Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 7, 2022

A. Choudhary
Cisco Systems
I. Chen
The MITRE Corporation

March 06, 2022

YANG Model for QoS Operational Parameters
draft-asechoud-rtgwg-qos-oper-model-10

Abstract

This document describes a YANG model for Quality of Service (QoS) operational parameters.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 7, 2022.

Table of Contents

1. Introduction 2
1.1. Tree Diagrams 2
2. Terminology 2
3. QoS Operational Model Design 3
4. Modules Tree Structure 4
5. Modules 6
5.1. ietf-qos-oper 6
6. Security Considerations 14
7. Acknowledgement 14
8. References 15
8.1. Normative References 15
8.2. Informative References 16
Authors' Addresses 16

1. Introduction

This document defines a base YANG [RFC6020] [RFC7950] data module for Quality of Service (QoS) operational parameters. Remote Procedure Calls (RPC) or notification definition is currently not part of this document and will be added later if necessary. QoS configuration modules are defined by [I-D.ietf-rtgwg-qos-model].

Editorial Note: (To be removed by RFC Editor)

This draft contains several placeholder values that need to be replaced with finalized values at the time of publication. Please apply the following replacements: o "XXXX" --> the assigned RFC value for this draft both in this draft and in the YANG models under the revision statement. o The "revision" date in model, in the format XXXX-XX-XX, needs to be updated with the date the draft gets approved.

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA) [RFC8342].

1.1. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340]

2. Terminology

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

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

3. QoS Operational Model Design

QoS operational model include QoS policy applied to an interface in each direction of traffic. For each QoS policy applied to an interface the model further includes counters for associated Classifiers, Meters and Queues in a particular direction. To modularize and for reusability, grouping have been defined for various counters of classifier, Meters and Queues. The target is assumed to be interface but the groupings can be used for any other target type where QoS policy is applied.

[I-D.ietf-rtgwg-qos-model] defines various building blocks for applying a QoS Policy on a target. It includes QoS Policy configuration, which is a container of various classifiers and corresponding actions which are configured for traffic conditioning. This drafts defines the various counters for these building blocks. ietf-qos-oper module defined in this draft augments ietf-interfaces [RFC8343] module.

Classifier statistics contains counters for packets and bytes matched to the traffic in a direction and also average rate at which traffic is hitting a classifier. Classification criterion may be based on IP, MPLS or Ethernet. Counters defined in this draft are agnostic to underlying data plane technology.

Statistics of meter is modeled based on commonly used algorithms in industry, Single Rate Tri Color Marking (srTCM) [RFC2697] meter, Two Rate Tri Color Marking (trTCM) [RFC2698] meter. Metering statistics includes counters corresponding to various rates configured. A metering container is referred by a metering identifier. This identifier could be a classifier name if the metering configuration is inline with classifier or it could be metering template name if the metering is configured as separate entity and associated with the classifier.

Queuing statistics includes counters corresponding to various queues associated with the policy. A queuing container is referred by queuing identifier. This identifier could be a classifier name if the queuing configuration is inline with classifier and hence there is one-to-one mapping between a classifier and a queue or it could be a separate queue identifier if one or more than one classifiers are associated with a queue.

4. Modules Tree Structure

This document defines counters for classifiers, meters and queues.

Classifier statistics consists of list of classifier entries identified by a classifier entry name. Classifier counters include matched packets, bytes and average rate of traffic matching a particular classifier.

Metering statistics consists of meters identified by an identifier. Metering counters include conform, exceed, violate and drop packets and bytes.

Queuing counters include instantaneous, peak, average queue length, as well as output conform, exceed, tail drop packets and bytes.

