Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 3 September 2026

A. Choudhary
Cisco Systems
M. Jethanandani
Kloud Services
E. Aries
Juniper Networks
I. Chen
The MITRE Corporation

2 March 2026

YANG Models for Quality of Service (QoS) in IP networks
draft-ietf-rtgwg-qos-model-15

Abstract

This document describes a YANG model for management of Quality of Service (QoS) in IP networks.

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 3 September 2026.

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG 1.1 [RFC7950] model for Quality of Service (QoS) configuration and operational parameters as they relate to IP networks. Traffic Policy module defines the basic building blocks to define a classifier, policy and target. QoS Action module defines QoS action related parameters. Differentiated Servicers (Diffserv) policy, Queue policy and Schedular policy modules augments Traffic policy with various packet match and action parameters. Each of these policies are defined as separate modules. The Diffserv module is based on Diffserv architecture, and various references have been made to available standard architecture documents. QoS statistics counters are defined in QoS Operational module.

While there are other approaches to offering QoS services, this document focuses on Diffserv as the approach for providers to offer services to different customers based on their network QoS objectives. The DSCP markings are applied by upstream node or by the edge router on entry to the Diffserv domain. The traffic streams are differentiated based on Diffserv Code Points (DSCP) carried in the IP header of each packet and per-hop-behavior according to the DSCP marking is applied.

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

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

1.1. Note to RFC Editor

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:

* "XXXX" --> the assigned RFC value for this draft both in this draft and in the yang modules under the revision statement.

* The "revision" date in model, in the format XXXX-XX-XX, needs to be updated with the date the draft gets approved.

1.2. 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.

1.3. Definitions and Acronyms

This document uses definitions and acronyms defined in Definitions of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers [RFC2474], An Architecture for Differentiated Services [RFC2475], and other documents. Here are some of them.

* Active Queue Management (AQM): An activity that marks or drops packets before the queue is full.

* Behavior Aggregate (BA): A DS behavior aggregate.

* Classifier: an entity which selects packets based on the content of packet headers according to defined rules.

* CoDel: Controlled Delay AQM algorithm

* DS behavior aggregate: A collection of packets with the same DS codepoint crossing a link in a particular direction.

* DS code point: A specific value of the DSCP portion of the DS field, used to select a PHB.

* Diffserv: Differentiated Services enhancements to the Internet protocol are intended to enable scalable service discrimination in the Internet without the need for per-flow state and signaling at every hop. A variety of services may be built from a small, well-defined set of building blocks which are deployed in network nodes.

* DSCP: Differentiated Services Code Point

* ECN: Explicit Congestion Notification

* FQ-CoDel: Flow Queue Controlled Delay AQM algorithm

* Marking: the process of setting the DS codepoint in a packet based on defined rules; pre-marking, re-marking.

* Metering: the process of measuring the temporal properties (e.g., rate) of a traffic stream selected by a classifier. The instantaneous state of this process may be used to affect the operation of a marker, shaper, or dropper, and/or may be used for accounting and measurement purposes.

* MF Classifier: A multi-field (MF) classifier which selects packets based on the content of some arbitrary number of header fields; typically some combination of source address, destination address, DS field, protocol ID, source port and destination port.

* Per-Hop-Behavior (PHB): A description of the externally observable forwarding treatment applied at a differentiated services-compliant node to a behavior aggregate [RFC2474]. The description of a PHB SHOULD be sufficiently detailed to allow the construction of predictable services, as documented in [RFC2475].

* Policing: the process of discarding packets (by a dropper) within a traffic stream in accordance with the state of a corresponding meter enforcing a traffic profile.

* Policy: A set of rules each with a set of conditions and a set of actions.

* RED: Random Early Detection

* Shaping: the process of delaying packets within a traffic stream to cause it to conform to some defined traffic profile.

* Target: The network physical or logical element (e.g., interface or device) where a Policy is applied.

* Traffic-Group: a logical collection of packets or flows that share similar service requirements and are treated collectively for traffic management purposes, including prioritization, shaping, policing, or resource reservation.

* WRED: Weighted Random Early Detection

2. QoS Model Design

The overall objective of this model design is to offer a base set of modules that can then be further augmented to provide other models, such as Differentiated Services (Diffserv) and Policy Based Routing (PBR). Those models would augment the base modules defined in this draft to define their own model. This document defines one such model, Diffserv. The design also takes into consideration that vendors might want to extend the model to add their own extensions,

such as the Traffic Policy module, to provide constructs of policy for different QoS functionalities."

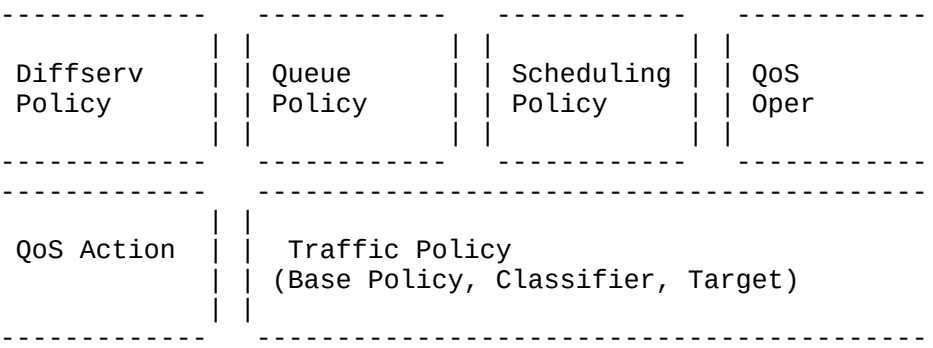

Figure 1: QoS Yang Modules Layout

The above figure depicts the design of the QoS model. It uses Traffic Policy and QoS Action as base modules, which are augmented by the remaining modules on the top to define different aspects of QoS.

The Classifier is defined as part of Traffic Policy Module. A classifier consists of set of filters and an operation that is performed on those filters. Each filter defines a rule on a particular packet header or packet metadata field that dictate how the packet will be classified based on that particular field, e.g. whether it will be classified based on DSCP, or will be classified based on source IPv4 address prefix. The operation defines whether these filters are logically AND or OR. The filters may be based on a combination of values or range of values of different packet header fields or packet metadata fields. An example of how these filter definitions are used can be seen in the Diffserv model. The Diffserv model augments the filter node in the Classifier to add in Diffserv specific filter parameters.

One or more packet conditioning functions may be applied to a classifier which may drop, mark, or delay packets. These are defined in the QoS Action module. A set of classifiers with corresponding conditioning functions when arranged in order of priority represent a QoS policy. These policies are defined in Traffic Policy module.

The QoS Action module defines marking, metering and queuing. QoS actions are configured in line or referred to in Diffserv, Queuing, and Scheduling Policy modules.

A Meter meters each packet and passes the packet and the metering result to the Marker. Meter is modeled based on commonly used algorithms in industry, A Single Rate Three Color Marker (srTCM) [RFC2697], A Two Rate Three Color Marker (trTCM) [RFC2698], and A Single Rate Two Color Marker. Different vendors can extend it with other types of meters as well. Meter is defined in QoS Action module.

Queue Policy module allows queues to be referred in a policy. The match is based on Traffic Group and action parameters are used as defined in QoS Action module.

Scheduling Policy module defines a set of scheduling parameters and associates Queue Policy with it.

QoS counters are defined in QoS Operational (ietf-qos-oper) module. It includes counters for Classifiers, Meters and Queues associated with a QoS policy applied in each direction of traffic. To modularize and for reusability, grouping have been defined for various counters of Classifier, Meters and Queues. A Traffic Policy is applied to a Target. A Target is a network logical or physical interface or sub-interface. However, the groupings can be reused to define a traffic policy which then can be applied to any logical or physical entities such as a Policy to form hierarchical policies.

In addition, ietf-qos-types module is defined which serves as the central repository for shared identities, groupings, and typedefs utilized across the entire QoS model. It defines the fundamental building blocks such as traffic direction, rate units, and filter types that ensure consistency between the other functional modules and future extensibility.

3. Diffserv Model Design

Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers [RFC2475] describes the architecture as a simple model where traffic entering a network is classified and possibly conditioned at the boundary of the network and assigned a different Behavior Aggregate (BA). Each BA is identified by a specific value of DSCP, and is used to select a Per Hop Behavior (PHB) at the edge node as well as at the core of a DS domain. In a DiffServ-compliant node, the PHB is selected based on the DSCP value found in the IP header of an incoming packet. Specifically, a Behavior Aggregate (BA) classifier is used to map the DSCP to a particular PHB, which determines the externally observable forwarding treatment the packet receives at that hop. The mapping of a DSCP to a PHB is local to each node. While a single DSCP identifies a Behavior Aggregate (BA) across the network, different nodes may be

configured to apply different PHBs to that same BA based on local policy and resource availability. Multiple DSCP values can be mapped to the same PHB at a single node. This allows for a simplified core configuration where various traffic streams (identified by different DSCPs at the edge) receive the same forwarding treatment within the backbone. This YANG model facilitates this by allowing the DiffServ module to augment the base Traffic Policy module. This enables the configuration of classifiers (like the BA classifier) that select packets based on DSCP ranges and associate them with specific actions (PHBs), such as marking, metering, or queuing, at each network interface.

The packet classification policy identifies the subset of traffic which may receive a Diffserv by being conditioned or mapped. Packet classifiers select packets within a stream based on the content of some portion of the packet header. There are two types of classifiers, the BA classifier, and the Multi-Field (MF) classifier which selects packets based on a value which is combination of value or range of values of one or more header fields. Diffserv domain gateway router classifies and remarks packet to suit domain internal Diffserv policy and here RFC8436 [RFC8436] must be respected. Diffserv Domain internal routers may classify by ranges of DSCPs. This allows keeping e.g. a backbone DS config simple and comprehensible. In the Diffserv Policy module, this packet classification is realized by augmenting the classifier in Traffic Policy module.

Traffic conditioning includes metering, shaping and/or marking. A Meter is used to measure the traffic against a given traffic profile. The traffic profile specifies the temporal property of the traffic. A packet that arrives is first determined to be in or out of the profile, which will result in the action of marked, dropped or shaped. This is realized in vendor specific modules based on the parameters defined in action module. The metering parameters are augmented to the QoS policy module when metering is defined inline, and to the metering template when metering profile is referred in policy module.

3.1. A Single Rate Three Color Marker

This document defines support for A Single Rate Three Color Marker (srTCM) [RFC2697], which is one of the components in a An Architecture for Differentiated Services [RFC2475]. The srTCM meters a traffic stream and marks its packets according to three traffic parameters, Committed Information Rate (CIR), Committed Burst Size (CBS), and Excess Burst Size (EBS), to be either green, yellow, or red. A packet is marked green if it doesn’t exceed the CBS, yellow if it does exceed the CBS, but not the EBS, and red otherwise.

3.2. A Two Rate Three Color Marker

This document defines support for A Two Rate Three Color Marker (trTCM) [RFC2698], which is one of the components in a An Architecture for Differentiated Services [RFC2475]. The trTCM meters a traffic stream and marks its packets according to two rates, Peak Information Rate (PIR), the Committed Information Rate (CIR), and their associated burst sizes to either green, yellow, or red. A packet is marked red if it exceeds the PIR. Otherwise, it is marked yellow or green depending on whether it exceeds or does not exceed the CIR.

4. YANG Modules

Modules defined in this draft import definitions from Common YANG Data Types [RFC9911] and A YANG Data Model for Interface Management [RFC8343].

4.1. Traffic Policy Module

Traffic Policy module contains list of classifiers identified by a classifier name. Each classifier MAY contain a list of filters. While a single filter is sufficient for simple classification, like one would expect at a core interface, a list of filters allows for more sophisticated logic to be applied on the ingress edge nodes.

This module also contains list of policy objects identified by a policy name and policy type which MUST be provided. Policy-type is of type identity and is populated in a vendor specific manner. With different values of policy types, each vendor MAY define their own construct of policy for different QoS functionalities. Each vendor MAY augment classifier entry in a policy definition with a set of actions.

