  +-- prefix?            binary
grouping alias:
  +-- vlan*              uint16
  +-- prefix*            inet:ip-prefix
  +-- port-range* [lower-port]
    | +-- lower-port      inet:port-number
    | +-- upper-port?     inet:port-number
  +-- protocol*          uint8
  +-- fqdn*              inet:domain-name
  +-- uri*               inet:uri
grouping icmpv4-header-fields:
  +-- type?              iana-icmpv4-types:icmpv4-type
  +-- code?              uint8
  +-- rest-of-header?    binary
grouping icmpv6-header-fields:
  +-- type?              iana-icmpv6-types:icmpv6-type
  +-- code?              uint8
  +-- rest-of-header?    binary
grouping acl-complementary-actions:
  +-- log-action
    | +-- log-type?       identityref
    | +-- log-id?         string
  +-- counter-action
    +-- counter-type?     identityref
    +-- counter-name*     string

```

Figure 2: Enhanced ACL Groupings

### 3.2. Defined Sets

The augmented ACL structure includes several containers to manage reusable sets of elements that can be matched in an ACL entry. Each set is uniquely identified by a name and can be called from the relevant entry. The following sets are defined:

- \***IPv4 prefix set:** It contains a list of IPv4 prefixes. A match will be considered if the IP address (source or destination, depending on the ACL entry) is contained in any of the prefixes.
- \***IPv6 prefix set:** It contains a list of IPv6 prefixes. A match will be considered if the IP address (source or destination, depending on the ACL entry) is contained in any of the prefixes.
- \***Port sets:** It contains a list of port numbers to be used in TCP/UDP entries. The port numbers can be individual port numbers, a range of ports, and an operation.
- \***Protocol sets:** It contains a list of protocol values. Each protocol can be identified either by a number (e.g., 17) or a name (e.g., UDP).
- \***ICMP sets:** It contains a list of ICMPv4 [[RFC0792](#)] or ICMPv6 [[RFC4443](#)] types, each of them identified by a type value, optionally the code and the rest of the header.
- \***Aliases:** An alias is defined by a combination of various parameters (e.g., IP prefix, protocol, port number, or VLAN). Sets of aliases can be defined and referred to in match criteria.

### 3.3. IPv6 Extension Headers

The module can be used to manage ACLs that require matching against IPv6 extension headers. To that aim, a new IANA-maintained module for IPv6 extension header types is defined in this document.

### 3.4. TCP Flags Handling

The augmented ACL structure ([Figure 1](#)) includes a new leaf 'flags-bitmask' to better handle TCP flags [[RFC9293](#)].

Clients that support both 'flags-bitmask' and 'flags' matching fields **MUST NOT** set these fields in the same request.

### 3.5. Fragments Handling

The augmented ACL structure ([Figure 1](#)) includes a new leaf 'fragment' to better handle fragments.

Clients that support both 'fragment' and 'flags' matching fields **MUST NOT** set these fields in the same request.

### 3.6. Payload-based Filtering

Some transport protocols use existing protocols (e.g., TCP or UDP) as substrate. The match criteria for such protocols may rely upon the 'protocol' under 'l3', TCP/UDP match criteria, part of the TCP/UDP payload, or a combination thereof. A new feature, called "match-on-payload", is defined in the document. This can be used, for example, for QUIC [[RFC9000](#)] or for tunneling protocols.

### 3.7. Match on MPLS Headers

The enhanced ACL module can be used to create rules to match against MPLS fields of a packet. The MPLS header defined in [[RFC3032](#)] and [[RFC5462](#)] contains the following fields:

\*Traffic Class: 3 bits 'EXP' renamed to 'Traffic Class Field.'

\*Label Value: A 20-bit field that carries the actual value of the MPLS Label.

\*TTL: An eight-bit field that is used to encode a time-to-live value.

The augmented ACL structure ([Figure 1](#)) allows an operator to configure ACLs that match based upon the following data nodes:

\*"traffic-class"

\*"label-position" (e.g., top, bottom)

\*"upper-label-range"

\*"lower-label-range"

\*"label-block-name"

\*"ttl-value"

### 3.8. VLAN Filtering

Being able to filter all packets that are bridged within a VLAN or that are routed into or out of a bridge domain is part of the VPN

control requirements derived of the EVPN definition in [\[RFC7209\]](#). All packets that are bridged within a VLAN or that are routed into or out of a VLAN can be captured, forwarded, translated, or discarded based on the network policy.

### **3.9. Instance Service Identifier (I-SID) Filtering**

Provider backbone bridging (PBB) was originally defined as Virtual Bridged Local Area Networks [\[IEEE-802-1ah\]](#) standard. However, instead of multiplexing VLANs, PBB duplicates the MAC layer of the customer frame and separates it from the provider domain, by encapsulating it in a 24-bit instance service identifier (I-SID). This provides more transparency between the customer network and the provider network.

The I-component forms the customer or access facing interface or routing instance. The I-component is responsible for mapping customer Ethernet traffic to the appropriate I-SID. It is mandatory to configure the default service identifier in the network.

Being able to filter by I-component Service identifier is a feature of the EVNP-PBB configuration.

### **3.10. Additional Actions**

In order to support rate-limiting (see [Appendix D.6](#)), a new action called "rate-limit" is defined. Also, the model supports new actions to complement existing ones: Log ('log-action') and write a counter ('counter-action'). The version of the module defined in this document supports only local actions.

## **4. Enhanced ACL YANG Module**

This model imports types from [\[RFC6991\]](#), [\[RFC8519\]](#), and [\[RFC8294\]](#).

```
<CODE BEGINS> file "ietf-acl-enh@2022-10-24.yang"
```

```
module ietf-acl-enh {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-acl-enh";
  prefix acl-enh;

  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types";
  }
  import ietf-access-control-list {
    prefix acl;
    reference
      "RFC 8519: YANG Data Model for Network Access
        Control Lists (ACLs), Section 4.1";
  }
  import ietf-packet-fields {
    prefix packet-fields;
    reference
      "RFC 8519: YANG Data Model for Network Access
        Control Lists (ACLs), Section 4.2";
  }
  import ietf-routing-types {
    prefix rt-types;
    reference
      "RFC 8294: Common YANG Data Types for the Routing Area";
  }
  import iana-icmpv4-types {
    prefix iana-icmpv4-types;
    reference
      "RFC XXXX: Extensions to the Access Control Lists (ACLs)
        YANG Model";
  }
  import iana-icmpv6-types {
    prefix iana-icmpv6-types;
    reference
      "RFC XXXX: Extensions to the Access Control Lists (ACLs)
        YANG Model";
  }
  import iana-ipv6-ext-types {
    prefix iana-ipv6-ext-types;
    reference
      "RFC XXXX: Extensions to the Access Control Lists (ACLs)
        YANG Model";
  }

  organization
```

```
"IETF NETMOD Working Group";
contact
  "WG Web:  https://datatracker.ietf.org/wg/netmod/
  WG List:  mailto:netmod@ietf.org

  Author:   Mohamed Boucadair
            mailto:mohamed.boucadair@orange.com
  Author:   Samier Barguil
            mailto:samier.barguilgiraldo.ext@telefonica.com
  Author:   Oscar Gonzalez de Dios
            mailto:oscar.gonzalezdedios@telefonica.com";
description
  "This module contains YANG definitions for enhanced ACLs.

  Copyright (c) 2023 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 2022-10-24 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: Extensions to the Access Control Lists (ACLs)
    YANG Model";
}

feature match-on-payload {
  description
    "Match based on a pattern is supported.";
}

feature match-on-vlan-filter {
  description
    "Match based on a VLAN range of vlan list is supported.";
}

feature match-on-isid-filter {
  description
    "Match based on a ISID range of vlan list is supported.";
}
```

```
feature match-on-alias {
  description
    "Match based on aliases.";
}

feature match-on-mpls {
  description
    "Match based on MPLS headers.";
}

identity offset-type {
  description
    "Base identity for payload offset type.";
}

identity layer2 {
  base offset-type;
  description
    "The offset starts at the beginning of the Data Link layer
    header.";
}

identity layer3 {
  base offset-type;
  description
    "The offset starts at the beginning of the IP header.";
}

identity layer4 {
  base offset-type;
  description
    "The offset start right after the IP header. This can be
    typically the beginning of transport header (e.g., TCP
    or UDP).";
}

identity payload {
  base offset-type;
  description
    "The offset start right after the end of the transport
    payload. For example, this represents the beginning of the
    TCP data right after any TCP options or the beginning of
    the UDP payload right after the UDP header.";
}

identity tcp-flag {
  description
    "Base Identity for the TCP Flags.";
  reference
```

```
    "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
}

identity ack {
    base tcp-flag;
    description
        "Acknowledgment TCP flag bit.";
    reference
        "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
}

identity syn {
    base tcp-flag;
    description
        "Synchronize sequence numbers.";
    reference
        "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
}

identity fin {
    base tcp-flag;
    description
        "No more data from the sender.";
    reference
        "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
}

identity urg {
    base tcp-flag;
    description
        "Urgent pointer TCP flag bit.";
    reference
        "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
}

identity psh {
    base tcp-flag;
    description
        "The Push function flag is similar to the URG flag and tells
        the receiver to process these packets as they are received
        instead of buffering them.";
    reference
        "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
}

identity rst {
    base tcp-flag;
    description
        "Reset TCP flag bit.";
```