Named statistics is defined as statistics which are tagged by a name. This could be aggregated or non-aggregated. Aggregated named statistics is defined as counters which are aggregated across classifier entries in a policy applied to an interface in a particular direction. Non-aggregated named statistics are counters of classifier, metering or queuing which have the same tag name but maintained separately.

```
module: ietf-qos-oper
  augment /if:interfaces/if:interface:
    +--ro qos-interface-statistics
       +--ro stats-per-direction* []
          +--ro direction?               identityref
          +--ro policy-name?             string
          +--ro classifier-statistics* []
          |  +--ro classifier-entry-name?   string
          |  +--ro classified-pkts?         uint64
          |  +--ro classified-bytes?        uint64
          |  +--ro classified-rate?         uint64
          +--ro named-statistics* []
          |  +--ro stats-name?       string
          |  +--ro aggregated
          |  |  +--ro pkts?    uint64
          |  |  +--ro bytes?   uint64
          |  |  +--ro rate?    uint64
          |  +--ro non-aggregated
          |     +--ro classifier-statistics* []
          |     |  +--ro classifier-entry-name?   string
          |     |  +--ro classified-pkts?         uint64
          |     |  +--ro classified-bytes?        uint64
          |     |  +--ro classified-rate?         uint64
```

```
|     +--ro metering-statistics* []
|     |  +--ro meter-id?           string
|     |  +--ro conform-pkts?       uint64
|     |  +--ro conform-bytes?      uint64
|     |  +--ro conform-rate?       uint64
|     |  +--ro exceed-pkts?        uint64
|     |  +--ro exceed-bytes?       uint64
|     |  +--ro exceed-rate?        uint64
|     |  +--ro violate-pkts?       uint64
|     |  +--ro violate-bytes?      uint64
|     |  +--ro violate-rate?       uint64
|     |  +--ro meter-drop-pkts?    uint64
|     |  +--ro meter-drop-bytes?   uint64
|     +--ro queueing-statistics* []
|        +--ro queue-id?                    string
|        +--ro output-conform-pkts?         uint64
|        +--ro output-conform-bytes?        uint64
|        +--ro output-exceed-pkts?          uint64
|        +--ro output-exceed-bytes?         uint64
|        +--ro queue-current-size-bytes?    uint64
|        +--ro queue-average-size-bytes?    uint64
|        +--ro queue-peak-size-bytes?       uint64
|        +--ro tailed-drop-pkts?            uint64
|        +--ro tailed-drop-bytes?           uint64
|        +--ro red-drop-pkts?               uint64
|        +--ro red-drop-bytes?              uint64
|        +--ro red-ecn-marked-pkts?         uint64
|        +--ro red-ecn-marked-bytes?        uint64
|        +--ro wred-stats* []
|           +--ro profile-id?              uint64
|           +--ro red-drop-pkts?           uint64
|           +--ro red-drop-bytes?          uint64
|           +--ro red-ecn-marked-pkts?     uint64
|           +--ro red-ecn-marked-bytes?    uint64
+--ro metering-statistics* []
|  +--ro meter-id?           string
|  +--ro conform-pkts?       uint64
|  +--ro conform-bytes?      uint64
|  +--ro conform-rate?       uint64
|  +--ro exceed-pkts?        uint64
|  +--ro exceed-bytes?       uint64
|  +--ro exceed-rate?        uint64
|  +--ro violate-pkts?       uint64
|  +--ro violate-bytes?      uint64
|  +--ro violate-rate?       uint64
|  +--ro meter-drop-pkts?    uint64
|  +--ro meter-drop-bytes?   uint64
+--ro queueing-statistics* []
```

```
               +--ro queue-id?                    string
               +--ro output-conform-pkts?         uint64
               +--ro output-conform-bytes?        uint64
               +--ro output-exceed-pkts?          uint64
               +--ro output-exceed-bytes?         uint64
               +--ro queue-current-size-bytes?    uint64
               +--ro queue-average-size-bytes?    uint64
               +--ro queue-peak-size-bytes?       uint64
               +--ro tailed-drop-pkts?            uint64
               +--ro tailed-drop-bytes?           uint64
               +--ro red-drop-pkts?               uint64
               +--ro red-drop-bytes?              uint64
               +--ro red-ecn-marked-pkts?         uint64
               +--ro red-ecn-marked-bytes?        uint64
               +--ro wred-stats* []
                  +--ro profile-id?             uint64
                  +--ro red-drop-pkts?          uint64
                  +--ro red-drop-bytes?         uint64
                  +--ro red-ecn-marked-pkts?    uint64
                  +--ro red-ecn-marked-bytes?   uint64
```