The module also augments A YANG Data Model for Interface Management [RFC8343] module to add a target policy. A single policy of a particular policy-type can be applied on an interface in each direction of traffic.

The figure below presents the tree diagram for the ietf-traffic-policy module. This module serves as the central framework for the QoS module, defining how Classifiers are associated with Actions (traffic treatment) with a Policy. It also includes the qos-target-poilcy, which facilitates the attachment of these Policies to specific network Targets, such as interfaces, in either the inbound or outbound direction.

```
module: ietf-traffic-policy
  +--rw classifiers
  |  +--rw classifier* [name]
  |     +--rw name                string
  |     +--rw description?        string
  |     +--rw filter-operation?   qos-types:match-operation-type
  |     +--rw filter* [type logical-not]
  |           ...
  +--rw policies
     +--rw policy* [name type]
        +--rw name           string
        +--rw type           identityref
        +--rw description?   string
        +--rw classifier* [name]
              ...

  augment /if:interfaces/if:interface:
    +--rw qos-target-policy* [direction type]
       +--rw direction    identityref
       +--rw type         identityref
       +--rw name         -> /policies/policy/name
```

Figure 2: ietf-traffic-policy abridged tree diagram

```
<CODE BEGINS> file "ietf-traffic-policy@2026-03-02.yang"
module ietf-traffic-policy {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-traffic-policy";
  prefix traffic-policy;

  import ietf-interfaces {
    prefix if;
  }
  import ietf-qos-types {
    prefix qos-types;
    reference
      "RFC XXXX: YANG Data Models for Quality of Service (QoS).";
  }

  organization
    "IETF Routing Area Working Group.";

  contact
    "WG Web:   <https://datatracker.ietf.org/wg/rtgwg/>
     WG List:  <mailto:rtgwg@ietf.org>

     Editor:   Aseem Choudhary
               <mailto:asechoud@cisco.com>
```

```
        Editor:   Mahesh Jethanandani
                  <mailto:mjethanandani@gmail.com>";

     description
       "This module contains a collection of YANG definitions for
        configuring QoS Traffic Policies.

        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 2026-03-02 {
       description
         "Initial version.";
       reference
         "RFC XXXX: YANG Models for Quality of Service (QoS).";
     }

     grouping filters {
       description
         "Filters types in a Classifier entry.";

       leaf type {
         type identityref {
           base qos-types:filter-type;
         }
         description
           "This leaf defines the type of filter.";
       }

       leaf logical-not {
         type boolean;
         description
           "This is logical-not operator for filter values and/or value
            ranges. When true, it indicates filter looks for absence
            of values and/or value ranges defined by the filter.";
       }
     }
```

```
    grouping generic-classifier-attr {
      description
        "Generic Classifier attributes like name,
         description, operation type.";

      leaf name {
        type string;
        description
          "A name for Classifier entry.";
      }

      leaf description {
        type string;
        description
          "Description statement for classifier entry.";
      }

      leaf filter-operation {
        type qos-types:match-operation-type;
        description
          "match-any or match-all filters.";
      }
    }

    grouping inline-attr {
      description
        "Attributes of inline classifier in a policy.";

      leaf filter-operation {
        type qos-types:match-operation-type;
        description
          "match-any or match-all filters.";
      }

      list filter {
        key "type logical-not";
        uses filters;
        description
          "Filters configured inline in a policy.";
      }
    }

    container classifiers {
      description
        "List of classifier entries.";

      list classifier {
        key "name";
```

```
        description
          "Each classifier entry contains a list of filters.";

        uses generic-classifier-attr;

        list filter {
          key "type logical-not";
          uses filters;
          description
            "Filter entries for a classifier.";
        }
      }
    }

    container policies {
      description
        "List of policy templates.";

      list policy {
        key "name type";
        description
          "Policy template.";

        leaf name {
          type string;
          description
            "A name for the policy.";
        }

        leaf type {
          type identityref {
            base qos-types:policy-type;
          }
          description
            "The type of policy.";
        }

        leaf description {
          type string;
          description
            "A description for policy.";
        }

        list classifier {
          key "name";
          ordered-by user;
          description
            "Classifier configuration in a policy.";
```

```
          leaf name {
            type union {
              type string;
              type leafref {
                path "/traffic-policy:classifiers/" +
                "traffic-policy:classifier/traffic-policy:name";
              }
            }
            description
              "A name for classifier entry.";
          }

          container inline {
            presence "This container is present for when the device
                      is configured for inline classification.";
            uses inline-attr;
            description
              "Container for when an inline classification
               is desired.";
          }

          list action {
            key "type";
            ordered-by user;
            description
              "List of action for a Classifier.";

            leaf type {
              type identityref {
                base qos-types:action-type;
              }
              description
                "Type of action.";
            }

          }
        }
      }
    }

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

      list qos-target-policy {
        key "direction type";
        description
          "Policy target for inbound or outbound direction.";
```

```
          leaf direction {
            type identityref {
              base qos-types:direction;
            }
            description
              "Direction of the traffic flow either inbound or
               outbound.";
          }

          leaf type {
            type identityref {
              base qos-types:policy-type;
            }
            description
              "Policy type.";
          }

          leaf name {
            type leafref  {
              path "/traffic-policy:policies/" +
                  "traffic-policy:policy/traffic-policy:name";
            }
            mandatory true;
            description
              "A name for the Policy.";
          }
        }
      }
    }
    <CODE ENDS>
```

Figure 3: ietf-traffic-policy module

4.2. QoS Action Module

QoS Action module includes actions such marking, metering, or queuing. Metering, marking, queuing, and scheduling actions are realized by augmenting the Traffic Policy module. Marking sets Diffserv codepoint value in the classified packet.

The figure below depicts the tree diagram for the ietf-qos-action module. This module defines the common set of Actions that can be applied to traffic once it has been classified. These actions include fundamental QoS operations such as Marking (setting DSCP or precedence values), Metering, and Queuing. The structure is designed to be extensible, allowing for the addition of new action types as required by specific hardware capabilities.

```
module: ietf-qos-action
  +--rw meters
  |  +--rw meter* [name]
  |     +--rw name                                  string
  |     +--rw (meter-types)?
  |           ...
  +--rw queues
     +--rw queue* [name]
        +--rw name     string
        +--rw queue
              ...
```