```
reference
  "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
}

identity ece {
  base tcp-flag;
  description
    "ECN-Echo TCP flag bit.";
  reference
    "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
}

identity cwr {
  base tcp-flag;
  description
    "Congestion Window Reduced flag bit";
  reference
    "RFC 9293: Transmission Control Protocol (TCP), Section 3.1";
}

identity mpls-acl-type {
  base acl:acl-base;
  description
    "An ACL that matches on fields from the MPLS header.";
}

identity label-position {
  description
    "Base identity for deriving MPLS label position.";
}

identity top {
  base label-position;
  description
    "Top of the label stack.";
}

identity bottom {
  base label-position;
  description
    "Bottom of the label stack.";
}

identity log-types {
  description
    "Base identity for deriving the Log actions.";
}

identity local-log {
  base log-types;
```

```

description
    "A local log is used to record the ACL results.";
}

identity counter-type {
    description
        "Base identity for deriving the counter actions.";
}

identity counter-name {
    base counter-type;
    description
        "Identity for counter name to be updated based on
        the ACL match actions.";
}

typedef operator {
    type bits {
        bit not {
            position 0;
            description
                "If set, logical negation of operation.";
        }
        bit match {
            position 1;
            description
                "Match bit. This is a bitwise match operation defined as
                '(data & value) == value'.";
        }
        bit any {
            position 2;
            description
                "Any bit. This is a match on any of the bits in bitmask.
                It evaluates to 'true' if any of the bits in the
                value mask are set in the data, i.e.,
                '(data & value) != 0'.";
        }
    }
}
description
    "Specifies how to apply the defined bitmask.
    'any' and 'match' bits must not be set simultaneously.";
}

typedef fragment-type {
    type bits {
        bit df {
            position 0;
            description

```

```

        "Don't fragment bit for IPv4.
        Must be set to 0 when it appears in an IPv6 filter.";
    }
    bit isf {
        position 1;
        description
            "Is a fragment.";
    }
    bit ff {
        position 2;
        description
            "First fragment.";
    }
    bit lf {
        position 3;
        description
            "Last fragment.";
    }
}
description
    "Different fragment types to match against.";
}

typedef ipv4-prefix-set-ref {
    type leafref {
        path "/acl-enh:defined-sets/acl-enh:ipv4-prefix-sets"
            + "/acl-enh:prefix-set/acl-enh:name";
    }
    description
        "Defines a reference to an IPv4 prefix set.";
}

typedef ipv6-prefix-set-ref {
    type leafref {
        path "/acl-enh:defined-sets/acl-enh:ipv6-prefix-sets"
            + "/acl-enh:prefix-set/acl-enh:name";
    }
    description
        "Defines a reference to an IPv6 prefix set.";
}

typedef port-set-ref {
    type leafref {
        path "/acl-enh:defined-sets/acl-enh:port-sets"
            + "/acl-enh:port-set/acl-enh:name";
    }
    description
        "Defines a reference to a port set.";
}

```

```

typedef protocol-set-ref {
  type leafref {
    path "/acl-enh:defined-sets/acl-enh:protocol-sets"
      + "/acl-enh:protocol-set/acl-enh:name";
  }
  description
    "Defines a reference to a protocol set.";
}

typedef icmpv4-type-set-ref {
  type leafref {
    path "/acl-enh:defined-sets/acl-enh:icmpv4-type-sets"
      + "/acl-enh:icmpv4-type-set/acl-enh:name";
  }
  description
    "Defines a reference to an ICMPv4 type set.";
}

typedef icmpv6-type-set-ref {
  type leafref {
    path "/acl-enh:defined-sets/acl-enh:icmpv6-type-sets"
      + "/acl-enh:icmpv6-type-set/acl-enh:name";
  }
  description
    "Defines a reference to an ICMPv6 type set.";
}

typedef alias-ref {
  type leafref {
    path "/acl-enh:aliases/acl-enh:alias/acl-enh:name";
  }
  description
    "Defines a reference to an alias.";
}

grouping tcp-flags {
  description
    "Operations on TCP flags.";
  choice mode {
    description
      "Choice of how flags are indicated.";
    case explicit {
      leaf operator {
        type operator;
        default "match";
        description
          "How to interpret the TCP flags.";
      }
    }
  }
}

```

```

    leaf-list explicit-tcp-flag {
        type identityref {
            base tcp-flag;
        }
        description
            "An explicit list of the TCP flags that are to be
            matched.";
    }
}
case builtin {
    leaf bitmask {
        type uint16;
        description
            "The bitmask matches the last 4 bits of byte 12 and 13 of
            the TCP header. For clarity, the 4 bits of byte 12
            corresponding to the TCP data offset field are not
            included in any matching.";
        reference
            "RFC 9293: Transmission Control Protocol (TCP),
            Section 3.1";
    }
}
}
}

grouping fragment-fields {
    description
        "Operations on fragment types.";
    leaf operator {
        type operator;
        default "match";
        description
            "How to interpret the fragment type.";
    }
    leaf type {
        type fragment-type;
        description
            "What fragment type to look for.";
    }
}

grouping mpls-match-parameters-config {
    description
        "Parameters for the configuration of MPLS match rules.";

    leaf traffic-class {
        type uint8 {
            range "0..7";
        }
    }
}

```

```

description
  "The value of the MPLS traffic class (TC) bits,
  formerly known as the EXP bits.";
}

leaf label-position {
  type identityref {
    base label-position;
  }
  description
    "Position of the label";
}

leaf upper-label-range {
  type rt-types:mpls-label;
  description
    "Match MPLS label value on the MPLS header.
    The usage of this field indicated the upper
    range value in the top of the stack.
    This label value does not include the
    encodings of Traffic Class and TTL.";
  reference
    "RFC 3032: MPLS Label Stack Encoding";
}

leaf lower-label-range {
  type rt-types:mpls-label;
  description
    "Match MPLS label value on the MPLS header.
    The usage of this field indicated the lower
    range value in the top of the stack.
    This label value does not include the
    encodings of Traffic Class and TTL.";
  reference
    "RFC 3032: MPLS Label Stack Encoding";
}

leaf label-block-name {
  type string;
  description
    "Reference to a label block predefiend in the
    implementation.";
}

leaf ttl-value {
  type uint8;
  description
    "Time-to-live MPLS packet value match.";
  reference

```

```

    "RFC 3032: MPLS Label Stack Encoding";
}
}

grouping payload {
  description
    "Operations on payload match.";
  leaf offset {
    type identityref {
      base offset-type;
    }
    description
      "Indicates the payload offset. This will indicate
      the position of the data in packet to use for
      the match.";
  }
  leaf offset-end {
    type uint64;
    units "bytes";
    description
      "Indicates the number of bytes, starting from the offset to
      cover when performing the prefix match.";
  }
  leaf operator {
    type operator;
    default "match";
    description
      "How to interpret the prefix match.";
  }
  leaf prefix {
    type binary;
    description
      "The binary pattern to match against.";
  }
}

grouping alias {
  description
    "Specifies an alias.";
  leaf-list vlan {
    type uint16;
    description
      "VLAN of the alias.";
  }
  leaf-list prefix {
    type inet:ip-prefix;
    description
      "IPv4 or IPv6 prefix of the alias.";
  }
}

```

```

list port-range {
  key "lower-port";
  description
    "Port range.  When only lower-port is
    present, it represents a single port number.";
  leaf lower-port {
    type inet:port-number;
    mandatory true;
    description
      "Lower port number of the port range.";
  }
  leaf upper-port {
    type inet:port-number;
    must '. >= ../lower-port' {
      error-message
        "The upper-port number must be greater than
        or equal to the lower-port number.";
    }
    description
      "Upper port number of the port range.";
  }
}
leaf-list protocol {
  type uint8;
  description
    "Identifies the target protocol number.
    For example, 6 for TCP or 17 for UDP.";
}
leaf-list fqdn {
  type inet:domain-name;
  description
    "FQDN identifying the target.";
}
leaf-list uri {
  type inet:uri;
  description
    "URI identifying the target.";
}
}

grouping icmpv4-header-fields {
  description
    "Collection of ICMPv4 header fields that can be
    used to set up a match filter.";
  leaf type {
    type iana-icmpv4-types:icmpv4-type;
    description
      "Also known as control messages.";
    reference

```



```

        "RFC 792: Internet Control Message Protocol.";
    }
    leaf code {
        type uint8;
        description
            "ICMP subtype.";
        reference
            "RFC 792: Internet Control Message Protocol.";
    }
    leaf rest-of-header {
        type binary;
        description
            "Unbounded in length, the contents vary based on the
            ICMP type and code.";
        reference
            "RFC 792: Internet Control Message Protocol";
    }
}

grouping icmpv6-header-fields {
    description
        "Collection of ICMPv6 header fields that can be
        used to set up a match filter.";
    leaf type {
        type iana-icmpv6-types:icmpv6-type;
        description
            "Also known as control messages.";
        reference
            "RFC 4443: Internet Control Message Protocol (ICMPv6)
            for Internet Protocol Version 6 (IPv6)
            Specification.";
    }
    leaf code {
        type uint8;
        description
            "ICMP code.";
        reference
            "RFC 4443: Internet Control Message Protocol (ICMPv6)
            for Internet Protocol Version 6 (IPv6)
            Specification.";
    }
    leaf rest-of-header {
        type binary;
        description
            "Unbounded in length, the contents vary based on the
            ICMP type and code. Also referred to as 'Message Body'
            in ICMPv6.";
        reference
            "RFC 4443: Internet Control Message Protocol (ICMPv6)

```

```

        for Internet Protocol Version 6 (IPv6)
        Specification.";
    }
}

grouping acl-complementary-actions {
    description
        "Collection of complementary ACL actions.";

    container log-action {
        description
            "Container for defining log actions.";

        leaf log-type {
            type identityref {
                base acl-enh:log-types;
            }
            description
                "The type of log action to be performed.";
        }
        leaf log-id {
            when "derived-from-or-self(..../log-type, "
                + "'acl-enh:local-log')" {
                description
                    "Name of the log file updated when type is 'local-log'.";
            }
            type string;
            description
                "The name of the counter action.";
        }
    }
}

container counter-action {
    description
        "Container for defining counter actions.";

    leaf counter-type {
        type identityref {
            base acl-enh:counter-type;
        }
        description
            "The type of counter action to be performed.";
    }
    leaf-list counter-name {
        when "derived-from-or-self(..../counter-type, "
            + "'acl-enh:counter-name')" {
            description
                "Name for the counter or variable to update when

```