5. Modules

5.1. ietf-qos-oper

<CODE BEGINS>file "ietf-qos-oper.yang"

```
module ietf-qos-oper {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-qos-oper";
  prefix oper;
  import ietf-interfaces {
    prefix if;
    reference
      "RFC8343: A YANG Data Model for Interface Management";
  }
  organization "IETF RTG (Routing Area) Working Group";
  contact
    "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
     WG List:  <mailto:rtgwg@ietf.org>
     Editor:   Aseem Choudhary
               <mailto:asechoud@cisco.com>";
  description
    "This module contains a collection of YANG definitions for
     qos operational specification.
     Copyright (c) 2019 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 Simplified 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 2022-03-06 {
      description
        "Latest revision for qos operational statistics";
      reference
        "RFC XXXX: YANG Model for QOS Operational Parameters";
    }
    identity direction {
      description
        "This is identity of traffic direction";
    }
    identity inbound {
      base direction;
      description
        "Direction of traffic coming into the network entry";
    }
    identity outbound {
      base direction;
      description
        "Direction of traffic going out of the network entry";
    }
    grouping classifier-entry-stats {
      description
        "
         This group defines the classifier filter counters of
         each classifier entry
        ";
      leaf classified-pkts {
        type uint64;
        description
          " Number of total packets which filtered
            to a classifier-entry";
      }
      leaf classified-bytes {
        type uint64;
        description
          " Number of total bytes which filtered
            to a classifier-entry";
      }
      leaf classified-rate {
        type uint64;
```

```
        units "bits-per-second";
        description
          " Rate of average data flow through a
            classifier-entry";
      }
    }
    grouping named-stats {
      description
        "QoS matching statistics associated with a stats-name";
        leaf pkts {
          type uint64;
          description
            " Number of total matched packets associated
              to a statistics name";
        }
        leaf bytes {
          type uint64;
          description
            " Number of total matched bytes associated
              to a statistics name";
        }
        leaf rate {
          type uint64;
          units "bits-per-second";
          description
            " Rate of average matched data which is associated
              to a statistics name";
        }
    }
    grouping queue-stats {
      description
        "Queuing Counters";
      leaf output-conform-pkts {
        type uint64;
        description
          "Number of packets transmitted from queue ";
      }
      leaf output-conform-bytes {
        type uint64;
        description
          "Number of bytes transmitted from queue ";
      }
      leaf output-exceed-pkts {
        type uint64;
        description
          "Number of packets transmitted from queue ";
      }
      leaf output-exceed-bytes {
```

```
        type uint64;
        description
          "Number of bytes transmitted from queue ";
      }
      leaf queue-current-size-bytes {
        type uint64;
        description
          "Number of bytes currently buffered ";
      }
      leaf queue-average-size-bytes {
        type uint64;
        description
          "Average queue size in number of bytes";
      }
      leaf queue-peak-size-bytes {
        type uint64;
        description
          "Peak buffer queue size in bytes ";
      }
      leaf tailed-drop-pkts {
        type uint64;
        description
          "Total number of packets tail-dropped ";
      }
      leaf tailed-drop-bytes {
        type uint64;
        description
          "Total number of bytes tail-dropped ";
      }
      leaf red-drop-pkts {
        type uint64;
        description
          "Total number of packets dropped through RED mechanism";
      }
      leaf red-drop-bytes {
        type uint64;
        description
          "Total number of bytes dropped through RED mechanism";
      }
      leaf red-ecn-marked-pkts {
        type uint64;
        description
          "Total number of packets ECN marked through RED mechanism";
      }
      leaf red-ecn-marked-bytes {
        type uint64;
        description
          "Total number of bytes ECN marked through RED mechanism";
```

```
      }
    }
    grouping meter-stats {
      description
        "Metering counters";
      leaf conform-pkts {
        type uint64;
        description
          "Number of conform packets";
      }
      leaf conform-bytes {
        type uint64;
        description
          "Bytes of conform packets";
      }
      leaf conform-rate {
        type uint64;
        units "bits-per-second";
        description
          "Traffic Rate measured as conformimg";
      }
      leaf exceed-pkts {
        type uint64;
        description
          "Number of packets counted as exceeding";
      }
      leaf exceed-bytes {
        type uint64;
        description
          "Bytes of packets counted as exceeding";
      }
      leaf exceed-rate {
        type uint64;
        units "bits-per-second";
        description
          "Traffic Rate measured as exceeding";
      }
      leaf violate-pkts {
        type uint64;
        description
          "Number of packets counted as violating";
      }
      leaf violate-bytes {
        type uint64;
        description
          "Bytes of packets counted as violating";
      }
      leaf violate-rate {
```

```
        type uint64;
        units "bits-per-second";
        description
          "Traffic Rate measured as violating";
      }
      leaf meter-drop-pkts {
        type uint64;
        description
          "Number of packets dropped by meter";
      }
      leaf meter-drop-bytes {
        type uint64;
        description
          "Bytes of packets dropped by meter";
      }
    }
    grouping classifier-entry-statistics {
      description
        "Statistics for a classifier entry";