Figure 3: ietf-qos-actions abridged tree diagram

```
<CODE BEGINS> file "ietf-qos-action@2026-03-02.yang"
module ietf-qos-action {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-qos-action";
  prefix qos-action;

  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 9911: Common YANG Data Types";
  }
  import ietf-qos-types {
    prefix qos-types;
    reference
      "RFC XXXX: YANG Model for Quality of Service (QoS).";
  }

  organization
    "IETF Routing Area Working Group";

  contact
    "WG Web:   <https://datatracker.ietf.org/wg/rtgwg/>
     WG List:  <mailto:rtgwg@ietf.org>

     Editor:   Aseem Choudhary
               <mailto:asechoud@cisco.com>
     Editor:   Mahesh Jethanandani
               <mailto:mjethanandani@gmail.com>";

  description
    "This module contains a collection of YANG definitions for
     configuring QoS actions.

     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 2026-03-02 {
      description
        "Initial version";
      reference
        "RFC XXXX: YANG Models for Quality of Service (QoS).";
    }

    grouping rate-value-unit {
      leaf value {
        type uint64;
        description
          "Value for the rate that is configured, either in
           bits-per-second or percentage.";
      }

      leaf unit {
        type identityref {
          base qos-types:rate-unit-type;
        }
        default qos-types:bits-per-second;
        description
          "The unit for the value above, either in bits-per-second or
           percentage.";
      }
      description
        "Grouping for configuration of value and unit.";
    }

    grouping meter-action {
      description
        "Configuration of meter actions.";

      leaf action-type {
        type identityref {
          base qos-types:meter-action-type;
```

```
        }
        description
          "Action type for metering.";
      }

      leaf dscp-mark {
        type inet:dscp;
        must "../action-type = 'qos-types:action-mark'" {
          error-message
            "DSCP must be marked when action-type is action-mark.";
        }
        description
          "DSCP marking";
      }

      leaf traffic-group-mark {
        type string;
        must "../action-type = 'qos-types:action-mark'" {
          error-message
            "Traffic-group must be marked when action-type is
             action-mark.";
        }
        description
          "A name to mark as a traffic group.";
      }
    }

    grouping single-rate-two-color-meter {
      container single-rate-two-color-meter {
        description
          "A container for Basic Single Rate Two Color Marker.";

        leaf committed-information-rate {
          type uint64;
          units "bytes-per-second";
          description
            "The value for Committed Information Rate (CIR).";
        }

        leaf committed-burst-size {
          type uint64;
          units "bytes";
          description
            "The value for Committed Burst Size (CBS).";
        }

        container conform-action {
          uses meter-action;
```

```
          description
            "A packet is marked conforming, or green if it does not
             exceed CIR.";
        }

        container exceed-action {
          uses meter-action;
          description
            "A packet is marked exceeding, or yellow if it does
             exceed CIR.";
        }
      }
      description
        "A Basic Single Rate Two Color Meter.";
    }

    grouping single-rate-three-color-meter {
      container single-rate-three-color-meter {
        description
          "Container that defines the Single Rate Three Color Meter
           (srTCM).";

        leaf committed-information-rate {
          type uint64;
          units "bytes-per-second";
          description
            "The value for Committed Information Rate (CIR).";
          reference
            "RFC 2697: A Single Rate Three Color Marker.";
        }

        leaf committed-burst-size {
          type uint64;
          units "bytes";
          description
            "The value for Committed Burst Size (CBS).";
          reference
            "RFC 2697: A Single Rate Three Color Marker.";
        }

        leaf excess-burst-size {
          type uint64;
          units "bytes";
          description
            "The value for Excess Burst Size (EBS).";
          reference
            "RFC 2697: A Single Rate Three Color Marker.";
        }
```

```
        container conform-action {
          uses meter-action;
          description
            "A packet is marked conforming, or green if it does not
             exceed CBS.";
          reference
            "RFC 2697: A Single Rate Three Color Marker.";
        }

        container exceed-action {
          uses meter-action;
          description
            "A packet is marked exceeding, or yellow if it does
             exceed CBS, but not the EBS.";
          reference
            "RFC 2697: A Single Rate Three Color Marker.";
        }

        container violate-action {
          uses meter-action;
          description
            "A packet is marked violating, or red if it is exceeds
             both CBS and EBS.";
          reference
            "RFC 2697: A Single Rate Three Color Marker.";
        }
      }
      description
        "Singe Rate Three Color Meter (srTCM).";
      reference
        "RFC 2697: A Single Rate Three Color Marker.";
    }

    grouping two-rate-three-color-meter {
      container two-rate-three-color-meter {
        description
          "A container that defines A Two Rate Three Color Meter
           (trTCM).";

        leaf committed-information-rate {
          type uint64;
          units "bytes-per-second";
          description
            "The value for Committed Information Rate (CIR).";
          reference
            "RFC 2698: A Two Rate Three Color Marker.";
        }
```

```
          leaf committed-burst-size {
            type uint64;
            units "bytes";
            description
              "The value for Committed Burst Size (CBS).";
            reference
              "RFC 2698: A Two Rate Three Color Marker.";
          }

          leaf peak-information-rate {
            type uint64;
            units "bytes-per-second";
            description
              "The value for Peak Information Rate (PIR).";
            reference
              "RFC 2698: A Two Rate Three Color Marker.";
          }

          leaf peak-burst-size {
            type uint64;
            units "bytes";
            description
              "The value for Peak Burst Size (PBS).";
            reference
              "RFC 2698: A Two Rate Three Color Marker.";
          }

          container conform-action {
            uses meter-action;
            description
              "A packet is marked conforming, or green, if it does not
               exceed CIR.";
            reference
              "RFC 2698: A Two Rate Three Color Marker.";
          }

          container exceed-action {
            uses meter-action;
            description
              "A packet is marked exceeding, or yellow, if it exceeds
               CIR but not PIR.";
            reference
              "RFC 2698: A Two Rate Three Color Marker.";
          }

          container violate-action {
            uses meter-action;
            description
```

```
            "A packet is marked as violating, or red, if it exceeds
             both CIR and PIR.";
          reference
            "RFC 2698: A Two Rate Three Color Marker.";
        }
      }
      description
        "Container for a Two Rate Three Color Marker.";
    }

    grouping meter {
      choice meter-types {
        case single-rate-two-color-meter-type {
          uses single-rate-two-color-meter;
        }

        case single-rate-three-color-meter-type {
          uses single-rate-three-color-meter;
        }

        case two-rate-three-color-meter-type {
          uses two-rate-three-color-meter;
        }
        description
          "A meter action based on choice of meter action type.";
      }
      description
        "Meter attributes.";
    }

    container meters {
      description
        "List of meter templates.";

      list meter {
        key "name";
        description
          "Meter entry template.";

        leaf name {
          type string;
          description
            "A name to identify the meter.";
        }
        uses meter;
      }
    }
```

```
    grouping meter-reference {
      container meter {
        leaf name {
          type leafref  {
            path "/qos-action:meters/" +
                 "qos-action:meter/qos-action:name";
          }
          mandatory true;
          description
            "This leaf defines name of the meter referenced.";
        }

        leaf type {
          type identityref {
            base qos-types:meter-type;
          }
          mandatory true;
          description
            "This leaf defines type of the meter.";
        }
        description
          "The name and type for meter reference.";
      }
      description
        "Grouping to define meter reference.";
    }

    grouping qos-count {
      container qos-count {
      if-feature qos-types:qos-count;

        leaf qos-count-action {
          type empty;
          description
            "Take an action if this is defined.";
        }
        description
          "Container for whether to take an action for QoS count.";
      }
      description
        "The qos-count action grouping.";
    }

    grouping named-qos-count {
      container named-qos-count {
      if-feature qos-types:named-qos-count;
        leaf name {
          type string;
```

```
          description
            "A name for named QoS count.";
        }
        description
          "The named traffic counter action.";
      }
      description
        "The named traffic counter action grouping.";
    }

    grouping discard {
      container discard {
        leaf discard {
          type empty;
          description
            "If defined this enables discard action.";
        }
        description
          "Container for discard action.";
      }
      description
        "Grouping for discard action.";
    }

    grouping priority {
      container priority {
        leaf level {
          type uint8;
          description
            "Priority level.";
        }
        description
          "Container for priority.";
      }
      description
        "Grouping for priority.";
    }

    grouping min-rate {
      container min-rate {
        uses rate-value-unit;
        description
          "Minimum guaranteed bandwidth.";
      }
      description
        "Grouping for minimum rate.";
    }
```

```
    grouping dscp-marking {
      container dscp {
        leaf dscp {
          type inet:dscp;
          description
            "DSCP marking.";
        }
        description
          "Container for DSCP marking.";
      }
      description
        "Grouping for DSCP marking.";
    }

    grouping traffic-group-marking {
      container traffic-group {
        leaf traffic-group {
          type string;
          description
            "A name for traffic group.";
        }
        description
          "Container for traffic group marking container.";
      }
      description
        "Grouping for traffic group marking.";
    }

    grouping child-policy {
      container child-policy {
        if-feature qos-types:child-policy;
        leaf name {
          type string;
          description
            "A name for hierarchical policy.";
        }
        description
          "Hierarchical Policy configuration container.";
      }
      description
        "Grouping of Hierarchical Policy configuration.";
    }

    grouping max-rate {
      container max-rate {
        uses rate-value-unit;

        leaf burst-value {
```

```
          type uint64;
          description
            "Burst value in bytes.";
        }

        leaf burst-unit {
          type identityref {
            base qos-types:burst-unit-type;
          }
          default qos-types:bytes;
          description
            "Burst value in bytes or milliseconds, with a default
             of bytes.";
        }
        description
          "Maximum rate attributes container.";
      }
      description
        "Grouping for maximum rate attributes.";
    }

    grouping red-config-parameters {
      leaf min-threshold-val {
        type uint64;
        description
          "Minimum value of RED threshold.";
      }

      leaf min-threshold-unit {
        type identityref {
          base qos-types:red-threshold-unit;
        }
        default qos-types:red-threshold-bytes;
        description
          "The unit for minimum RED threshold value, with the default
           of bytes.";
      }

      leaf max-threshold-val {
        type uint64;
        description
          "Maximum value of RED threshold.";
      }

      leaf max-threshold-unit {
        type identityref {
          base qos-types:red-threshold-unit;
        }
```

```
        default qos-types:red-threshold-bytes;
        description
          "The unit for maximum RED threshold value, with the default
           of bytes.";
      }

      leaf weight {
        type uint8;
        description
          "The decay factor for the average queue size
           calculation. The numbers is a 2's exponent.";
      }

      leaf max-probability-val {
        type uint64;
        description
          "Value of maximum probability value. This value need
           be interpreted along with max-probability-unit";
      }

      leaf max-probability-unit {
        type identityref {
          base qos-types:probability-unit;
        }
        default qos-types:probability-pc;
        description
         "Probability unit type, with a default of percentage.";
      }
      description
        "Random Early Detect (RED) configuration parameters.";
    }

    grouping codel-config-parameters {
      leaf target {
        type uint64;
        units "microseconds";
        default 5000;
        description
          "Target time in microseconds spent by each packet in queue.";
      }

      leaf interval {
        type uint64;
        units "microseconds";
        default 100000;
        description
          "The time in microsecond for minimum interval for congestion
           to persist before algorithm kicks in. ";
```

```
      }
      description
        "CoDel configuration parameters.";
    }

    grouping queue {
      container queue {
        uses priority;
        uses min-rate;
        uses max-rate;
        container algorithmic-drop {
          choice drop-algorithm {
            case tail-drop {
              container tail-drop {
                leaf tail-drop {
                  type empty;
                  description
                    "When defined, tail drop algorithm is enabled.";
                }
                description
                  "Tail Drop configuration container.";
              }
              description
                "Tail Drop choice.";
            }

            case red {
              container red {
                uses red-config-parameters;
                leaf ecn-enabled {
                  type boolean;
                  default "false";
                  description
                    "ECN is enabled on the queue.";
                }
                description
                  "Random Early Detect (RED) configuration.";
              }
            }

            case wred {
              container wred {
                list wred {
                  key "profile";

                  leaf profile {
                    type uint8;
                    description
```

```
                    "Profile id of each WRED profile.";
                }

                leaf color-type {
                  type identityref {
                    base qos-types:wred-color-type;
                  }
                  default qos-types:wred-color-dscp;
                  description
                    "WRED color-type of each profile.";
                }

                list color-val {
                  key "min max";

                  leaf min {
                    type uint8;
                    description
                      "Minimum value of color types.";
                  }

                  leaf max {
                    type uint8;
                    description
                      "Maximum value of color types.";
                  }
                  description
                    "List of color markings which constitute
                     a traffic profile.";
                }
                uses red-config-parameters;
                description
                  "List of RED profiles each with its own
                   threshold values.";
              }

              leaf ecn-enabled {
                type boolean;
                default "false";
                description
                  "When configure as true, ECN is enabled on
                   the queue.";
              }
              description
                "Weighted Random Early Detect (WRED) configuration.";
            }
          }
```

```
            case codel {
              container codel {
                uses codel-config-parameters;
                leaf ecn-enabled {
                  type boolean;
                  default "false";
                  description
                    "When configure as true, ECN is enabled on
                     the queue.";
                }
                description
                  "CoDel configuration.";
              }
            }

            case fq-codel {
              container fq-codel {
                uses codel-config-parameters;
                leaf flows {
                  type uint64;
                  default 1024;
                  description
                    "The maximum number of flow queues.";
                }
                leaf ecn-enabled {
                  type boolean;
                  default "false";
                  description
                    "ECN is enabled on the queue.";
                }
                description
                  "FQ-CoDel configuration.";
              }
            }

            description
              "Choice of Drop Algorithm.";
          }
          description
            "Algorithmic Drop configuration container.";
        }
        description
          "Queue configuration container.";
      }
      description
        "Queue grouping.";
    }
```

```
    container queues {
      description
        "List of queue templates.";

      list queue {
        key "name";
        description
          "Queue entry template.";

        leaf name {
          type string;
          description
            "A name identifying this queue template";
        }
        uses queue;
      }
    }

    grouping queue-reference {
      container queue-reference {
        leaf queue-name {
          type leafref  {
            path "/qos-action:queues/" +
                 "qos-action:queue/qos-action:name";
          }
          mandatory true;
          description
            "This leaf defines name of the  queue template
             referenced.";
        }
        description
          "Queue template reference.";
      }
      description
        "Queue template reference grouping.";
    }

    grouping scheduler {
      container scheduler {
        uses min-rate;
        uses max-rate;
        description
          "Scheduler configuration container.";
      }
      description
        "Scheduler configuration grouping.";
    }
  }
```

<CODE ENDS>

Figure 5: ietf-qos-actions module

4.3. Diffserv Module

Diffserv module augments Traffic Policy module to enable diffrentiated services for different types of packets.

The figure below presents the tree diagram for the ietf-diffserv module. This module augments the base ietf-traffic-policy module to enable Differentiated Services (Diffserv) for various packet types. It adds several filter parameters to the traffic policy's classifiers, as well as, various action parameters that can be applied within a policy's classifier.

```
module: ietf-diffserv

  augment /traffic-policy:classifiers/traffic-policy:classifier
            /traffic-policy:filter:
    +--rw (filter-param)?
       +--:(dscp)
       |  +--rw dscp* [min max]
       |        ...
       +--:(source-ipv4-prefix)
       |  +--rw source-ipv4-prefix*        inet:ipv4-prefix
       +--:(destination-ipv4-prefix)
       |  +--rw destination-ipv4-prefix*   inet:ipv4-prefix
       +--:(source-ipv6-prefix)
       |  +--rw source-ipv6-prefix*        inet:ipv6-prefix
       +--:(destination-ipv6-prefix)
       |  +--rw destination-ipv6-prefix*   inet:ipv6-prefix
       +--:(source-port)
       |  +--rw source-port* [min max]
       |        ...
       +--:(destination-port)
       |  +--rw destination-port* [min max]
       |        ...
       +--:(protocol)
       |  +--rw protocol* [min max]
       |        ...
       +--:(traffic-group)
       |  +--rw traffic-group
       |        ...
       +--:(filter-match-all)
          +--rw match-all
                ...
  augment /traffic-policy:policies/traffic-policy:policy
```

```
              /traffic-policy:classifier/traffic-policy:inline
              /traffic-policy:filter:
       +--rw (filter-params)?
          +--:(dscp)
          |  +--rw dscp* [min max]
          |        ...
          +--:(source-ipv4-prefix)
          |  +--rw source-ipv4-prefix*        inet:ipv4-prefix
          +--:(destination-ipv4-prefix)
          |  +--rw destination-ipv4-prefix*   inet:ipv4-prefix
          +--:(source-ipv6-prefix)
          |  +--rw source-ipv6-prefix*        inet:ipv6-prefix
          +--:(destination-ipv6-prefix)
          |  +--rw destination-ipv6-prefix*   inet:ipv6-prefix
          +--:(source-port)
          |  +--rw source-port* [min max]
          |        ...
          +--:(destination-port)
          |  +--rw destination-port* [min max]
          |        ...
          +--:(protocol)
          |  +--rw protocol* [min max]
          |        ...
          +--:(traffic-group)
          |  +--rw traffic-group
          |        ...
          +--:(filter-match-all)
             +--rw match-all
                   ...
     augment /traffic-policy:policies/traffic-policy:policy
              /traffic-policy:classifier/traffic-policy:action:
       +--rw (action-params)?
          +--:(dscp-marking)
          |  +--rw dscp
          |        ...
          +--:(meter-inline)
          |  +--rw (meter-types)?
          |        ...
          +--:(meter-reference)
          |  +--rw meter
          |        ...
          +--:(traffic-group-marking)
          |  +--rw traffic-group
          |        ...
          +--:(child-policy) {qos-types:child-policy}?
          |  +--rw child-policy {qos-types:child-policy}?
          |        ...
          +--:(qos-count) {qos-types:qos-count}?
```

```
          |  +--rw qos-count {qos-types:qos-count}?
          |        ...
          +--:(named-qos-count) {qos-types:named-qos-count}?
          |  +--rw named-qos-count {qos-types:named-qos-count}?
          |        ...
          +--:(queue-inline)
          |  +--rw queue
          |        ...
          +--:(scheduler-inline)
             +--rw scheduler
                   ...
```