```

        'counter-type' is 'counter-name'.";
    }
    type string;
    description
        "List of possible variables or counter names to
        update based on match criteria.";
    }
}
}

augment "/acl:acls/acl:acl/acl:aces/acl:ace"
    + "/acl:matches" {
    description
        "Adds a match type based on the payload.";
    choice payload {
        description
            "Matches based upon a prefix pattern.";
        container prefix-pattern {
            if-feature "match-on-payload";
            description
                "Indicates the rule to perform the payload-based match.";
            uses payload;
        }
    }
    choice alias {
        description
            "Matches based upon aliases.";
        leaf-list alias-name {
            type alias-ref;
            description
                "Indicates one or more aliases.";
        }
    }
    choice mpls {
        container mpls-values {
            if-feature "match-on-mpls";
            uses mpls-match-parameters-config;
            description
                "Provides the rule set that matches MPLS headers.";
        }
        description
            "Matches against MPLS headers, for example, label
            values";
    }
}

augment "/acl:acls/acl:acl/acl:aces"
    + "/acl:ace/acl:matches/acl:l2" {
    description

```

```

"Adds a match type based on MAC VLAN and ISID filters.";
container vlan-filter {
  if-feature "match-on-vlan-filter";
  description
    "Indicates how to handle MAC VLANs.";
  leaf frame-type {
    type string;
    description
      "Entering the frame type allows the
        filter to match a specific type of frame format";
  }
  choice vlan-type {
    description
      "VLAN definition from range or operator.";
    case range {
      leaf lower-vlan {
        type uint16;
        must '. <= ../upper-vlan' {
          error-message
            "The lower-vlan must be less than or equal to
              the upper-vlan.";
        }
        mandatory true;
        description
          "Lower boundary for a vlan.";
      }
      leaf upper-vlan {
        type uint16;
        mandatory true;
        description
          "Upper boundary for a vlan.";
      }
    }
  }
  case operator {
    leaf operator {
      type packet-fields:operator;
      default "eq";
      description
        "Operator to be applied on the vlan below.";
    }
    leaf-list vlan {
      type uint16;
      description
        "VLAN number along with the operator on which to
          match.";
    }
  }
}
}
}
}

```

```

container isid-filter {
  if-feature "match-on-isid-filter";
  description
    "Indicates how to handle ISID filters.
    The I-component is responsible for mapping customer
    Ethernet traffic to the appropriate ISID.";
  choice isid-type {
    description
      "ISID definition from range or operator.";
    case range {
      leaf lower-isid {
        type uint16;
        must '. <= ../upper-isid' {
          error-message
            "The lower-isid must be less than or equal to
            the upper-isid.";
        }
        mandatory true;
        description
          "Lower boundary for a ISID.";
      }
      leaf upper-isid {
        type uint16;
        mandatory true;
        description
          "Upper boundary for a ISID.";
      }
    }
    case operator {
      leaf operator {
        type packet-fields:operator;
        default "eq";
        description
          "Operator to be applied on the ISID below.";
      }
      leaf-list isid {
        type uint16;
        description
          "ISID number along with the operator on which to
          match.";
      }
    }
  }
}

augment "/acl:acls/acl:acl/acl:aces"
  + "/acl:ace/acl:matches/acl:l3/acl:ipv4/acl:ipv4" {
  description

```

```

    "Handle non-initial and initial fragments for IPv4 packets.";
container ipv4-fragment {
    must "not(..acl:flags)" {
        error-message
            "Either flags or fragment should be provided, but not
            both.";
    }
    description
        "Indicates how to handle IPv4 fragments.";
    uses fragment-fields;
}
leaf source-ipv4-prefix-list {
    type ipv4-prefix-set-ref;
    description
        "A reference to an IPv4 prefix list to match the source
        address.";
}
leaf destination-ipv4-prefix-list {
    type ipv4-prefix-set-ref;
    description
        "A reference to a prefix list to match the destination
        address.";
}
leaf next-header-set {
    type protocol-set-ref;
    description
        "A reference to a protocol set to match the next-header
        field.";
}
}

augment "/acl:acls/acl:acl/acl:aces"
    + "/acl:ace/acl:matches/acl:l3/acl:ipv6/acl:ipv6" {
    description
        "Handles non-initial and initial fragments for IPv6 packets.";
    container ipv6-fragment {
        description
            "Indicates how to handle IPv6 fragments.";
        uses fragment-fields;
    }
    leaf source-ipv6-prefix-list {
        type ipv6-prefix-set-ref;
        description
            "A reference to a prefix list to match the source address.";
    }
    leaf destination-ipv6-prefix-list {
        type ipv6-prefix-set-ref;
        description
            "A reference to a prefix list to match the destination

```

```

        address.";
    }
    leaf protocol-set {
        type protocol-set-ref;
        description
            "A reference to a protocol set to match the protocol field.";
    }
    leaf extension-header {
        type iana-ipv6-ext-types:ipv6-extension-header-type;
        description
            "IPv6 extension header value.";
    }
}

augment "/acl:acls/acl:acl/acl:aces"
    + "/acl:ace/acl:matches/acl:l4/acl:tcp/acl:tcp" {
    description
        "Handles TCP flags and port sets.";
    container flags-bitmask {
        must "not(..acl:flags)" {
            error-message
                "Either flags or flags-bitmask should be provided, but not
                both.";
        }
        description
            "Indicates how to handle TCP flags.";
        uses tcp-flags;
    }
    leaf source-tcp-port-set {
        type port-set-ref;
        description
            "A reference to a port set to match the source port.";
    }
    leaf destination-tcp-port-set {
        type port-set-ref;
        description
            "A reference to a port set to match the destination port.";
    }
}

augment "/acl:acls/acl:acl/acl:aces"
    + "/acl:ace/acl:matches/acl:l4/acl:udp/acl:udp" {
    description
        "Handle UDP port sets.";
    leaf source-udp-port-set {
        type port-set-ref;
        description
            "A reference to a port set to match the source port.";
    }
}

```

```
leaf destination-udp-port-set {
  type port-set-ref;
  description
    "A reference to a port set to match the destination port.";
}
}
```

```
augment "/acl:acls/acl:acl/acl:aces"
  + "/acl:ace/acl:matches/acl:l4/acl:icmp/acl:icmp" {
  description
    "Handle ICMP type sets.";
  leaf icmpv4-set {
    type icmpv4-type-set-ref;
    description
      "A reference to an ICMPv4 type set to match the ICMPv4 type
      field.";
  }
  leaf icmpv6-set {
    type icmpv6-type-set-ref;
    description
      "A reference to an ICMPv6 type set to match the ICMPv6 type
      field.";
  }
}
```

```
augment "/acl:acls/acl:acl/acl:aces"
  + "/acl:ace/acl:actions" {
  description
    "Complementary actions including Rate-limit action.";

  uses acl-complementary-actions;

  leaf rate-limit {
    when "../acl:forwarding = 'acl:accept'" {
      description
        "Rate-limit valid only when accept action is used.";
    }
    type decimal64 {
      fraction-digits 2;
    }
    units "bytes per second";
    description
      "Indicates a rate-limit for the matched traffic.";
  }
}
```

```
container defined-sets {
  description
```



```

    "Predefined sets of attributes used in policy match
    statements.";
container ipv4-prefix-sets {
  description
    "Data definitions for a list of IPv4 or IPv6
    prefixes which are matched as part of a policy.";
  list prefix-set {
    key "name";
    description
      "List of the defined prefix sets.";
    leaf name {
      type string;
      description
        "Name of the prefix set -- this is used as a label to
        reference the set in match conditions.";
    }
    leaf description {
      type string;
      description
        "Defined Set description.";
    }
    leaf-list prefix {
      type inet:ipv4-prefix;
      description
        "List of IPv4 prefixes to be used in match
        conditions.";
    }
  }
}
container ipv6-prefix-sets {
  description
    "Data definitions for a list of IPv6 prefixes which are
    matched as part of a policy.";
  list prefix-set {
    key "name";
    description
      "List of the defined prefix sets.";
    leaf name {
      type string;
      description
        "Name of the prefix set -- this is used as a label to
        reference the set in match conditions.";
    }
    leaf description {
      type string;
      description
        "A textual description of the prefix list.";
    }
    leaf-list prefix {

```

```

        type inet:ipv6-prefix;
        description
            "List of IPv6 prefixes to be used in match conditions.";
    }
}
}
container port-sets {
    description
        "Data definitions for a list of ports which can
        be matched in policies.";
    list port-set {
        key "name";
        description
            "List of port set definitions.";
        leaf name {
            type string;
            description
                "Name of the port set -- this is used as a label to
                reference the set in match conditions.";
        }
        list port {
            key "id";
            description
                "Port numbers along with the operator on which to
                match.";
            leaf id {
                type string;
                description
                    "Identifier of the list of port numbers.";
            }
        }
        choice port {
            description
                "Choice of specifying the port number or referring to a
                group of port numbers.";
            container port-range-or-operator {
                description
                    "Indicates a set of ports.";
                uses packet-fields:port-range-or-operator;
            }
        }
    }
}
}
}
container protocol-sets {
    description
        "Data definitions for a list of protocols which can be
        matched in policies.";
    list protocol-set {
        key "name";

```