      leaf classifier-entry-name {
        type string;
        description
          "Classifier Entry Name";
      }
      uses classifier-entry-stats;
    }

    grouping queuing-stats {
      description
        "Statistics for a queue";
      leaf queue-id {
        type string;
        description
          "Queue Identifier";
      }
      uses queue-stats;
      list wred-stats {
        config false;
        description
          "Qos RED statistics for each color of traffic";
        leaf profile-id {
          type uint64;
          description
            "profile-id for each color of traffic";
        }
        leaf red-drop-pkts {
          type uint64;
          description
```

```
            "Total number of packets dropped through RED mechanism";
        }
        leaf red-drop-bytes {
          type uint64;
          description
            "Total number of bytes dropped through RED mechanism";
        }
        leaf red-ecn-marked-pkts {
          type uint64;
          description
            "Total number of packets ECN marked through RED mechanism";
        }
        leaf red-ecn-marked-bytes {
          type uint64;
          description
            "Total number of bytes ECN marked through RED mechanism";
        }
      }
    }

    grouping metering-stats {
      description
        "Statistics for a meter";
      leaf meter-id {
        type string;
        description
          "Meter Identifier";
      }
      uses meter-stats;
    }

    augment "/if:interfaces/if:interface" {
      description
        "Augments Qos Target Entry to Interface module";

      container qos-interface-statistics {
        config false;
        description
          "Qos Interface statistics";

        list stats-per-direction {
          description
            "Qos Interface statistics for ingress or egress direction";

          leaf direction {
            type identityref {
              base direction;
            }
```

```
            description
              "Direction of the traffic flow either inbound
               or outbound";
          }
          leaf policy-name {
            type string;
            description
              "Policy entry name for single level policy as well as
               for Hierarchical policies. For Hierarchical policies,
               this represent relative path as well as the last level
               policy name.";
          }

          list classifier-statistics {
            description
              "Classifier Statistics for each Classifier Entry in a
               Policy applied in a particular direction";
            reference
              "RFC3289: Section 6";
            uses classifier-entry-statistics;
          }
          list named-statistics {
            config false;
            description
              "Statistics for a statistics-name";
            leaf stats-name {
              type string;
              description
                "Statistics name";
            }
            container aggregated {
              description
                "Matched aggregated statistics for a statistics-name";
              uses named-stats;
            }
            container non-aggregated {
              description
                "Statistics for non-aggregated statistics-name";
              list classifier-statistics {
                description
                  "Classifier Statistics for each Classifier Entry in a
                   Policy applied in a particular direction";
                uses classifier-entry-statistics;
              }
              list metering-statistics {
                config false;
                description
                  "Statistics for each Meter associated with
```

```
                    the Policy";
                  reference
                    "RFC2697: A Single Rate Three Color Marker
                     RFC2698: A Two Rate Three Color Marker";
                  uses metering-stats;
                }
                list queueing-statistics {
                  config false;
                  description
                    "Statistics for each Queue associated with
                     the Policy";
                  uses queuing-stats;
                }
              }
            }
            list metering-statistics {
              config false;
              description
                "Statistics for each Meter associated with the Policy";
              reference
                "RFC2697: A Single Rate Three Color Marker
                 RFC2698: A Two Rate Three Color Marker";
              uses metering-stats;
            }
            list queueing-statistics {
              config false;
              description
                "Statistics for each Queue associated with the Policy";
              uses queuing-stats;
            }
          }
        }
      }
    }
```

<CODE ENDS>

6. Security Considerations

7. Acknowledgement

MITRE has approved this document for Public Release, Distribution Unlimited, with Public Release Case Number 20-0518. The author's affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions or viewpoints expressed by the author.

8. References

8.1. Normative References

[I-D.ietf-rtgwg-qos-model]
Choudhary, A., Jethanandani, M., Aries, E., and I. Chen, "YANG Models for Quality of Service (QoS)", draft-ietf-rtgwg-qos-model-07 (work in progress), March 2022.

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

[RFC2697] Heinanen, J. and R. Guerin, "A Single Rate Three Color Marker", RFC 2697, DOI 10.17487/RFC2697, September 1999, <https://www.rfc-editor.org/info/rfc2697>.

[RFC2698] Heinanen, J. and R. Guerin, "A Two Rate Three Color Marker", RFC 2698, DOI 10.17487/RFC2698, September 1999, <https://www.rfc-editor.org/info/rfc2698>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

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

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

8.2. Informative References

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

Authors' Addresses

Aseem Choudhary
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95134
US

Email: asechoud@cisco.com

Ing-Wher Chen
The MITRE Corporation

Email: ingwherchen@mitre.org