Figure 4: ietf-diffserv abridged tree diagram

```
<CODE BEGINS> file "ietf-diffserv@2026-03-02.yang"
module ietf-diffserv {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-diffserv";
  prefix diffserv;

  import ietf-traffic-policy {
    prefix traffic-policy;
  }
  import ietf-qos-action {
    prefix qos-action;
  }
  import ietf-inet-types {
    prefix inet;
  }
  import ietf-qos-types {
    prefix qos-types;
    reference
      "RFC XXXX: YANG Models for Quality of Service (QoS).";
  }

  organization
    "IETF Routing Area Working Group";

  contact
    "WG Web:   <https://datatracker.ietf.org/wg/rtgwg/>
     WG List:  <mailto:rtgwg@ietf.org>

     Editor:   Aseem Choudhary
               <mailto:asechoud@cisco.com>
     Editor:   Mahesh Jethanandani
               <mailto:mjethanandani@gmail.com>";
  description
    "This module contains a collection of YANG definitions for
```

```
        configuring Diffserv specification implementations.

        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 2026-03-02 {
      description
        "Initial version.";
      reference
        "RFC XXXX: A YANG Model for Quality of Service (QoS).";
    }

    grouping dscp {
      list dscp {
        key "min max";
        description
          "List of DSCP ranges.";
        leaf min {
          type inet:dscp;
          description
            "Minimum value of DSCP min-max range.";
        }
        leaf max {
          type inet:dscp;
          must ". >= ../min" {
            error-message
            "max must be greater than or equal to min.";
          }
          description
            "Maximum value of DSCP min-max range.";
        }
      }
      description
        "Filter grouping containing list of DSCP ranges.";
    }

    grouping source-ipv4-prefix {
      leaf-list source-ipv4-prefix {
```

```
        type inet:ipv4-prefix;
        description
          "List of source IPv4 prefixes.";
      }
      description
        "Filter grouping containing list of source IPv4 prefixes.";
    }

    grouping destination-ipv4-prefix {
      leaf-list destination-ipv4-prefix {
        type inet:ipv4-prefix;
        description
          "List of destination IPv4 prefixes.";
      }
      description
        "Filter grouping containing list of destination IPv4
         prefixes.";
    }

    grouping source-ipv6-prefix {
      leaf-list source-ipv6-prefix {
        type inet:ipv6-prefix;
        description
          "List of source IPv6 prefixes.";
      }
      description
        "Filter grouping containing list of source IPv6
         prefixes.";
    }

    grouping destination-ipv6-prefix {
      leaf-list destination-ipv6-prefix {
        type inet:ipv6-prefix;
        description
          "List of destination IPv6 prefixes.";
      }
      description
        "Filter grouping containing list of destination IPv6
         prefixes.";
    }

    grouping source-port {
      list source-port {
        key "min max";
        description
          "List of ranges of source port.";

        leaf min {
```

```
          type inet:port-number;
          description
            "Minimum value of source port range.";
        }

        leaf max {
          type inet:port-number;
          must ". >= ../min" {
            error-message
            "max must be greater than or equal to min.";
          }
          description
            "Maximum value of source port range.";
        }
      }
      description
        "Filter grouping containing list of source port ranges.";
    }

    grouping destination-port {
      list destination-port {
        key "min max";
        description
          "List of min-max range of destination ports.";
        leaf min {
          type inet:port-number;
          description
            "Minimum value of destination port.";
        }
        leaf max {
          type inet:port-number;
          must ". >= ../min" {
            error-message
            "max must be greater than or equal to min.";
          }
          description
            "Maximum value of destination port range.";
        }
      }
      description
        "Filter grouping containing list of destination port ranges.";
    }

    grouping protocol {
      list protocol {
        key "min max";
        description
```

```
          "List of ranges of protocol values. Protocol refers to the
           value in the protocol field of the IPv4 header and value
           in the 'next-header' field of IPv6 header. In IPv6 header,
           'next-header' field indicates first extension header or the
            protocol in the 'upper-layer' header.";
        reference
          "RFC 791: Internet Protocol
           RFC 8200: Internet Protocol, Version 6 (IPv6)
                     Specification.";

        leaf min {
          type inet:protocol-number;
          description
            "Minimum value of protocol range.";
        }

        leaf max {
          type inet:protocol-number;
          must ". >= ../min" {
            error-message
            "max must be greater than or equal to min.";
          }
          description
            "Maximum value of protocol range.";
        }
      }
      description
        "Filter grouping containing list of protocol ranges.";
    }

    grouping traffic-group {
      container traffic-group {
        leaf name {
          type string ;
          description
            "This leaf defines name of the traffic group referenced.";
        }
      description
        "Traffic group container.";
      }
      description
        "Traffic group grouping.";
    }

    grouping filter-match-all {
      container match-all {
        leaf action {
          type empty;
```

```
          description
            "When set, the filter matches all packets.";
        }
        description
          "The match all action.";
      }
      description
        "The match all filter grouping.";
    }

    augment "/traffic-policy:classifiers/traffic-policy:classifier" +
            "/traffic-policy:filter" {
      choice filter-param {
        description
          "Choice of filter types.";
        case dscp {
          uses dscp;
          description
            "Filter containing list of DSCP ranges.";
        }

        case source-ipv4-prefix {
          uses source-ipv4-prefix;
          description
            "Filter containing list of source IPv4 prefixes.";
        }

        case destination-ipv4-prefix {
          uses destination-ipv4-prefix;
          description
            "Filter containing list of destination IPv4 prefix.";
        }

        case source-ipv6-prefix {
          uses source-ipv6-prefix;
          description
            "Filter containing list of source IPv6 prefixes.";
        }

        case destination-ipv6-prefix {
          uses destination-ipv6-prefix;
          description
            "Filter containing list of destination IPv6 prefix.";
        }

        case source-port {
          uses source-port;
          description
```

```
            "Filter containing list of source-port ranges.";
        }

        case destination-port {
          uses destination-port;
          description
            "Filter containing list of destination-port ranges.";
        }

        case protocol {
          uses protocol;
          description
            "Filter Type Protocol.";
        }

        case traffic-group {
          uses traffic-group;
          description
            "Filter Type traffic-group.";
        }

        case filter-match-all {
          uses filter-match-all;
          description
            "When the filter type is set to match all packets.";
        }
      }
      description
        "Augments QoS classifier to add in Diffserv filters.";
    }

    augment "/traffic-policy:policies/traffic-policy:policy" +
            "/traffic-policy:classifier/traffic-policy:inline" +
            "/traffic-policy:filter" {
      when "derived-from-or-self(../../../traffic-policy:type, " +
           "'qos-types:ipv4-diffserv-policy-type') or " +
           "derived-from-or-self(../../../traffic-policy:type, " +
           "'qos-types:ipv6-diffserv-policy-type') or " +
           "derived-from-or-self(../../../traffic-policy:type, " +
           "'qos-types:diffserv-policy-type')" {
        description
          "If policy type is v4, v6 or default Diffserv,
           this filter can be used.";
      }
      choice filter-params {
        description
          "Choice of action types.";
        case dscp {
```

```
          uses dscp;
          description
            "Filter containing list of DSCP ranges.";
        }

        case source-ipv4-prefix {
          when "derived-from-or-self(/traffic-policy:policies/" +
               "traffic-policy:policy/traffic-policy:type, " +
               "'qos-types:ipv6-diffserv-policy-type')" {
              description
                 "If policy type is v6, this filter cannot be used.";
          }
          uses source-ipv4-prefix;
          description
            "Filter containing list of source IPv4 prefixes.";
        }

        case destination-ipv4-prefix {
          when "derived-from-or-self(/traffic-policy:policies" +
               "/traffic-policy:policy/traffic-policy:type, " +
               "'qos-types:ipv6-diffserv-policy-type')" {
              description
                 "If policy type is v6, this filter cannot be used.";
          }
          uses destination-ipv4-prefix;
          description
            "Filter containing list of destination IPv4 prefix.";
        }

        case source-ipv6-prefix {
          when "derived-from-or-self(/traffic-policy:policies" +
               "/traffic-policy:policy/traffic-policy:type, "+
               "'qos-types:ipv4-diffserv-policy-type')" {
              description
                 "If policy type is v4, this filter cannot be used.";
          }
          uses source-ipv6-prefix;
          description
            "Filter containing list of source IPv6 prefixes.";
        }

        case destination-ipv6-prefix {
          when "derived-from-or-self(/traffic-policy:policies" +
               "/traffic-policy:policy/traffic-policy:type, " +
               "'qos-types:ipv4-diffserv-policy-type')" {
              description
                 "If policy type is v4, this filter cannot be used.";
          }
```

```
          uses destination-ipv6-prefix;
          description
            "Filter containing list of destination IPv6 prefix.";
        }

        case source-port {
          uses source-port;
          description
            "Filter containing list of source-port ranges.";
        }

        case destination-port {
          uses destination-port;
          description
            "Filter containing list of destination-port ranges.";
        }

        case protocol {
          uses protocol;
          description
            "Filter Type Protocol.";
        }

        case traffic-group {
          uses traffic-group;
          description
            "Filter Type traffic-group.";
        }

        case filter-match-all {
          uses filter-match-all;
          description
            "When the filter type is set to match all packets.";
        }
      }
      description
        "Augments Diffserv Classifier with common filter types.";
    }

    augment "/traffic-policy:policies/traffic-policy:policy/" +
            "traffic-policy:classifier/traffic-policy:action" {
      when "derived-from-or-self(../../traffic-policy:type, " +
           "'qos-types:ipv4-diffserv-policy-type') or " +
           "derived-from-or-self(../../traffic-policy:type, " +
           "'qos-types:ipv6-diffserv-policy-type') or " +
           "derived-from-or-self(../../traffic-policy:type, " +
           "'qos-types:diffserv-policy-type')" {
        description
```

```
          "If policy type is v4, v6 or default Diffserv,
           these actions can be used.";
      }
      description
        "Augments Diffserv Policy with action configuration.";

      choice action-params {
        description
          "Choice of different action parameters that can be taken.";

        case dscp-marking {
          uses qos-action:dscp-marking;
        }

        case meter-inline {
          uses qos-action:meter;
        }

        case meter-reference {
          uses qos-action:meter-reference;
        }

        case traffic-group-marking {
          uses qos-action:traffic-group-marking;
        }

        case child-policy {
          if-feature qos-types:child-policy;
          uses qos-action:child-policy;
        }

        case qos-count {
          if-feature qos-types:qos-count;
          uses qos-action:qos-count;
        }

        case named-qos-count {
          if-feature qos-types:named-qos-count;
          uses qos-action:named-qos-count;
        }

        case queue-inline {
          uses qos-action:queue;
        }

        case scheduler-inline {
          uses qos-action:scheduler;
        }
```

```
      }
    }
  }
<CODE ENDS>
```

Figure 7: ietf-diffserv module

4.4. QoS Operational Module

QoS Operational module contains all operational statistics. It contains statistics related to classifier, metering, queueing.

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

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

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

In addition, Named statistics is defined as statistics which is 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 counts are maintained separately.

A clear action is provided to clear all statistics or statistics of a particular kind.