```

description
  "List of protocol set definitions.";
leaf name {
  type string;
  description
    "Name of the protocols set -- this is used as a
    label to reference the set in match conditions.";
}
leaf-list protocol {
  type union {
    type uint8;
    type string;
  }
  description
    "Value of the protocol set.";
  //Check if we can reuse an IANA-maintained module
}
}
}
container icmpv4-type-sets {
  description
    "Data definitions for a list of ICMPv4 types which can be
    matched in policies.";
  list icmpv4-type-set {
    key "name";
    description
      "List of ICMP type set definitions.";
    leaf name {
      type string;
      description
        "Name of the ICMPv4 type set -- this is used as a label
        to reference the set in match conditions.";
    }
    list types {
      key "type";
      description
        "Includes a list of ICMPv4 types.";
      uses icmpv4-header-fields;
    }
  }
}
}
container icmpv6-type-sets {
  description
    "Data definitions for a list of ICMPv6 types which can be
    matched in policies.";
  list icmpv6-type-set {
    key "name";
    description
      "List of ICMP type set definitions.";

```



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. These are the subtrees and data nodes and their sensitivity/vulnerability:

**'defined-sets' and 'aliases':** These lists specify a set of sets and aliases. Similar to [[RFC8519](#)], unauthorized write access to these list can allow intruders to modify the entries so as to permit traffic that should not be permitted, or deny traffic that should be permitted. The former may result in a DoS attack, or compromise a device. The latter may result in a DoS attack.

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:

**'defined-sets' and 'aliases':** Unauthorized read access of these lists will allow an attacker to identify the actual resources that are bound to ACLs.

## 6. IANA Considerations

### 6.1. URI Registrations

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:ietf-acl-enh  
Registrant Contact: The IESG.  
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:iana-icmpv4-types  
Registrant Contact: The IESG.  
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:iana-icmpv6-types  
Registrant Contact: The IESG.  
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:iana-ipv6-ext-types  
Registrant Contact: The IESG.  
XML: N/A; the requested URI is an XML namespace.

## 6.2. YANG Module Name Registrations

This document requests IANA to register the following YANG modules in the "YANG Module Names" subregistry [[RFC6020](#)] within the "YANG Parameters" registry.

```
name: ietf-acl-enh
namespace: urn:ietf:params:xml:ns:yang:ietf-acl-enh
maintained by IANA: N
prefix: acl-enh
reference: RFC XXXX
```

```
name: iana-icmpv4-types
namespace: urn:ietf:params:xml:ns:yang:iana-icmpv4-types
maintained by IANA: Y
prefix: iana-icmpv4-types
reference: RFC XXXX
```

```
name: iana-icmpv6-types
namespace: urn:ietf:params:xml:ns:yang:iana-icmpv6-types
maintained by IANA: Y
prefix: iana-icmpv6-types
reference: RFC XXXX
```

```
name: iana-ipv6-ext-types
namespace: urn:ietf:params:xml:ns:yang:iana-ipv6-ext-types
maintained by IANA: Y
prefix: iana-ipv6-ext-types
reference: RFC XXXX
```

## 6.3. Considerations for IANA-Maintained Modules

### 6.3.1. ICMPv4 Types IANA Module

IANA is requested to create and post the initial version of the "iana-icmpv4-types" YANG module by applying the XSLT stylesheet from [Appendix A.1](#) to the XML version of [[IANA-ICMPv4](#)].

This document defines the initial version of the IANA-maintained "iana-icmpv4-types" YANG module. The most recent version of the YANG module is available from the "YANG Parameters" registry [[IANA-YANG-PARAMETERS](#)].

IANA is requested to add this note to the registry [[IANA-YANG-PARAMETERS](#)]:

```
New values must not be directly added to the "iana-icmpv4-types"
YANG module. They must instead be added to the "ICMP Type Numbers"
registry [IANA-ICMPv4].
```

When a value is added to the "ICMP Type Numbers" registry, a new "enum" statement must be added to the "iana-icmpv4-types" YANG module. The "enum" statement, and sub-statements thereof, should be defined:

**"enum"**: Replicates a name from the registry.

**"value"**: Contains the decimal value of the IANA-assigned value.

**"status"**: Is included only if a registration has been deprecated or obsoleted. IANA "deprecated" maps to YANG status "deprecated", and IANA "obsolete" maps to YANG status "obsolete".

**"description"**: Replicates the description from the registry.

**"reference"**: Replicates the reference(s) from the registry with the title of the document(s) added.

Unassigned or reserved values are not present in the module.

When the "iana-icmpv4-types" YANG module is updated, a new "revision" statement with a unique revision date must be added in front of the existing revision statements.

IANA is requested to add this note to "ICMP Type Numbers" [[IANA-ICMPv4](#)]:

When this registry is modified, the YANG module "iana-icmpv4-types" [IANA\_ICMPv4\_YANG\_URL] must be updated as defined in RFC XXXX.

IANA is requested to updated the "Reference" in the "ICMP Type Numbers" registry as follows:

**OLD:** [[RFC2780](#)]

**NEW:** [[RFC2780](#)][RFCXXXX]

### 6.3.2. ICMPv6 Types IANA Module

IANA is requested to create and post the initial version of the "iana-icmpv6-types" YANG module by applying the XSLT stylesheet from [Appendix B.1](#) to the XML version of [[IANA-ICMPv4](#)].

This document defines the initial version of the IANA-maintained "iana-icmpv6-types" YANG module. The most recent version of the YANG module is available from the "YANG Parameters" registry [[IANA-YANG-PARAMETERS](#)].

IANA is requested to add this note to the registry [[IANA-YANG-PARAMETERS](#)]:

New values must not be directly added to the "iana-icmpv6-types" YANG module. They must instead be added to the "ICMPv6 "type" Numbers" registry [[IANA-ICMPv6](#)].

When a value is added to the "ICMPv6 "type" Numbers" registry, a new "enum" statement must be added to the "iana-icmpv6-types" YANG module. The "enum" statement, and sub-statements thereof, should be defined:

**"enum"**: Replicates a name from the registry.

**"value"**: Contains the decimal value of the IANA-assigned value.

**"status"**: Is included only if a registration has been deprecated or obsoleted. IANA "deprecated" maps to YANG status "deprecated", and IANA "obsolete" maps to YANG status "obsolete".

**"description"**: Replicates the description from the registry.

**"reference"**: Replicates the reference(s) from the registry with the title of the document(s) added.

Unassigned or reserved values are not present in the module.

When the "iana-icmpv6-types" YANG module is updated, a new "revision" statement with a unique revision date must be added in front of the existing revision statements.

IANA is requested to add this note to "ICMPv6 "type" Numbers" [[IANA-ICMPv6](#)]:

When this registry is modified, the YANG module "iana-icmpv6-types" [[IANA\\_ICMPv6\\_YANG\\_URL](#)] must be updated as defined in RFC XXXX.

IANA is requested to updated the "Reference" in the "ICMPv6 "type" Numbers" registry as follows:

**OLD:** [[RFC4443](#)]

**NEW:** [[RFC4443](#)][RFCXXXX]

### 6.3.3. IPv6 Extension Header Types IANA Module

IANA is requested to create and post the initial version of the "iana-ipv6-ext-types" YANG module by applying the XSLT stylesheet from [Appendix C.1](#) to the XML version of [[IANA-IPv6](#)].



This document defines the initial version of the IANA-maintained "iana-ipv6-ext-types" YANG module. The most recent version of the YANG module is available from the "YANG Parameters" registry [[IANA-YANG-PARAMETERS](#)].

IANA is requested to add this note to the registry [[IANA-YANG-PARAMETERS](#)]:

New values must not be directly added to the "iana-ipv6-ext-types" YANG module. They must instead be added to the "IPv6 Extension Header Types" registry [[IANA-ICMPV6](#)].

When a value is added to the "IPv6 Extension Header Types" registry, a new "enum" statement must be added to the "iana-ipv6-ext-types" YANG module. The "enum" statement, and sub-statements thereof, should be defined:

**"enum"**: Replicates a name from the registry.

**"value"**: Contains the decimal value of the IANA-assigned value.

**"status"**: Is included only if a registration has been deprecated or obsoleted. IANA "deprecated" maps to YANG status "deprecated", and IANA "obsolete" maps to YANG status "obsolete".

**"description"**: Replicates the description from the registry.

**"reference"**: Replicates the reference(s) from the registry with the title of the document(s) added.

Unassigned or reserved values are not present in the module.

When the "iana-ipv6-ext-types" YANG module is updated, a new "revision" statement with a unique revision date must be added in front of the existing revision statements.

IANA is requested to add this note to the "IPv6 Extension Header Types" registry [[IANA-IPv6](#)]:

When this registry is modified, the YANG module "iana-ipv6-ext-types" [[IANA\\_IPV6\\_YANG\\_URL](#)] must be updated as defined in RFC XXXX.

IANA is requested to updated the "Reference" in the "IPv6 Extension Header Types" registry as follows:

**OLD:** [[RFC2780](#)][[RFC5237](#)][[RFC7045](#)]

**NEW:** [[RFC2780](#)][[RFC5237](#)][[RFC7045](#)][[RFCXXXX](#)]

## 7. References

### 7.1. Normative References

- [RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <<https://www.rfc-editor.org/rfc/rfc792>>.
- [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>>.
- [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/rfc/rfc3032>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/rfc/rfc3688>>.
- [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/rfc/rfc4443>>.
- [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/rfc/rfc5462>>.
- [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/rfc/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/rfc/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/rfc/rfc6242>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/rfc/rfc6991>>.

- [RFC7209] Sajassi, A., Aggarwal, R., Uttaro, J., Bitar, N., Henderickx, W., and A. Isaac, "Requirements for Ethernet VPN (EVPN)", RFC 7209, DOI 10.17487/RFC7209, May 2014, <<https://www.rfc-editor.org/rfc/rfc7209>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/rfc/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/rfc/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/rfc/rfc8174>>.
- [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/rfc/rfc8294>>.
- [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/rfc/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/rfc/rfc8342>>.
- [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/rfc/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/rfc/rfc8519>>.
- [RFC8956] Loibl, C., Ed., Raszuk, R., Ed., and S. Hares, Ed., "Dissemination of Flow Specification Rules for IPv6", RFC 8956, DOI 10.17487/RFC8956, December 2020, <<https://www.rfc-editor.org/rfc/rfc8956>>.
- [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>>.

## 7.2. Informative References

- [I-D.ietf-netmod-rfc8407bis] Boucadair, M. and Q. Wu, "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-netmod-rfc8407bis-06, 18 December 2023, <<https://datatracker.ietf.org/doc/html/draft-ietf-netmod-rfc8407bis-06>>.
- [IANA-ICMPv4] "ICMP Type Numbers", n.d., <<https://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml>>.
- [IANA-ICMPv6] "ICMPv6 type Numbers", n.d., <<https://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xhtml>>.
- [IANA-IPv6] "IPv6 Extension Header Types", n.d., <<https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml>>.
- [IANA-YANG-PARAMETERS] "YANG Parameters", n.d., <<https://www.iana.org/assignments/yang-parameters>>.
- [IEEE-802-1ah] IEEE, "IEEE Standard for Local and metropolitan area networks -- Virtual Bridged Local Area Networks Amendment 7: Provider Backbone Bridges", August 2008, <[https://standards.ieee.org/standard/802\\_1ah-2008.html](https://standards.ieee.org/standard/802_1ah-2008.html)>.
- [RFC2780] Bradner, S. and V. Paxson, "IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers", BCP 37, RFC 2780, DOI 10.17487/RFC2780, March 2000, <<https://www.rfc-editor.org/rfc/rfc2780>>.
- [RFC5237] Arkko, J. and S. Bradner, "IANA Allocation Guidelines for the Protocol Field", BCP 37, RFC 5237, DOI 10.17487/RFC5237, February 2008, <<https://www.rfc-editor.org/rfc/rfc5237>>.
- [RFC7045] Carpenter, B. and S. Jiang, "Transmission and Processing of IPv6 Extension Headers", RFC 7045, DOI 10.17487/RFC7045, December 2013, <<https://www.rfc-editor.org/rfc/rfc7045>>.
- [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/rfc/rfc8340>>.
- [RFC8955] Loibl, C., Hares, S., Raszuk, R., McPherson, D., and M. Bacher, "Dissemination of Flow Specification Rules", RFC 8955, DOI 10.17487/RFC8955, December 2020, <<https://www.rfc-editor.org/rfc/rfc8955>>.

**[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>>.

**[RFC9132]**

Boucadair, M., Ed., Shallow, J., and T. Reddy.K, "Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification", RFC 9132, DOI 10.17487/RFC9132, September 2021, <<https://www.rfc-editor.org/rfc/rfc9132>>.

## Appendix A. ICMPv4 Types

## **A.1. XSLT Template to Generate The ICMPv4 Types IANA-Maintained Module**

<CODE BEGINS>