The figure below shows the tree structure for the ietf-qos-oper module. This module is dedicated to providing visibility into the real-time performance of QoS policies. It contains operational state data and statistics, organized by classifier, meter, and queue. This includes ’named’ counters, which allow operators to track traffic statistics across selected classifiers, meters and queues.

```
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* []
     |  |     ...
     |  +--ro named* []
     |  |     ...
     |  +--ro metering* []
     |  |     ...
     |  +--ro queueing* []
     |        ...
     +---x clear
        +---w input
        |     ...
        +--ro output
              ...
```

Figure 5: ietf-qos-oper abridged tree diagram

```
<CODE BEGINS> file "ietf-qos-oper@2026-03-02.yang"
module ietf-qos-oper {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-qos-oper";
  prefix qos-oper;

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 9911:Common YANG Data Types.";
  }
  import ietf-interfaces {
    prefix if;
    reference
      "RFC8343: A YANG Data Model for Interface Management.";
  }
  import ietf-qos-types {
    prefix qos-types;
    reference
      "RFC XXXX: YANG Models for Quality of Service (QoS).";
  }

  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>
        Editor:   Mahesh Jethanandani
                <mailto:mjethanandani@gmail.com>";
    description
      "This module contains a collection of YANG definitions for
       QoS operational specification.

       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 2026-03-02 {
      description
        "Initial version.";
      reference
        "RFC XXXX: YANG Models for Quality of Service (QoS).";
    }

    grouping classifier-stats {
      description
        "Statistics for a classifier.";

      leaf name {
        type string;
        description
          "A identifier for Classifier name.";
      }

      leaf classified-pkts {
        type yang:zero-based-counter64;
        description
          "Number of total packets filtered for a classifier.";
      }

      leaf classified-bytes {

        type yang:zero-based-counter64;
        description
          "Number of total bytes filtered for a classifier.";
```

```
      }

      leaf classified-rate {
        type yang:gauge64;
        units "bits-per-second";
        description
          "Rate of average data flow through a classifier.";
      }
    }

    grouping queuing-stats {
      description
        "Statistics for a queue.";

      leaf queue-id {
        type string;
        description
          "A identifier for the Queue.";
      }

      leaf transmit-pkts {
        type yang:zero-based-counter64;
        description
          "Number of packets transmitted from queue.";
      }

      leaf transmit-bytes {
        type yang:zero-based-counter64;
        description
          "Number of bytes transmitted from queue.";
      }

      leaf queue-current-size-bytes {
        type yang:gauge64;
        description
          "Number of bytes currently buffered.";
      }

      leaf queue-average-size-bytes {
        type yang:gauge64;
        description
          "Average queue size in number of bytes.";
      }

      leaf queue-peak-size-bytes {

        type yang:gauge64;
        description
```

```
          "Peak buffer queue size in bytes.";
      }

      leaf tail-drop-pkts {
        type yang:zero-based-counter64;
        description
          "Total number of packets tail-dropped.";
      }

      leaf tail-drop-bytes {
        type yang:zero-based-counter64;
        description
          "Total number of bytes tail-dropped.";
      }

      container red-statistics {
        leaf drop-pkts {
          type yang:zero-based-counter64;
          description
            "Total number of packets dropped because of RED.";
        }

        leaf drop-bytes {
          type yang:zero-based-counter64;
          description
            "Total number of bytes dropped because of RED.";
        }

        leaf ecn-marked-pkts {
          type yang:zero-based-counter64;
          description
            "Total number of packets that were marked with ECN
             because of RED.";
        }

        leaf ecn-marked-bytes {
          type yang:zero-based-counter64;
          description
            "Total number of bytes that were marked with ECN because of
             RED.";
        }

        list wred-stats {
          config false;
          description
            "QoS WRED statistics.";

          leaf profile {
```

```
            type uint8;
            description
              "Profile identifier for each color of traffic.";
          }

          leaf drop-pkts {
            type yang:zero-based-counter64;
            description
              "Total number of packets dropped because of WRED.";
          }

          leaf drop-bytes {
            type yang:zero-based-counter64;
            description
              "Total number of bytes dropped because of WRED.";
          }

          leaf ecn-marked-pkts {
            type yang:zero-based-counter64;
            description
              "Total number of packets that were marked with ECN
               because of WRED.";
          }

          leaf ecn-marked-bytes {
            type yang:zero-based-counter64;
            description
              "Total number of bytes that were marked with ECN because
               of WRED.";
          }
        }
        description
          "Container of all RED statistics.";
      }

      container codel-statistics {
        leaf drop-pkts {
          type yang:zero-based-counter64;
          description
            "Total number of packets dropped because of CoDel
             or FQ-CoDel.";
        }

        leaf drop-bytes {
          type yang:zero-based-counter64;
          description
            "Total number of bytes dropped because of CoDel
             or FQ-CoDel.";
```

```
        }

        leaf ecn-marked-pkts {
          type yang:zero-based-counter64;
          description
            "Total number of packets that were marked with ECN
             because of CoDel or FQ-CoDel.";
        }

        leaf ecn-marked-bytes {
          type yang:zero-based-counter64;
          description
            "Total number of bytes that were marked with ECN because
             of CoDel or FQ-CoDel.";
        }

        leaf new-flow-count {
          type yang:zero-based-counter64;
          description
            "Total number of flows that FQ-CoDel has identified and is
             managing or has managed.";
        }
        description
          "Container of all CoDel or FQ-CoDel statistics.";
      }
    }

    grouping metering-stats {
      description
        "Statistics for a meter.";

      leaf meter-id {
        type string;
        description
          "A identifier that identifies this set of metering
           statistics.";
      }

      leaf conform-pkts {
        type yang:zero-based-counter64;
        description
          "Number of packets that conform to the rate and burst.";
      }

      leaf conform-bytes {
        type yang:zero-based-counter64;
        description
```

```
          "Total bytes count that conform to the rate and burst.";
        }

        leaf conform-rate {
          type yang:gauge64;
          units "bits-per-second";
          description
            "Traffic Rate measured as conformimg.";
        }

        leaf exceed-pkts {
          type yang:zero-based-counter64;
          description
            "Number of packets that exceed the peak rate and peak
             burst.";
        }

        leaf exceed-bytes {
          type yang:zero-based-counter64;
          description
            "Total number of bytes which exceed the peak rate and peak
             burst.";
        }

        leaf exceed-rate {
          type yang:gauge64;
          units "bits-per-second";
          description
            "Traffic Rate measured as exceeding.";
        }

        leaf violate-pkts {
          type yang:zero-based-counter64;
          description
            "Number of packets that were beyond peak rate
             and peak burst.";
        }

        leaf violate-bytes {
          type yang:zero-based-counter64;
          description
            "Total number of bytes which were beyond peak rate and
             peak burst.";
        }

        leaf violate-rate {
          type yang:gauge64;
          units "bits-per-second";
```

```
        description
          "Traffic Rate measured as violating.";
      }

      leaf drop-pkts {
        type yang:zero-based-counter64;
        description
          "Number of packets dropped by meter.";
      }

      leaf drop-bytes {
        type yang:zero-based-counter64;
        description
          "Number of bytes dropped by meter.";
      }
    }

    augment "/if:interfaces/if:interface" {
      description
        "Augments interface module to add QoS Target entry.";

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

        list stats-per-direction {
          description
            "QoS interface statistics in ingress or egress direction.";

          leaf direction {
            type identityref {
              base qos-types:direction;
            }
            description
              "Direction of the traffic flow. Either ingress or
               egress.";
          }

          leaf policy-name {
            type string;
            description
              "Policy 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 {
              description
                "Statistics for each Classifier in a
                 Policy applied in a particular direction.";
              uses classifier-stats;
            }

            list named {
              description
                "Statistics for a statistics-name.";

              leaf name {
                type string;
                description
                  "A name for statistics.";
              }

              container aggregated {
                description
                  "Matched aggregated statistics for a statistics-name.";

                leaf pkts {
                  type yang:zero-based-counter64;
                  description
                    "Number of total matched packets associated
                     with a statistics name.";
                }

                leaf bytes {
                  type yang:zero-based-counter64;
                  description
                    "Number of total matched bytes associated
                     with a statistics name.";
                }

                leaf rate {
                  type yang:gauge64;
                  units "bits-per-second";
                  description
                    "Rate of average matched data which is associated
                     to a statistics name.";
                }
              }

              container non-aggregated {
                description
                  "Statistics for non-aggregated statistics-name.";
                list classifier {
```

```
                description
                  "Statistics for each Classifier in a
                   Policy applied in a particular direction.";
                uses classifier-stats;
              }

              list metering {
                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 {
                description
                  "Statistics for each Queue associated with
                   the Policy.";
                uses queuing-stats;
              }
            }
          }

          list metering {
            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 {
            description
              "Statistics for each Queue associated with the Policy.";
            uses queuing-stats;
          }
        }

        action clear {
          description
            "Clear statistics action command. Execution of this command
             should result in all the QoS interface counters to be
             cleared and set to 0.";

          input {
```

```
            leaf category {
              type identityref {
                base qos-types:clear-counters-type;
              }
              description
                "Category of counters which is being cleared.";
            }

            leaf started-at {
              type yang:date-and-time;
              description
                "The date and time when the counters should be
                 cleared.";
            }
          }

          output {
            leaf finished-at {
              type yang:date-and-time;
              description
                "The date and time when the counters finished
                 clearing.";
            }
          }
        }
      }
    }
  }
  <CODE ENDS>
```

Figure 9: ietf-qos-oper module

4.5. Scheduler Policy Module

Scheduling Policy module defines a set of scheduling parameters and associates a Queue Policy with it.

module: ietf-scheduler-policy

```
  augment /traffic-policy:policies/traffic-policy:policy
            /traffic-policy:classifier/traffic-policy:inline
            /traffic-policy:filter:
    +--rw (filter-params)?
       +--:(filter-match-all)
          +--rw match-all
                  ...
  augment /traffic-policy:policies/traffic-policy:policy
            /traffic-policy:classifier/traffic-policy:action:
```

```
       +--rw (action-params)?
          +--:(scheduler)
          |  +--rw scheduler
          |        ...
          +--:(queue-policy-name)
             +--rw queue-policy-name
                   ...
```

Figure 10: ietf-scheduler-policy abridged abridged tree diagram

```
<CODE BEGINS>
file "ietf-scheduler-policy@2026-03-02.yang"
module ietf-scheduler-policy {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-scheduler-policy";
  prefix scheduler-policy;

  import ietf-traffic-policy {
    prefix traffic-policy;
  }
  import ietf-qos-action {
    prefix qos-action;
  }
  import ietf-diffserv {
    prefix diffserv;
  }
  import ietf-qos-types {
    prefix qos-types;
    reference
      "RFC XXXX: YANG Models for Quality of Service (QoS).";
  }

  organization
    "IETF Routing Area Working Group.";

  contact
    "WG Web:   <https://datatracker.ietf.org/wg/rtgwg/>
     WG List:  <mailto:rtgwg@ietf.org>

     Editor:   Aseem Choudhary
               <mailto:asechoud@cisco.com>
     Editor:   Mahesh Jethanandani
               <mailto:mjethanandani@gmail.com>";

  description
    "This module contains a collection of YANG definitions for
     configuring QoS scheduler policies.
```

```
        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 2026-03-02 {
        description
          "Initial version.";
        reference
          "RFC XXXX: YANG Models for Quality of Service (QoS).";
      }

      augment "/traffic-policy:policies/traffic-policy:policy" +
              "/traffic-policy:classifier/traffic-policy:inline" +
              "/traffic-policy:filter" {
        when "derived-from-or-self(../../../traffic-policy:type, " +
             "'qos-types:scheduler-policy-type')" {
          description
            "This filter can be used only when policy type is
             scheduler-policy.";
        }
        choice filter-params {
          description
            "Choice of filter parameters.";
          case filter-match-all {
            uses diffserv:filter-match-all;
            description
              "A filter that matches all packets.";
          }

        }
        description
          "Augments Queue Policy Classifier to add filter parameters.";
      }

      augment "/traffic-policy:policies/traffic-policy:policy/" +
              "traffic-policy:classifier/traffic-policy:action" {
        when "derived-from-or-self(../../traffic-policy:type, " +
             "'qos-types:scheduler-policy-type')" {
          description
```

```
            "These actions apply only when policy type is
             scheduler-policy.";
        }

        choice action-params {
          description
            "Choice of action parameters that can be taken.";

          case scheduler {
            uses qos-action:scheduler;
          }

          case queue-policy-name {
            container queue-policy-name {
              leaf queue-policy {
                type leafref  {
                  path "/traffic-policy:policies/" +
                       "traffic-policy:policy/traffic-policy:name";
                }
                must "derived-from-or-self(/traffic-policy:policies" +
                     "/traffic-policy:policy/traffic-policy:type," +
                     "'qos-types:queue-policy-type')";
                mandatory true;
                description
                  "This leafref refers to Queue Policy name.";
              }
              description
                "Container for Queue Policy name.";
            }
          }
        }
        description
          "Augments Queue Policy Classifier to add in Action parameters
           for Scheduler policy.";
      }
    }
    <CODE ENDS>
```

Figure 11: ietf-scheduler-policy module

4.6. Queue Policy Module

Queue Policy module augments the Traffic Policy module. It augments to allow queues to be referred to in a policy. The match is based on Traffic Group and action parameters which are defined in QoS Action module.

The figure below provides the tree diagram for the ietf-queue-policy module. This module describes the mechanisms used for managing congestion. It includes configurations for queue buffers, scheduling algorithms (such as Strict Priority or Weighted Round Robin), and drop policies (such as Tail Drop or Weighted Random Early Detection). The model supports nested scheduling structures to accommodate complex hierarchical QoS requirements.

```
module: ietf-queue-policy

  augment /traffic-policy:policies/traffic-policy:policy
            /traffic-policy:classifier/traffic-policy:inline
            /traffic-policy:filter:
    +--rw (filter-params)?
       +--:(traffic-group)
       |  +--rw traffic-group
       |        ...
       +--:(filter-match-all)
          +--rw match-all
                ...
  augment /traffic-policy:policies/traffic-policy:policy
            /traffic-policy:classifier/traffic-policy:action:
    +--rw (action-params)?
       +--:(queue-template-name)
       |  +--rw queue-reference
       |        ...
       +--:(queue-inline)
          +--rw queue
                ...
```

Figure 6: ietf-queue-policy abridged tree diagram

```
<CODE BEGINS> file "ietf-queue-policy@2026-03-02.yang"
module ietf-queue-policy {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-queue-policy";
  prefix queue-policy;

  import ietf-traffic-policy {
    prefix traffic-policy;
  }
  import ietf-qos-action {
    prefix qos-action;
  }
  import ietf-diffserv {
    prefix diffserv;
  }
  import ietf-qos-types {
```

```
      prefix qos-types;
      reference
        "RFC XXXX: YANG Models for Quality of Service (QoS).";
    }

    organization
      "IETF Routing Area Working Group.";

    contact
      "WG Web:   <https://datatracker.ietf.org/wg/rtgwg/>
       WG List:  <mailto:rtgwg@ietf.org>

       Editor:   Aseem Choudhary
                 <mailto:achoudhary@aviatrix.com>
       Editor:   Mahesh Jethanandani
                 <mailto:mjethanandani@gmail.com>";

    description
      "This module contains a collection of YANG definitions for
       QoS Queue policies.

       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 2026-03-02 {
      description
        "Initial version.";
      reference

        "RFC XXXX: YANG Models for Quality of Service (QoS).";
    }

    augment "/traffic-policy:policies/traffic-policy:policy" +
            "/traffic-policy:classifier/traffic-policy:inline" +
            "/traffic-policy:filter" {
      when "derived-from-or-self(../../../traffic-policy:type, " +
           "'qos-types:queue-policy-type')" {
        description
```

```
            "If policy type is Queue Policy, this filter can be used.";
        }

        choice filter-params {
          description
            "Choice of Action types.";
          case traffic-group {
            uses diffserv:traffic-group;
            description
              "Traffic group name.";
          }
          case filter-match-all {
            uses diffserv:filter-match-all;
            description
              "When the filter type is set to match all packets.";
          }
        }
        description
          "Augments Queue Policy to add filter types.";
      }

      augment "/traffic-policy:policies/traffic-policy:policy" +
              "/traffic-policy:classifier/traffic-policy:action" {
        when "derived-from-or-self(../../traffic-policy:type,"  +
             "'qos-types:queue-policy-type')" {
          description
            "Queue Policy actions.";
        }

        choice action-params {
          description
            "Choice of different action parameters that can be taken.";

          case queue-template-name {
            uses qos-action:queue-reference;
          }

          case queue-inline {
            uses qos-action:queue;

          }
        }
        description
          "Augments Queue Policy action to add Queue template reference
           or Queue inline.";
      }
    }
    <CODE ENDS>
```

Figure 13: ietf-queue-policy module

4.7. IETF QoS Types Module

The QoS Types module contains all the type definitions used by the other QoS modules.

```
<CODE BEGINS> file "ietf-qos-types@2026-03-02.yang"
module ietf-qos-types {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-qos-types";
  prefix qos-types;

  organization
    "IETF Routing Area Working Group";

  contact
    "WG Web:   <https://datatracker.ietf.org/wg/rtgwg/>
     WG List:  <mailto:rtgwg@ietf.org>

     Editor:   Aseem Choudhary
               <mailto:asechoud@cisco.com>
     Editor:   Mahesh Jethanandani
               <mailto:mjethanandani@gmail.com>";

  description
    "This module contains type definitions for all QoS types.

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

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject
     to the license terms contained in, the 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 2026-03-02 {
    description
      "Initial version";
    reference
      "RFC XXXX: YANG Models for Quality of Service (QoS).";
  }
```

```
     /*
      * Features.
      */
     feature child-policy {
       description
         " This feature allows configuration of hierarchical policy.";
     }
     feature qos-count {
       description
         "This feature allows action configuration to enable
          counter in a classifier";
     }
     feature named-qos-count {
       description
         "This feature allows action configuration to enable
          named counter in a classifier";
     }

     /*
      * Identities.
      */
     identity policy-type {
       description
         "Base identity for policy type.";
     }
     identity diffserv-policy-type {
       base policy-type;
       description
         "Policy type defined as Diffserv.";
     }
     identity ipv4-diffserv-policy-type {
       base policy-type;
       description
         "Policy type defined as a Diffserv IPv4 policy type.";
     }
     identity ipv6-diffserv-policy-type {
       base policy-type;
       description

         "Policy type defined as a Diffserv IPv6 policy type.";
     }
     identity queue-policy-type {
       base policy-type;
       description
         "Policy type defined as a queue policy type.";
     }
     identity scheduler-policy-type {
       base policy-type;
```

```
      description
        "Policy type defined as a scheduler policy type.";
    }

    identity action-type {
      description
        "Base identity for the action type.";
    }
    identity dscp-marking {
      base action-type;
      description
        "Action type defined as DSCP marking.";
    }
    identity meter-inline {
      base action-type;
      description
        "Action type defined as meter inline.";
    }
    identity meter-reference {
      base action-type;
      description
        "Action type defined as meter reference.";
    }
    identity queue {
      base action-type;
      description
        "Action type defined as queue.";
    }
    identity scheduler {
      base action-type;
      description
        "Action type defined as scheduler.";
    }
    identity discard {
      base action-type;
      description
        "Action type defined as discard.";
    }
    identity child-policy {

      if-feature child-policy;
      base action-type;
      description
        "Action type defined as child policy.";
    }
    identity qos-count {
      if-feature qos-count;
      base action-type;
```

```
      description
        "Action type defined as qos count.";
    }
    identity named-qos-count {
      if-feature named-qos-count;
      base action-type;
      description
        "Action type specified as a named qos counter.";
    }
    identity queue-policy-name {
      base action-type;
      description
        "Action type specified as a queue policy name.";
    }

    identity meter-type {
      description
        "Base identity for types of meter.";
    }
    identity one-rate-two-color {
      base meter-type;
      description
        "Meter type specified as one rate two color meter.";
    }
    identity one-rate-tri-color {
      base meter-type;
      description
        "Meter type specified as one rate three color meter.";
    }
    identity two-rate-tri-color {
      base meter-type;
      description
        "Meter type specified as two rate three color meter.";
    }

    identity drop-type {
      description
        "Base identity for drop algorithm.";
    }
    identity tail-drop {

      base drop-type;
      description
        "Drop algorithm specified as tail drop.";
    }
    identity red {
      base drop-type;
      description
```

```
        "Drop algorithm specified as RED.";
    }
    identity wred {
      base drop-type;
      description
        "Drop algorithm specified as WRED.";
    }

    identity rate-unit-type {
      description
        "Base definition for the type of rate unit.";
    }
    identity bits-per-second {
      base rate-unit-type;
      description
        "Rate specified in bits per second.";
    }
    identity percent {
      base rate-unit-type;
      description
        "Rate specified as a percentage of queue depth.";
    }

    identity burst-unit-type {
      description
        "Base definition for burst specification.";
    }
    identity bytes {
      base burst-unit-type;
      description
        "Burst specified in bytes.";
    }
    identity microsecond {
      base burst-unit-type;
      description
        "Burst specified in microseconds.";
    }

    identity red-threshold-unit {
      description
        "Base definition for RED threshold specification.";
    }
    identity red-threshold-bytes {
      base red-threshold-unit;
      description
        "RED threshold specified in bytes.";
    }
    identity red-threshold-us {
```

```
      base red-threshold-unit;
      description
        "RED threshold specified in microseconds.";
    }
    identity red-threshold-pc {
      base red-threshold-unit;
      description
        "RED threshold specified as a percentage.";
    }
    identity red-theshold-pt {
      base red-threshold-unit;
      description
        "RED threshold specified per-thousand.";
    }

    identity wred-color-type {
      description
        "Base identity for WRED color type.";
    }
    identity wred-color-dscp {
      base wred-color-type;
      description
        "WRED color specified by DSCP.";
    }

    identity probability-unit {
      description
        "Base definition for probability unit.";
    }
    identity probability-pc {
      base probability-unit;
      description
        "Probability specified in percentage.";
    }
    identity probability-pt {
      base probability-unit;
      description
        "Probability specified in unit of per thousand.";
    }
    identity probability-denominator {
      base probability-unit;

      description
        "Probability value specified as a denominator value
         while numerator is always 1.";
    }

    identity filter-type {
```

```
      description
        "Base identity for filter type.";
    }

    identity dscp {
      base filter-type;
      description
        "Filter type specified as DSCP.";
    }
    identity source-ipv4-prefix {
      base filter-type;
      description
        "Filter type specified as source IPv4 prefix.";
    }
    identity destination-ipv4-prefix {
      base filter-type;
      description
        "Filter type specified as destination IPv4 prefix.";
    }
    identity source-ipv6-prefix {
      base filter-type;
      description
        "Filter type specified as source IPv6 prefix.";
    }
    identity destination-ipv6-prefix {
      base filter-type;
      description
        "Filter type specified as destination IPv6 prefix.";
    }
    identity source-port {
      base filter-type;
      description
        "Filter type specified as source port.";
    }
    identity destination-port {
      base filter-type;
      description
        "Filter type specified as destination port.";
    }
    identity protocol {
      base filter-type;
      description

        "Filter type specified as protocol.";
    }
    identity traffic-group-name {
      base filter-type;
      description
```

```
        "Filter type specified as traffic group name.";
    }
    identity filter-match-all {
      base filter-type;
      description
        "Filter type specified as match all.";
    }

    typedef match-operation-type {
      type enumeration {
        enum match-all {
          description
            "Classifier entry filters logical AND operation";
        }
        enum match-any {
          description
            "Classifier entry filters logical OR operation";
        }
      }
      default "match-all";
      description
        "Filter match logical operation type";
    }

    identity direction {
      description
        "Base identity for traffic direction";
    }
    identity ingress {
      base direction;
      description
        "Direction of traffic coming into the network entry";
    }
    identity egress {
      base direction;
      description
        "Direction of traffic going out of the network entry";
    }

    identity meter-action-type {
      description
        "Base identify for actions related to metering";
    }

    identity action-drop {
      base meter-action-type;
        description
          "Drop packets that conform/exceed/violate to the set value.";
```

```
    }
    identity action-transmit {
      base meter-action-type;
        description
          "Transmit packets that conform/exceed/violate to the set
           value.";
    }
    identity action-mark {
      base meter-action-type;
        description
          "Mark packets that conform/exceed/violate to the set value.";
    }

    identity clear-counters-type {
      description
        "Base identify for clear interface counters related options";
    }
    identity all-counters {
      base clear-counters-type;
        description
          "Clear all counters in both directions.";
    }
    identity inbound-counters {
      base clear-counters-type;
        description
          "Clear counters in inbound direction.";
    }
    identity outbound-counters {
      base clear-counters-type;
        description
          "Clear counters in outbound direction.";
    }
  }
  <CODE ENDS>
```

Figure 14: ietf-qos-types module

5. IANA Considerations

This document registers seven URIs and seven YANG modules.