```
<?xml version="1.0" encoding="utf-8"?>
<stylesheet
  xmlns="http://www.w3.org/1999/XSL/Transform"
  xmlns:html="http://www.w3.org/1999/xhtml"
  xmlns:iana="http://www.iana.org/assignments"
  xmlns:yin="urn:ietf:params:xml:ns:yang:yin:1"
  version="1.0">
<import href="../../../xslt/iana-yinx.xsl"/>
<output method="xml" encoding="utf-8"/>
<strip-space elements="*" />

<template match="iana:registry[@id='icmp-parameters-types']">
  <element name="yin:typedef">
    <attribute name="name">icmpv4-type-name</attribute>
    <element name="yin:type">
      <attribute name="name">enumeration</attribute>
      <apply-templates
        select="iana:record[not(iana:description = 'Unassigned' or
          starts-with(iana:description, 'Reserved') or
          starts-with(iana:description, 'RFC3692')) or
          contains(iana:description, 'experimental')]"/>
    </element>
    <element name="yin:description">
      <element name="yin:text">
        This enumeration type defines mnemonic names and
        corresponding numeric values of ICMPv4 types.
      </element>
    </element>
    <element name="yin:reference">
      <element name="yin:text">
        RFC 2708: IANA Allocation Guidelines For Values In
        the Internet Protocol and Related Headers
      </element>
    </element>
  </element>
</template>

<template match="iana:record[@id='icmp-parameters-types']">
  <element name="yin:typedef">
    <attribute name="name">icmpv4-type</attribute>
    <element name="yin:type">
      <attribute name="name">union</attribute>
      <element name="yin:type">
        <attribute name="name">uint8</attribute>
      </element>
      <element name="yin:type">
        <attribute name="name">icmpv4-type-name</attribute>
      </element>
    </element>
  </element>
  <element name="yin:description">
```



```
<element name="yin:text">
  This type allows reference to an ICMPv4 type using either
  the assigned mnemonic name or numeric value.
</element>
</element>
</element>
</template>

<template match="iana:record">
  <call-template name="enum">
    <with-param name="id">
      <choose>
        <when test="contains(iana:description, '(Deprecated'))">
          <value-of select="translate(normalize-space(substring-before
            '(Deprecated)'), ' ', '')"/>
        </when>
        <otherwise>
          <value-of select="translate(normalize-space(iana:description
            </otherwise>
        </choose>
      </with-param>
      <with-param name="deprecated"
        select="contains(iana:description, '(Deprecated'))"/>
    </call-template>
  </template>

</stylesheet>

<CODE ENDS>
```

## A.2. Initial Version of the The ICMPv4 Types IANA-Maintained Module

<CODE BEGINS> file "iana-icmpv4-types@2020-09-25.yang"

```
module iana-icmpv4-types {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-icmpv4-types";
  prefix iana-icmpv4-types;

  organization
    "Internet Assigned Numbers Authority (IANA)";

  contact
    "Internet Assigned Numbers Authority

    ICANN
    12025 Waterfront Drive, Suite 300
    Los Angeles, CA 90094

    Tel: +1 424 254 5300

    <mailto:iana@iana.org>";

  description
    "This YANG module translates IANA registry 'ICMP Type Numbers' to
    YANG derived types.

    Copyright (c) 2020 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).

    The initial version of this YANG module is part of RFC XXXX;
    see the RFC itself for full legal notices.

    This version of this YANG module was generated from the
    corresponding IANA registry using an XSLT stylesheet from the
    'iana-yang' project (https://github.com/llhotka/iana-yang).";

  reference
    "Internet Control Message Protocol (ICMP) Parameters
    (https://www.iana.org/assignments/icmp-parameters/)";

  revision 2020-09-25 {
    description
```

```
    "Current revision as of the revision date specified in the XML
    representation of the registry page.";
reference
    "https://www.iana.org/assignments/icmp-parameters/icmp-parameters.
}
```

```
/* Typedefs */
```

```
typedef icmpv4-type-name {
    type enumeration {
        enum EchoReply {
            value 0;
            description
                "Echo Reply";
            reference
                "RFC 792";
        }
        enum DestinationUnreachable {
            value 3;
            description
                "Destination Unreachable";
            reference
                "RFC 792";
        }
        enum SourceQuench {
            value 4;
            status deprecated;
            description
                "Source Quench (Deprecated)";
            reference
                "- RFC 792
                - RFC 6633";
        }
        enum Redirect {
            value 5;
            description
                "Redirect";
            reference
                "RFC 792";
        }
        enum AlternateHostAddress {
            value 6;
            status deprecated;
            description
                "Alternate Host Address (Deprecated)";
            reference
                "RFC 6918";
        }
        enum Echo {
```

```
    value 8;
    description
        "Echo";
    reference
        "RFC 792";
}
enum RouterAdvertisement {
    value 9;
    description
        "Router Advertisement";
    reference
        "RFC 1256";
}
enum RouterSolicitation {
    value 10;
    description
        "Router Solicitation";
    reference
        "RFC 1256";
}
enum TimeExceeded {
    value 11;
    description
        "Time Exceeded";
    reference
        "RFC 792";
}
enum ParameterProblem {
    value 12;
    description
        "Parameter Problem";
    reference
        "RFC 792";
}
enum Timestamp {
    value 13;
    description
        "Timestamp";
    reference
        "RFC 792";
}
enum TimestampReply {
    value 14;
    description
        "Timestamp Reply";
    reference
        "RFC 792";
}
enum InformationRequest {
```

```
value 15;
status deprecated;
description
    "Information Request (Deprecated)";
reference
    "- RFC 792
    - RFC 6918";
}
enum InformationReply {
value 16;
status deprecated;
description
    "Information Reply (Deprecated)";
reference
    "- RFC 792
    - RFC 6918";
}
enum AddressMaskRequest {
value 17;
status deprecated;
description
    "Address Mask Request (Deprecated)";
reference
    "- RFC 950
    - RFC 6918";
}
enum AddressMaskReply {
value 18;
status deprecated;
description
    "Address Mask Reply (Deprecated)";
reference
    "- RFC 950
    - RFC 6918";
}
enum Traceroute {
value 30;
status deprecated;
description
    "Traceroute (Deprecated)";
reference
    "- RFC 1393
    - RFC 6918";
}
enum DatagramConversionError {
value 31;
status deprecated;
description
    "Datagram Conversion Error (Deprecated)";
```