5.1. URI Registration

Following the format in the IETF XML registry [RFC3688], the following registration is requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-qos-types URI:

urn:ietf:params:xml:ns:yang:ietf-diffserv URI:
urn:ietf:params:xml:ns:yang:ietf-qos-action URI:
urn:ietf:params:xml:ns:yang:ietf-qos-oper URI:
urn:ietf:params:xml:ns:yang:ietf-queue-policy URI:
urn:ietf:params:xml:ns:yang:ietf-scheduler-policy URI:
urn:ietf:params:xml:ns:yang:ietf-traffic-policy

Registrant Contact: The IESG. XML: N/A, the requested URI is an XML namespace.

5.2. YANG Module Name Registration

This document registers seven YANG modules in the YANG Module Names registry YANG [RFC6020].

```
            name: ietf-qos-types
            namespace: urn:ietf:params:xml:ns:yang:ietf-qos-types
            prefix: qos-types
            reference: RFC XXXX

            name: ietf-diffserv
            namespace: urn:ietf:params:xml:ns:yang:ietf-diffserv
            prefix: diffserv
            reference: RFC XXXX

            name: ietf-qos-action
            namespace: urn:ietf:params:xml:ns:yang:ietf-qos-action
            prefix: action
            reference: RFC XXXX

            name: ietf-qos-oper
            namespace: urn:ietf:params:xml:ns:yang:ietf-qos-oper
            prefix: oper
            reference: RFC XXXX

            name: ietf-queue-policy
            namespace: urn:ietf:params:xml:ns:yang:ietf-queue-policy
            prefix: queue-policy
            reference: RFC XXXX

            name: ietf-scheduler-policy
            namespace: urn:ietf:params:xml:ns:yang:ietf-scheduler-policy
            prefix: scheduler-policy
            reference: RFC XXXX

            name: ietf-traffic-policy
            namespace: urn:ietf:params:xml:ns:yang:ietf-traffic-policy
            prefix: policy
            reference: RFC XXXX
```

6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446]. The NETCONF Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. Some of the subtrees and data nodes and their sensitivity/vulnerability are described here.

- 'filter-operation' and 'filter'. 'filter-operation' includes the ability to 'match all', which is a logical AND operation or 'match any', which is a logical OR operation. Both the operations have an impact on the traffic that is being classified. Similarly, the 'filter' list decides what parts of the packet will examined, which will also impact the traffic that is being classified.

- 'action'. The 'action' decides what action will be taken on the packet. That includes whether the packet will be marked, queued, or just discarded.

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

- 'conform-pkts', 'conform-bytes', 'exceed-pkts', 'exceed-bytes', 'voilate-pkts' or 'voilate-bytes'. All these statistics combined with their 'meter-id' are an indication of what is happening in the network, and can allow for an intruder insight into how to disrupt the traffic.

- 'tail-drop-pkts', 'red-drop-bytes', 'wred-stats' are examples of statistics that indicate the kind of traffic that is being profiled for drop or marked for Explicit Congestion Notification (ECN), and can give an insight into the operation of a network.

Some of the RPC operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

- The model allows for statistics to be cleared by the 'clear' RPC operation, causing all the individual statistics to be cleared.

7. Acknowledgement

The authors wish to thank Ruediger Geib, Fred Baker, Greg Mirsky, Tom Petch, Acee Lindem, Adrian Farrel, David Black, Jurgen Schonwalder, Colin Perkins, Joseph Touch, Zaheduzzaman Sarker many others for their helpful comments.

8. Contributors

The following people have substantially contributed to the editing of this document:

Norm Strahle
Email: nstrahle@juniper.net .

Shitanshu Shah
Email: svshah@cisco.com .

9. References

9.1. Normative References

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

[RFC9911] Sch旦nw辰lder, J., Ed., "Common YANG Data Types", RFC 9911, DOI 10.17487/RFC9911, December 2025, <https://www.rfc-editor.org/info/rfc9911>.

[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>.

[RFC8436] Fairhurst, G., "Update to IANA Registration Procedures for Pool 3 Values in the Differentiated Services Field Codepoints (DSCP) Registry", RFC 8436, DOI 10.17487/RFC8436, August 2018, <https://www.rfc-editor.org/info/rfc8436>.

9.2. Informative References

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <https://www.rfc-editor.org/info/rfc2474>.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, DOI 10.17487/RFC2475, December 1998, <https://www.rfc-editor.org/info/rfc2475>.

[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>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

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

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

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

Appendix A. Complete tree Diagrams

This section carries complete tree diagrams of each module defined in this draft. The tree diagrams use the notation defined in YANG Tree Diagrams [RFC8340].

A.1. Traffic Policy Module Tree Diagram

```
   module: ietf-traffic-policy
     +--rw classifiers
     |  +--rw classifier* [name]
     |     +--rw name                string
     |     +--rw description?        string
     |     +--rw filter-operation?   qos-types:match-operation-type
     |     +--rw filter* [type logical-not]
     |        +--rw type           identityref
     |        +--rw logical-not    boolean
     +--rw policies
        +--rw policy* [name type]
           +--rw name           string
           +--rw type           identityref
           +--rw description?   string
           +--rw classifier* [name]
              +--rw name      union
              +--rw inline!
              |  +--rw filter-operation?
              |  |       qos-types:match-operation-type
              |  +--rw filter* [type logical-not]
              |     +--rw type           identityref
              |     +--rw logical-not    boolean
              +--rw action* [type]
                 +--rw type    identityref

     augment /if:interfaces/if:interface:
       +--rw qos-target-policy* [direction type]
          +--rw direction    identityref
          +--rw type         identityref
          +--rw name         -> /policies/policy/name
```

Figure 7: Traffic Policy Tree Diagram

A.2. QoS Action Module Tree Diagram

```
   module: ietf-qos-action
     +--rw meters
     |  +--rw meter* [name]
     |     +--rw name                                    string
     |     +--rw (meter-types)?
     |        +--:(single-rate-two-color-meter-type)
     |        |  +--rw single-rate-two-color-meter
     |        |     +--rw committed-information-rate?   uint64
     |        |     +--rw committed-burst-size?         uint64
     |        |     +--rw conform-action
     |        |     |  +--rw action-type?          identityref
     |        |     |  +--rw dscp-mark?            inet:dscp
     |        |     |  +--rw traffic-group-mark?   string
```

```
|        |     +--rw exceed-action
|        |        +--rw action-type?          identityref
|        |        +--rw dscp-mark?            inet:dscp
|        |        +--rw traffic-group-mark?   string
|        +--:(single-rate-three-color-meter-type)
|        |  +--rw single-rate-three-color-meter
|        |     +--rw committed-information-rate?   uint64
|        |     +--rw committed-burst-size?         uint64
|        |     +--rw excess-burst-size?            uint64
|        |     +--rw conform-action
|        |     |  +--rw action-type?          identityref
|        |     |  +--rw dscp-mark?            inet:dscp
|        |     |  +--rw traffic-group-mark?   string
|        |     +--rw exceed-action
|        |     |  +--rw action-type?          identityref
|        |     |  +--rw dscp-mark?            inet:dscp
|        |     |  +--rw traffic-group-mark?   string
|        |     +--rw violate-action
|        |        +--rw action-type?          identityref
|        |        +--rw dscp-mark?            inet:dscp
|        |        +--rw traffic-group-mark?   string
|        +--:(two-rate-three-color-meter-type)
|           +--rw two-rate-three-color-meter
|              +--rw committed-information-rate?   uint64
|              +--rw committed-burst-size?         uint64
|              +--rw peak-information-rate?        uint64
|              +--rw peak-burst-size?              uint64
|              +--rw conform-action
|              |  +--rw action-type?          identityref
|              |  +--rw dscp-mark?            inet:dscp
|              |  +--rw traffic-group-mark?   string
|              +--rw exceed-action
|              |  +--rw action-type?          identityref
|              |  +--rw dscp-mark?            inet:dscp
|              |  +--rw traffic-group-mark?   string
|              +--rw violate-action
|                 +--rw action-type?          identityref
|                 +--rw dscp-mark?            inet:dscp
|                 +--rw traffic-group-mark?   string
|
+--rw queues
   +--rw queue* [name]
      +--rw name     string
      +--rw queue
         +--rw priority
         |  +--rw level?   uint8
         +--rw min-rate
         |  +--rw value?   uint64
         |  +--rw unit?    identityref
```

```
            +--rw max-rate
            |  +--rw value?         uint64
            |  +--rw unit?          identityref
            |  +--rw burst-value?   uint64
            |  +--rw burst-unit?    identityref
            +--rw algorithmic-drop
               +--rw (drop-algorithm)?
                  +--:(tail-drop)
                  |  +--rw tail-drop
                  |     +--rw tail-drop?   empty
                  +--:(red)
                  |  +--rw red
                  |     +--rw min-threshold-val?      uint64
                  |     +--rw min-threshold-unit?     identityref
                  |     +--rw max-threshold-val?      uint64
                  |     +--rw max-threshold-unit?     identityref
                  |     +--rw weight?                 uint8
                  |     +--rw max-probability-val?    uint64
                  |     +--rw max-probability-unit?   identityref
                  |     +--rw ecn-enabled?            boolean
                  +--:(wred)
                  |  +--rw wred
                  |     +--rw wred* [profile]
                  |     |  +--rw profile                 uint8
                  |     |  +--rw color-type?             identityref
                  |     |  +--rw color-val* [min max]
                  |     |  |  +--rw min    uint8
                  |     |  |  +--rw max    uint8
                  |     |  +--rw min-threshold-val?      uint64
                  |     |  +--rw min-threshold-unit?     identityref
                  |     |  +--rw max-threshold-val?      uint64
                  |     |  +--rw max-threshold-unit?     identityref
                  |     |  +--rw weight?                 uint8
                  |     |  +--rw max-probability-val?    uint64
                  |     |  +--rw max-probability-unit?   identityref
                  |     +--rw ecn-enabled?   boolean
                  +--:(codel)
                  |  +--rw codel
                  |     +--rw target?        uint64
                  |     +--rw interval?      uint64
                  |     +--rw ecn-enabled?   boolean
                  +--:(fq-codel)
                     +--rw fq-codel
                        +--rw target?        uint64
                        +--rw interval?      uint64
                        +--rw flows?         uint64
                        +--rw ecn-enabled?   boolean
```