```
reference
  "- RFC 1475
    - RFC 6918";
}
enum MobileHostRedirect {
  value 32;
  status deprecated;
  description
    "Mobile Host Redirect (Deprecated)";
  reference
    "- David Johnson <>
      - RFC 6918";
}
enum IPv6Where-Are-You {
  value 33;
  status deprecated;
  description
    "IPv6 Where-Are-You (Deprecated)";
  reference
    "- Bill Simpson <mailto:Bill.Simpson@um.cc.umich.edu>
      - RFC 6918";
}
enum IPv6I-Am-Here {
  value 34;
  status deprecated;
  description
    "IPv6 I-Am-Here (Deprecated)";
  reference
    "- Bill Simpson <mailto:Bill.Simpson@um.cc.umich.edu>
      - RFC 6918";
}
enum MobileRegistrationRequest {
  value 35;
  status deprecated;
  description
    "Mobile Registration Request (Deprecated)";
  reference
    "- Bill Simpson <mailto:Bill.Simpson@um.cc.umich.edu>
      - RFC 6918";
}
enum MobileRegistrationReply {
  value 36;
  status deprecated;
  description
    "Mobile Registration Reply (Deprecated)";
  reference
    "- Bill Simpson <mailto:Bill.Simpson@um.cc.umich.edu>
      - RFC 6918";
}
```

```
enum DomainNameRequest {
    value 37;
    status deprecated;
    description
        "Domain Name Request (Deprecated)";
    reference
        "- RFC 1788
        - RFC 6918";
}
enum DomainNameReply {
    value 38;
    status deprecated;
    description
        "Domain Name Reply (Deprecated)";
    reference
        "- RFC 1788
        - RFC 6918";
}
enum SKIP {
    value 39;
    status deprecated;
    description
        "SKIP (Deprecated)";
    reference
        "- Tom Markson <mailto:markson@osmosys.incog.com>
        - RFC 6918";
}
enum Photuris {
    value 40;
    description
        "Photuris";
    reference
        "RFC 2521";
}
enum ICMPmessagesutilizedbyexperimentalprotocolsuchasSeam
    value 41;
    description
        "ICMP messages utilized by experimental mobility protocols
        such as Seamoby";
    reference
        "RFC 4065";
}
enum ExtendedEchoRequest {
    value 42;
    description
        "Extended Echo Request";
    reference
        "RFC 8335";
}
```



```
enum ExtendedEchoReply {
    value 43;
    description
        "Extended Echo Reply";
    reference
        "RFC 8335";
}
}
description
    "This enumeration type defines mnemonic names and corresponding
    numeric values of ICMPv4 types.";
reference
    "RFC 2708: IANA Allocation Guidelines For Values In the
    Internet Protocol and Related Headers";
}

typedef icmpv4-type {
    type union {
        type uint8;
        type icmpv4-type-name;
    }
    description
        "This type allows reference to an ICMPv4 type using either the
        assigned mnemonic name or numeric value.";
}
}
```

<CODE ENDS>

## Appendix B. ICMPv6 Types

## **B.1. XSLT Template to Generate The ICMPv6 Types IANA-Maintained Module**

<CODE BEGINS>

```
<?xml version="1.0" encoding="utf-8"?>
<stylesheet
  xmlns="http://www.w3.org/1999/XSL/Transform"
  xmlns:html="http://www.w3.org/1999/xhtml"
  xmlns:iana="http://www.iana.org/assignments"
  xmlns:yin="urn:ietf:params:xml:ns:yang:yin:1"
  version="1.0">
<import href="../../../xslt/iana-yinx.xsl"/>
<output method="xml" encoding="utf-8"/>
<strip-space elements="*" />

<template match="iana:registry[@id='icmpv6-parameters-2']">
  <element name="yin:typedef">
    <attribute name="name">icmpv6-type-name</attribute>
    <element name="yin:type">
      <attribute name="name">enumeration</attribute>
      <apply-templates
        select="iana:record[not(iana:name = 'Unassigned' or
          starts-with(iana:name, 'Reserved') or
          starts-with(iana:name, 'Private'))]" />
    </element>
    <element name="yin:description">
      <element name="yin:text">
        This enumeration type defines mnemonic names and
        corresponding numeric values of ICMPV6 types.
      </element>
    </element>
    <element name="yin:reference">
      <element name="yin:text">
        RFC 2708: IANA Allocation Guidelines For Values In
        the Internet Protocol and Related Headers
      </element>
    </element>
  </element>
</template>

<template match="iana:icmpv6-type-name[@id='icmpv6-type-name']">
  <element name="yin:typedef">
    <attribute name="name">icmpv6-type</attribute>
    <element name="yin:type">
      <attribute name="name">union</attribute>
      <element name="yin:type">
        <attribute name="name">uint8</attribute>
      </element>
      <element name="yin:type">
        <attribute name="name">icmpv6-type-name</attribute>
      </element>
    </element>
  </template>

<template match="iana:icmpv6-type-name[@id='icmpv6-type-name']">
  <element name="yin:description">
    <element name="yin:text">
```

This type allows reference to an ICMPv6 type using either the assigned mnemonic name or numeric value.

```
</element>
</element>
</element>
</template>

<template match="iana:record">
  <call-template name="enum">
    <with-param name="id">
      <choose>
        <when test="contains(iana:name, '(Deprecated)')">
          <value-of select="translate(normalize-space(substring-before
            '(Deprecated)'), ' ', '')"/>
        </when>
        <otherwise>
          <value-of select="translate(normalize-space(iana:name), ' ', ' ')" />
        </otherwise>
      </choose>
    </with-param>
    <with-param name="description">
      <value-of select="concat(iana:name, '. ')" />
    </with-param>
    <with-param name="deprecated"
      select="contains(iana:name, '(Deprecated)')"/>
  </call-template>
</template>

</stylesheet>

<CODE ENDS>
```

## **B.2. Initial Version of the The ICMPv6 Types IANA-Maintained Module**

<CODE BEGINS> file "iana-icmpv6-types@2020-09-25.yang"

```
module iana-icmpv6-types {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-icmpv6-types";
  prefix iana-icmpv6-types;

  organization
    "Internet Assigned Numbers Authority (IANA)";

  contact
    "Internet Assigned Numbers Authority

    ICANN
    12025 Waterfront Drive, Suite 300
    Los Angeles, CA 90094

    Tel: +1 424 254 5300

    <mailto:iana@iana.org>";

  description
    "This YANG module translates IANA registry 'ICMPv6 \"/>

```

```
    "Current revision as of the revision date specified in the XML
    representation of the registry page.";
reference
    "https://www.iana.org/assignments/icmpv6-parameters/icmpv6-paramet
}
```

```
/* Typedefs */
```

```
typedef icmpv6-type-name {
    type enumeration {
        enum DestinationUnreachable {
            value 1;
            description
                "Destination Unreachable.";
            reference
                "RFC 4443";
        }
        enum PacketTooBig {
            value 2;
            description
                "Packet Too Big.";
            reference
                "RFC 4443";
        }
        enum TimeExceeded {
            value 3;
            description
                "Time Exceeded.";
            reference
                "RFC 4443";
        }
        enum ParameterProblem {
            value 4;
            description
                "Parameter Problem.";
            reference
                "RFC 4443";
        }
        enum EchoRequest {
            value 128;
            description
                "Echo Request.";
            reference
                "RFC 4443";
        }
        enum EchoReply {
            value 129;
            description
                "Echo Reply.";
        }
    }
}
```



```
reference
    "RFC 4443";
}
enum MulticastListenerQuery {
    value 130;
    description
        "Multicast Listener Query.";
    reference
        "RFC 2710";
}
enum MulticastListenerReport {
    value 131;
    description
        "Multicast Listener Report.";
    reference
        "RFC 2710";
}
enum MulticastListenerDone {
    value 132;
    description
        "Multicast Listener Done.";
    reference
        "RFC 2710";
}
enum RouterSolicitation {
    value 133;
    description
        "Router Solicitation.";
    reference
        "RFC 4861";
}
enum RouterAdvertisement {
    value 134;
    description
        "Router Advertisement.";
    reference
        "RFC 4861";
}
enum NeighborSolicitation {
    value 135;
    description
        "Neighbor Solicitation.";
    reference
        "RFC 4861";
}
enum NeighborAdvertisement {
    value 136;
    description
        "Neighbor Advertisement.";
```

```
    reference
      "RFC 4861";
  }
enum RedirectMessage {
  value 137;
  description
    "Redirect Message.";
  reference
    "RFC 4861";
}
enum RouterRenumbering {
  value 138;
  description
    "Router Renumbering.";
  reference
    "RFC 2894";
}
enum ICMPNodeInformationQuery {
  value 139;
  description
    "ICMP Node Information Query.";
  reference
    "RFC 4620";
}
enum ICMPNodeInformationResponse {
  value 140;
  description
    "ICMP Node Information Response.";
  reference
    "RFC 4620";
}
enum InverseNeighborDiscoverySolicitationMessage {
  value 141;
  description
    "Inverse Neighbor Discovery Solicitation Message.";
  reference
    "RFC 3122";
}
enum InverseNeighborDiscoveryAdvertisementMessage {
  value 142;
  description
    "Inverse Neighbor Discovery Advertisement Message.";
  reference
    "RFC 3122";
}
enum Version2MulticastListenerReport {
  value 143;
  description
    "Version 2 Multicast Listener Report.";
```

```
reference
    "RFC 3810";
}
enum HomeAgentAddressDiscoveryRequestMessage {
    value 144;
    description
        "Home Agent Address Discovery Request Message.";
    reference
        "RFC 6275";
}
enum HomeAgentAddressDiscoveryReplyMessage {
    value 145;
    description
        "Home Agent Address Discovery Reply Message.";
    reference
        "RFC 6275";
}
enum MobilePrefixSolicitation {
    value 146;
    description
        "Mobile Prefix Solicitation.";
    reference
        "RFC 6275";
}
enum MobilePrefixAdvertisement {
    value 147;
    description
        "Mobile Prefix Advertisement.";
    reference
        "RFC 6275";
}
enum CertificationPathSolicitationMessage {
    value 148;
    description
        "Certification Path Solicitation Message.";
    reference
        "RFC 3971";
}
enum CertificationPathAdvertisementMessage {
    value 149;
    description
        "Certification Path Advertisement Message.";
    reference
        "RFC 3971";
}
enum ICMPmessagesutilizedbyexperimentalprotocolsuchasSeam
    value 150;
    description
        "ICMP messages utilized by experimental mobility protocols
```

```
        such as Seamoby.";
    reference
        "RFC 4065";
}
enum MulticastRouterAdvertisement {
    value 151;
    description
        "Multicast Router Advertisement.";
    reference
        "RFC 4286";
}
enum MulticastRouterSolicitation {
    value 152;
    description
        "Multicast Router Solicitation.";
    reference
        "RFC 4286";
}
enum MulticastRouterTermination {
    value 153;
    description
        "Multicast Router Termination.";
    reference
        "RFC 4286";
}
enum FMIPv6Messages {
    value 154;
    description
        "FMIPv6 Messages.";
    reference
        "RFC 5568";
}
enum RPLControlMessage {
    value 155;
    description
        "RPL Control Message.";
    reference
        "RFC 6550";
}
enum ILNIPv6LocatorUpdateMessage {
    value 156;
    description
        "ILNIPv6 Locator Update Message.";
    reference
        "RFC 6743";
}
enum DuplicateAddressRequest {
    value 157;
    description
```

```

        "Duplicate Address Request.";
    reference
        "RFC 6775";
}
enum DuplicateAddressConfirmation {
    value 158;
    description
        "Duplicate Address Confirmation.";
    reference
        "RFC 6775";
}
enum MPLControlMessage {
    value 159;
    description
        "MPL Control Message.";
    reference
        "RFC 7731";
}
enum ExtendedEchoRequest {
    value 160;
    description
        "Extended Echo Request.";
    reference
        "RFC 8335";
}
enum ExtendedEchoReply {
    value 161;
    description
        "Extended Echo Reply.";
    reference
        "RFC 8335";
}
}
description
    "This enumeration type defines mnemonic names and corresponding
    numeric values of ICMPv6 types.";
reference
    "RFC 2708: IANA Allocation Guidelines For Values In the
    Internet Protocol and Related Headers";
}

typedef icmpv6-type {
    type union {
        type uint8;
        type icmpv6-type-name;
    }
}
description
    "This type allows reference to an ICMPv6 type using either the
    assigned mnemonic name or numeric value.";

```

```
}  
}
```

<CODE ENDS>

## Appendix C. IPv6 Extension Header Types

## **C.1. XSLT Template to Generate The IPv6 Extension Header Types IANA-Maintained Module**



<CODE BEGINS>

```
<?xml version="1.0" encoding="utf-8"?>
<stylesheet
  xmlns="http://www.w3.org/1999/XSL/Transform"
  xmlns:html="http://www.w3.org/1999/xhtml"
  xmlns:iana="http://www.iana.org/assignments"
  xmlns:yin="urn:ietf:params:xml:ns:yang:yin:1"
  version="1.0">
<import href="../../../xslt/iana-yinx.xsl"/>
<output method="xml" encoding="utf-8"/>
<strip-space elements="*" />

<template match="iana:registry[@id='extension-header']">
  <element name="yin:typedef">
    <attribute name="name">ipv6-extension-header-type-name</attribute>
    <element name="yin:type">
      <attribute name="name">enumeration</attribute>
      <apply-templates
        select="iana:record[not(iana:description = 'Unassigned' or
          starts-with(iana:description, 'Reserved') or
          starts-with(iana:description, 'Use for experimentati
            contains(iana:description, 'experimental'))]"/>
    </element>
    <element name="yin:description">
      <element name="yin:text">
        This enumeration type defines mnemonic names and
        corresponding numeric values of IPv6 Extension header types.
      </element>
    </element>
    <element name="yin:reference">
      <element name="yin:text">
        RFC 2708: IANA Allocation Guidelines For Values In
        the Internet Protocol and Related Headers
      </element>
    </element>
  </element>
</template>

<element name="yin:typedef">
  <attribute name="name">ipv6-extension-header-type</attribute>
  <element name="yin:type">
    <attribute name="name">union</attribute>
    <element name="yin:type">
      <attribute name="name">uint8</attribute>
    </element>
    <element name="yin:type">
      <attribute name="name">ipv6-extension-header-type-name</attrib
    </element>
  </element>
</element>
<element name="yin:description">
```

```

    <element name="yin:text">
      This type allows reference to an IPv6 Extension header type us
      the assigned mnemonic name or the numeric protocol number valu
    </element>
  </element>
</element>
</template>

<template match="iana:record">
  <call-template name="enum">
    <with-param name="id">
      <choose>
        <when test="contains(iana:description, '(Deprecated)')">
          <value-of select="translate(normalize-space(substring-before
            '(Deprecated)'), ' ', ' ')" />
        </when>
        <otherwise>
          <value-of select="translate(normalize-space(iana:description
            </otherwise>
        </choose>
      </with-param>
      <with-param name="deprecated"
        select="contains(iana:description, '(Deprecated)')"/>
    </call-template>
  </template>

</stylesheet>

<CODE ENDS>

```

## **C.2. Initial Version of The IPv6 Extension Header Types IANA-Maintained Module**

<CODE BEGINS> file "iana-ipv6-ext-types@2023-04-28.yang"

```
module iana-ipv6-ext-types {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:iana-ipv6-ext-types";
  prefix iana-ipv6-ext-types;

  organization
    "Internet Assigned Numbers Authority (IANA)";

  contact
    "Internet Assigned Numbers Authority

    ICANN
    12025 Waterfront Drive, Suite 300
    Los Angeles, CA 90094

    Tel: +1 424 254 5300

    <mailto:iana@iana.org>";

  description
    "This YANG module translates IANA registry 'IPv6 Extension Header
    Types' to YANG derived types.

    Copyright (c) 2023 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 was generated from the
    corresponding IANA registry using an XSLT stylesheet from the
    'iana-yang' project (https://github.com/llhotka/iana-yang).";

  reference
    "Internet Protocol Version 6 (IPv6) Parameters
    (https://www.iana.org/assignments/ipv6-parameters/)";

  revision 2023-09-29 {
    description
      "Current revision as of the revision date specified in the XML
      representation of the registry page.";
    reference
```

```
    "https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.  
}  
  
/* Typedefs */  
  
typedef ipv6-extension-header-type-name {  
    type enumeration {  
        enum IPv6Hop-by-HopOption {  
            value 0;  
            description  
                "IPv6 Hop-by-Hop Option";  
            reference  
                "RFC 8200";  
        }  
        enum RoutingHeaderforIPv6 {  
            value 43;  
            description  
                "Routing Header for IPv6";  
            reference  
                "- RFC 8200  
                - RFC 5095";  
        }  
        enum FragmentHeaderforIPv6 {  
            value 44;  
            description  
                "Fragment Header for IPv6";  
            reference  
                "RFC 8200";  
        }  
        enum EncapsulatingSecurityPayload {  
            value 50;  
            description  
                "Encapsulating Security Payload";  
            reference  
                "RFC 4303";  
        }  
        enum AuthenticationHeader {  
            value 51;  
            description  
                "Authentication Header";  
            reference  
                "RFC 4302";  
        }  
        enum DestinationOptionsforIPv6 {  
            value 60;  
            description  
                "Destination Options for IPv6";  
            reference  
                "RFC 8200";
```

```

    }
    enum MobilityHeader {
        value 135;
        description
            "Mobility Header";
        reference
            "RFC 6275";
    }
    enum HostIdentityProtocol {
        value 139;
        description
            "Host Identity Protocol";
        reference
            "RFC 7401";
    }
    enum Shim6Protocol {
        value 140;
        description
            "Shim6 Protocol";
        reference
            "RFC 5533";
    }
}
description
    "This enumeration type defines mnemonic names and corresponding
    numeric values of IPv6 Extension header types.";
reference
    "RFC 2708: IANA Allocation Guidelines For Values In the
    Internet Protocol and Related Headers";
}

typedef ipv6-extension-header-type {
    type union {
        type uint8;
        type ipv6-extension-header-type-name;
    }
    description
        "This type allows reference to an IPv6 Extension header type
        using either the assigned mnemonic name or the numeric
        protocol number value.";
}
}

```

<CODE ENDS>

## Appendix D. Problem Statement & Gap Analysis

### D.1. Suboptimal Configuration: Lack of Support for Lists of Prefixes

IP prefix-related data nodes, e.g., "destination-ipv4-network" or "destination-ipv6-network", do not support handling a list of IP prefixes, which may then lead to having to support large numbers of ACL entries in a configuration file.

The same issue is encountered when ACLs have to be in place to mitigate DDoS attacks that involve a set of sources (e.g., [[RFC9132](#)]). The situation is even worse when both a list of sources and destination prefixes are involved in the filtering.

[Figure 3](#) shows an example of the required ACL configuration for filtering traffic from two prefixes.

```

{
  "ietf-access-control-list:acls": {
    "acl": [
      {
        "name": "first-prefix",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [
            {
              "name": "my-test-ace",
              "matches": {
                "ipv6": {
                  "destination-ipv6-network":
                    "2001:db8:6401:1::/64",
                  "source-ipv6-network":
                    "2001:db8:1234::/96",
                  "protocol": 17,
                  "flow-label": 10000
                },
                "udp": {
                  "source-port": {
                    "operator": "lte",
                    "port": 80
                  },
                  "destination-port": {
                    "operator": "neq",
                    "port": 1010
                  }
                }
              },
              "actions": {
                "forwarding": "accept"
              }
            }
          ]
        }
      },
      {
        "name": "second-prefix",
        "type": "ipv6-acl-type",
        "aces": {
          "ace": [
            {
              "name": "my-test-ace",
              "matches": {
                "ipv6": {
                  "destination-ipv6-network":
                    "2001:db8:6401:c::/64",
                  "source-ipv6-network":

```





**Protocol sets:**

Used to create a list of protocols.

**\*Port number sets:** Used to create lists of TCP or UDP port values (or any other transport protocol that makes uses of port numbers). The identity of the protocols is identified by the protocol set, if present. Otherwise, a set applies to any protocol.

**\*ICMP sets:** Uses to create lists of ICMP-based filters. This applies only when the protocol is set to ICMP or ICMPv6.

Aliases may also be considered to manage resources that are identified by a combination of various parameters (e.g., prefix, protocol, port number, FQDN, or VLAN IDs). Note that some aliases can be provided by decomposing them into separate sets.

**D.3. Bind ACLs to Devices, Not Only Interfaces**

In the context of network management, an ACL may be enforced in many network locations. As such, the ACL module should allow for binding an ACL to multiple devices, not only (abstract) interfaces.

The ACL name must, thus, be unique at the scale of the network, but the same name may be used in many devices when enforcing node-specific ACLs.

**D.4. Partial or Lack of IPv4/IPv6 Fragment Handling**

[[RFC8519](#)] does not support fragment handling for IPv6 but offers a partial support for IPv4 through the use of 'flags'. Nevertheless, the use of 'flags' is problematic since it does not allow a bitmask to be defined. For example, setting other bits not covered by the 'flags' filtering clause in a packet will allow that packet to get through (because it won't match the ACE).

Defining a new IPv4/IPv6 matching field called 'fragment' is thus required to efficiently handle fragment-related filtering rules.

**D.5. Suboptimal TCP Flags Handling**

[[RFC8519](#)] supports including flags in the TCP match fields, however that structure does not support matching operations as those supported in BGP Flow Spec. Defining this field to be defined as a flag bitmask together with a set of operations is meant to efficiently handle TCP flags filtering rules.

## D.6. Rate-Limit Action

[RFC8519] specifies that forwarding actions can be 'accept' (i.e., accept matching traffic), 'drop' (i.e., drop matching traffic without sending any ICMP error message), or 'reject' (i.e., drop matching traffic and send an ICMP error message to the source). However, there are situations where the matching traffic can be accepted, but with a rate-limit policy. This capability is not supported by [RFC8519].

## D.7. Payload-based Filtering

Some transport protocols use existing protocols (e.g., TCP or UDP) as substrate. The match criteria for such protocols may rely upon the 'protocol' under 'l3', TCP/UDP match criteria, part of the TCP/UDP payload, or a combination thereof. [RFC8519] does not support matching based on the payload.

Likewise, the current version of the ACL model does not support filtering of encapsulated traffic.

## D.8. Reuse the ACLs Content Across Several Devices

Having a global network view of the ACLs is highly valuable for service providers. An ACL could be defined and applied based on the network topology hierarchy. So, an ACL can be defined at the network level and, then, that same ACL can be used (or referenced to) in several devices (including termination points) within the same network.

This network/device ACLs differentiation introduces several new requirements, e.g.:

- \*An ACL name can be used at both network and device levels.

- \*An ACL content updated at the network level should imply a transaction that updates the relevant content in all the nodes using this ACL.

- \*ACLs defined at the device level have a local meaning for the specific node.

- \*A device can be associated with a router, a VRF, a logical system, or a virtual node. ACLs can be applied in physical and logical infrastructure.

## D.9. Match MPLS Headers

The ACLs could be used to create rules to match MPLS fields on a packet. [RFC8519] does not support such function.

## Appendix E. Examples

This section provides a few examples to illustrate the use of the enhanced ACL module.

[Figure 4](#) shows an example of a request to install a filter to discard incoming TCP messages having all flags unset.

### E.1. TCP Flags Handling

```
{
  "ietf-access-control-list:acls":{
    "acl":[
      {
        "name":"tcp-flags-example",
        "aces":{
          "ace":[
            {
              "name":"null-attack",
              "matches":{
                "tcp":{
                  "acl-enh:flags-bitmask":{
                    "operator":"not any",
                    "bitmask":4095
                  }
                }
              },
              "actions":{
                "forwarding":"drop"
              }
            }
          ]
        }
      }
    ]
  }
}
```

Figure 4: Example of an ACL to Deny TCP Null Attack Messages (Request Body)

### E.2. Fragments Handling

[Figure 5](#) shows the content of a POST request to allow the traffic destined to 198.51.100.0/24 and UDP port number 53, but to drop all fragmented packets. The following ACEs are defined (in this order):

\*"drop-all-fragments" ACE: discards all fragments.

\*"allow-dns-packets" ACE: accepts DNS packets destined to 198.51.100.0/24.

```
{
  "ietf-access-control-list:acls":{
    "acl":[
      {
        "name":"dns-fragments",
        "type":"ipv4-acl-type",
        "aces":{
          "ace":[
            {
              "name":"drop-all-fragments",
              "matches":{
                "ipv4":{
                  "acl-enh:ipv4-fragment":{
                    "operator":"match",
                    "type":"isf"
                  }
                }
              },
              "actions":{
                "forwarding":"drop"
              }
            },
            {
              "name":"allow-dns-packets",
              "matches":{
                "ipv4":{
                  "destination-ipv4-network":"198.51.100.0/24"
                },
                "udp":{
                  "destination-port":{
                    "operator":"eq",
                    "port":53
                  }
                }
              },
              "actions":{
                "forwarding":"accept"
              }
            }
          ]
        }
      }
    ]
  }
}
```

Figure 5: Example Illustrating Candidate Filtering of IPv4 Fragmented Packets (Message Body)

[Figure 6](#) shows an example of the body of a POST request to allow the traffic destined to 2001:db8::/32 and UDP port number 53, but to drop all fragmented packets. The following ACEs are defined (in this order):

\*"drop-all-fragments" ACE: discards all fragments (including atomic fragments). That is, IPv6 packets that include a Fragment header (44) are dropped.

\*"allow-dns-packets" ACE: accepts DNS packets destined to 2001:db8::/32.

```

{
  "ietf-access-control-list:acls":{
    "acl":[
      {
        "name":"dns-fragments",
        "type":"ipv6-acl-type",
        "aces":{
          "ace":[
            {
              "name":"drop-all-fragments",
              "matches":{
                "ipv6":{
                  "acl-enh:ipv6-fragment":{
                    "operator":"match",
                    "type":"isf"
                  }
                }
              },
              "actions":{
                "forwarding":"drop"
              }
            },
            {
              "name":"allow-dns-packets",
              "matches":{
                "ipv6":{
                  "destination-ipv6-network":"2001:db8::/32"
                },
                "udp":{
                  "destination-port":{
                    "operator":"eq",
                    "port":53
                  }
                }
              },
              "actions":{
                "forwarding":"accept"
              }
            }
          ]
        }
      }
    ]
  }
}

```

Figure 6: An Example Illustrating Filtering of IPv6 Fragmented Packets (Message Body)

### E.3. VLAN Filtering

[Figure 7](#) shows an ACL example to illustrate how to apply a VLAN range filter.

```
{
  "ietf-access-control-list:acls":{
    "acl":[
      {
        "name":"VLAN_FILTER",
        "aces":{
          "ace":[
            {
              "name":"1",
              "matches":{
                "ietf-acl-enh:vlan-filter":{
                  "lower-vlan":10,
                  "upper-vlan":20
                }
              },
              "actions":{
                "forwarding":"ietf-access-control-list:accept"
              }
            }
          ]
        }
      }
    ]
  }
}
```

Figure 7: Example of VLAN Filter (Message Body)

### E.4. ISID Filtering

[Figure 8](#) shows an ACL example to illustrate the ISID range filtering.



```

{
  "ietf-acces-control-list:acls":{
    "acl":[
      {
        "name":"test",
        "aces":{
          "ace":[
            {
              "name":"1",
              "matches":{
                "ietf-acl-enh:isid-filter":{
                  "lower-isid":100,
                  "upper-isid":200
                }
              },
              "actions":{
                "forwarding":"ietf-acces-control-list:accept"
              }
            }
          ]
        }
      }
    ]
  }
}

```

Figure 8: Example ISID Filter (Message Body)

### E.5. Rate-Limit

[Figure 9](#) shows an ACL example to rate-limit incoming SYNs during a SYN flood attack.

```

{
  "ietf-access-control-list:acls":{
    "acl":[
      {
        "name":"tcp-flags-example-with-rate-limit",
        "aces":{
          "ace":[
            {
              "name":"rate-limit-syn",
              "matches":{
                "tcp":{
                  "acl-enh:flags-bitmask":{
                    "operator":"match",
                    "bitmask":2
                  }
                }
              },
              "actions":{
                "forwarding":"accept",
                "acl-enh:rate-limit":"20.00"
              }
            }
          ]
        }
      }
    ]
  }
}

```

Figure 9: An Example of Rate-Limit Incoming TCP SYNs (Message Body).

## Acknowledgements

Many thanks to Jon Shallow and Miguel Cros for the review and comments to the document, including prior to publishing the document.

Thanks to Qiufang Ma, Victor Lopez, Joe Clarke, and Mahesh Jethanandani for the comments and suggestions.

The IANA-maintained modules were generated using an XSLT stylesheet from the 'iana-yang' project (<https://github.com/llhotka/iana-yang>).

This work is partially supported by the European Commission under Horizon 2020 Secured autonomic traffic management for a Tera of SDN flows (Teraflow) project (grant agreement number 101015857).

## Authors' Addresses

Oscar Gonzalez de Dios  
Telefonica

Email: [oscar.gonzalezdedios@telefonica.com](mailto:oscar.gonzalezdedios@telefonica.com)

Samier Barguil  
Telefonica

Email: [samier.barguilgiraldo.ext@telefonica.com](mailto:samier.barguilgiraldo.ext@telefonica.com)

Mohamed Boucadair  
Orange

Email: [mohamed.boucadair@orange.com](mailto:mohamed.boucadair@orange.com)

Qin Wu  
Huawei

Email: [bill.wu@huawei.com](mailto:bill.wu@huawei.com)