Figure 16: QoS Action Module Tree Diagram

A.3. Diffserv Module Tree Diagram

```
   module: ietf-diffserv

     augment /traffic-policy:classifiers/traffic-policy:classifier
               /traffic-policy:filter:
       +--rw (filter-param)?
          +--:(dscp)
          |  +--rw dscp* [min max]
          |     +--rw min    inet:dscp
          |     +--rw max    inet:dscp
          +--:(source-ipv4-prefix)
          |  +--rw source-ipv4-prefix*        inet:ipv4-prefix
          +--:(destination-ipv4-prefix)
          |  +--rw destination-ipv4-prefix*   inet:ipv4-prefix
          +--:(source-ipv6-prefix)
          |  +--rw source-ipv6-prefix*        inet:ipv6-prefix
          +--:(destination-ipv6-prefix)
          |  +--rw destination-ipv6-prefix*   inet:ipv6-prefix
          +--:(source-port)
          |  +--rw source-port* [min max]
          |     +--rw min    inet:port-number
          |     +--rw max    inet:port-number
          +--:(destination-port)
          |  +--rw destination-port* [min max]
          |     +--rw min    inet:port-number
          |     +--rw max    inet:port-number
          +--:(protocol)
          |  +--rw protocol* [min max]
          |     +--rw min    inet:protocol-number
          |     +--rw max    inet:protocol-number
          +--:(traffic-group)
          |  +--rw traffic-group
          |     +--rw name?   string
          +--:(filter-match-all)
             +--rw match-all
                +--rw action?   empty
     augment /traffic-policy:policies/traffic-policy:policy
               /traffic-policy:classifier/traffic-policy:inline
               /traffic-policy:filter:
       +--rw (filter-params)?
          +--:(dscp)
          |  +--rw dscp* [min max]
          |     +--rw min    inet:dscp
          |     +--rw max    inet:dscp
          +--:(source-ipv4-prefix)

          |  +--rw source-ipv4-prefix*        inet:ipv4-prefix
```

```
        +--:(destination-ipv4-prefix)
        |  +--rw destination-ipv4-prefix*   inet:ipv4-prefix
        +--:(source-ipv6-prefix)
        |  +--rw source-ipv6-prefix*        inet:ipv6-prefix
        +--:(destination-ipv6-prefix)
        |  +--rw destination-ipv6-prefix*   inet:ipv6-prefix
        +--:(source-port)
        |  +--rw source-port* [min max]
        |     +--rw min    inet:port-number
        |     +--rw max    inet:port-number
        +--:(destination-port)
        |  +--rw destination-port* [min max]
        |     +--rw min    inet:port-number
        |     +--rw max    inet:port-number
        +--:(protocol)
        |  +--rw protocol* [min max]
        |     +--rw min    inet:protocol-number
        |     +--rw max    inet:protocol-number
        +--:(traffic-group)
        |  +--rw traffic-group
        |     +--rw name?   string
        +--:(filter-match-all)
           +--rw match-all
              +--rw action?   empty
    augment /traffic-policy:policies/traffic-policy:policy
              /traffic-policy:classifier/traffic-policy:action:
      +--rw (action-params)?
        +--:(dscp-marking)
        |  +--rw dscp
        |     +--rw dscp?   inet:dscp
        +--:(meter-inline)
        |  +--rw (meter-types)?
        |     +--:(single-rate-two-color-meter-type)
        |     |  +--rw single-rate-two-color-meter
        |     |     +--rw committed-information-rate?   uint64
        |     |     +--rw committed-burst-size?         uint64
        |     |     +--rw conform-action
        |     |     |  +--rw action-type?          identityref
        |     |     |  +--rw dscp-mark?            inet:dscp
        |     |     |  +--rw traffic-group-mark?   string
        |     |     +--rw exceed-action
        |     |        +--rw action-type?          identityref
        |     |        +--rw dscp-mark?            inet:dscp
        |     |        +--rw traffic-group-mark?   string
        |     +--:(single-rate-three-color-meter-type)
        |     |  +--rw single-rate-three-color-meter
        |     |     +--rw committed-information-rate?   uint64
        |     |     +--rw committed-burst-size?         uint64
```

```
|     |     +--rw excess-burst-size?           uint64
|     |     +--rw conform-action
|     |     |  +--rw action-type?          identityref
|     |     |  +--rw dscp-mark?            inet:dscp
|     |     |  +--rw traffic-group-mark?   string
|     |     +--rw exceed-action
|     |     |  +--rw action-type?          identityref
|     |     |  +--rw dscp-mark?            inet:dscp
|     |     |  +--rw traffic-group-mark?   string
|     |     +--rw violate-action
|     |        +--rw action-type?          identityref
|     |        +--rw dscp-mark?            inet:dscp
|     |        +--rw traffic-group-mark?   string
|     +--:(two-rate-three-color-meter-type)
|        +--rw two-rate-three-color-meter
|           +--rw committed-information-rate?   uint64
|           +--rw committed-burst-size?         uint64
|           +--rw peak-information-rate?        uint64
|           +--rw peak-burst-size?              uint64
|           +--rw conform-action
|           |  +--rw action-type?          identityref
|           |  +--rw dscp-mark?            inet:dscp
|           |  +--rw traffic-group-mark?   string
|           +--rw exceed-action
|           |  +--rw action-type?          identityref
|           |  +--rw dscp-mark?            inet:dscp
|           |  +--rw traffic-group-mark?   string
|           +--rw violate-action
|              +--rw action-type?          identityref
|              +--rw dscp-mark?            inet:dscp
|              +--rw traffic-group-mark?   string
+--:(meter-reference)
|  +--rw meter
|     +--rw name    -> /qos-action:meters/meter/name
|     +--rw type    identityref
+--:(traffic-group-marking)
|  +--rw traffic-group
|     +--rw traffic-group?   string
+--:(child-policy) {qos-types:child-policy}?
|  +--rw child-policy {qos-types:child-policy}?
|     +--rw name?   string
+--:(qos-count) {qos-types:qos-count}?
|  +--rw qos-count {qos-types:qos-count}?
|     +--rw qos-count-action?   empty
+--:(named-qos-count) {qos-types:named-qos-count}?
|  +--rw named-qos-count {qos-types:named-qos-count}?
|     +--rw name?   string
```

```
          +--:(queue-inline)
          |  +--rw queue
          |     +--rw priority
          |     |  +--rw level?   uint8
          |     +--rw min-rate
          |     |  +--rw value?   uint64
          |     |  +--rw unit?    identityref
          |     +--rw max-rate
          |     |  +--rw value?         uint64
          |     |  +--rw unit?          identityref
          |     |  +--rw burst-value?   uint64
          |     |  +--rw burst-unit?    identityref
          |     +--rw algorithmic-drop
          |        +--rw (drop-algorithm)?
          |           +--:(tail-drop)
          |           |  +--rw tail-drop
          |           |     +--rw tail-drop?   empty
          |           +--:(red)
          |           |  +--rw red
          |           |     +--rw min-threshold-val?      uint64
          |           |     +--rw min-threshold-unit?     identityref
          |           |     +--rw max-threshold-val?      uint64
          |           |     +--rw max-threshold-unit?     identityref
          |           |     +--rw weight?                 uint8
          |           |     +--rw max-probability-val?    uint64
          |           |     +--rw max-probability-unit?   identityref
          |           |     +--rw ecn-enabled?            boolean
          |           +--:(wred)
          |           |  +--rw wred
          |           |     +--rw wred* [profile]
          |           |     |  +--rw profile                uint8
          |           |     |  +--rw color-type?            identityref
          |           |     |  +--rw color-val* [min max]
          |           |     |  |  +--rw min    uint8
          |           |     |  |  +--rw max    uint8
          |           |     |  +--rw min-threshold-val?     uint64
          |           |     |  +--rw min-threshold-unit?    identityref
          |           |     |  +--rw max-threshold-val?     uint64
          |           |     |  +--rw max-threshold-unit?    identityref
          |           |     |  +--rw weight?                uint8
          |           |     |  +--rw max-probability-val?   uint64
          |           |     |  +--rw max-probability-unit?  identityref
          |           |     +--rw ecn-enabled?   boolean
          |           +--:(codel)
          |           |  +--rw codel
          |           |     +--rw target?        uint64
          |           |     +--rw interval?      uint64
          |           |     +--rw ecn-enabled?   boolean
```

```
         |           +--:(fq-codel)
         |              +--rw fq-codel
         |                 +--rw target?        uint64
         |                 +--rw interval?      uint64
         |                 +--rw flows?         uint64
         |                 +--rw ecn-enabled?   boolean
         +--:(scheduler-inline)
            +--rw scheduler
               +--rw min-rate
               |  +--rw value?   uint64
               |  +--rw unit?    identityref
               +--rw max-rate
                  +--rw value?         uint64
                  +--rw unit?          identityref
                  +--rw burst-value?   uint64
                  +--rw burst-unit?    identityref
```

Figure 8: Diffserv Module Tree Diagram

A.4. QoS Operational Module Tree Diagram

```
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* []
       |  |  +--ro name?               string
       |  |  +--ro classified-pkts?    yang:zero-based-counter64
       |  |  +--ro classified-bytes?   yang:zero-based-counter64
       |  |  +--ro classified-rate?    yang:gauge64
       |  +--ro named* []
       |  |  +--ro name?             string
       |  |  +--ro aggregated
       |  |  |  +--ro pkts?    yang:zero-based-counter64
       |  |  |  +--ro bytes?   yang:zero-based-counter64
       |  |  |  +--ro rate?    yang:gauge64
       |  |  +--ro non-aggregated
       |  |     +--ro classifier* []
       |  |     |  +--ro name?               string
       |  |     |  +--ro classified-pkts?
       |  |     |  |       yang:zero-based-counter64
       |  |     |  +--ro classified-bytes?
       |  |     |  |       yang:zero-based-counter64
       |  |     |  +--ro classified-rate?    yang:gauge64
       |  |     +--ro metering* []
```

```
|  |     |  +--ro meter-id?        string
|  |     |  +--ro conform-pkts?    yang:zero-based-counter64
|  |     |  +--ro conform-bytes?   yang:zero-based-counter64
|  |     |  +--ro conform-rate?    yang:gauge64
|  |     |  +--ro exceed-pkts?     yang:zero-based-counter64
|  |     |  +--ro exceed-bytes?    yang:zero-based-counter64
|  |     |  +--ro exceed-rate?     yang:gauge64
|  |     |  +--ro violate-pkts?    yang:zero-based-counter64
|  |     |  +--ro violate-bytes?   yang:zero-based-counter64
|  |     |  +--ro violate-rate?    yang:gauge64
|  |     |  +--ro drop-pkts?       yang:zero-based-counter64
|  |     |  +--ro drop-bytes?      yang:zero-based-counter64
|  |     +--ro queueing* []
|  |        +--ro queue-id?                    string
|  |        +--ro transmit-pkts?
|  |        |       yang:zero-based-counter64
|  |        +--ro transmit-bytes?
|  |        |       yang:zero-based-counter64
|  |        +--ro queue-current-size-bytes?   yang:gauge64
|  |        +--ro queue-average-size-bytes?   yang:gauge64
|  |        +--ro queue-peak-size-bytes?      yang:gauge64
|  |        +--ro tail-drop-pkts?
|  |        |       yang:zero-based-counter64
|  |        +--ro tail-drop-bytes?
|  |        |       yang:zero-based-counter64
|  |        +--ro red-statistics
|  |        |  +--ro drop-pkts?
|  |        |  |       yang:zero-based-counter64
|  |        |  +--ro drop-bytes?
|  |        |  |       yang:zero-based-counter64
|  |        |  +--ro ecn-marked-pkts?
|  |        |  |       yang:zero-based-counter64
|  |        |  +--ro ecn-marked-bytes?
|  |        |  |       yang:zero-based-counter64
|  |        |  +--ro wred-stats* []
|  |        |     +--ro profile?            uint8
|  |        |     +--ro drop-pkts?
|  |        |     |       yang:zero-based-counter64
|  |        |     +--ro drop-bytes?
|  |        |     |       yang:zero-based-counter64
|  |        |     +--ro ecn-marked-pkts?
|  |        |     |       yang:zero-based-counter64
|  |        |     +--ro ecn-marked-bytes?
|  |        |             yang:zero-based-counter64
|  |        +--ro codel-statistics
|  |           +--ro drop-pkts?
|  |           |       yang:zero-based-counter64
|  |           +--ro drop-bytes?
```

```
|  |  |           |       yang:zero-based-counter64
|  |  |           +--ro ecn-marked-pkts?
|  |  |           |       yang:zero-based-counter64
|  |  |           +--ro ecn-marked-bytes?
|  |  |           |       yang:zero-based-counter64
|  |  |           +--ro new-flow-count?
|  |  |                   yang:zero-based-counter64
|  +--ro metering* []
|  |  +--ro meter-id?        string
|  |  +--ro conform-pkts?    yang:zero-based-counter64
|  |  +--ro conform-bytes?   yang:zero-based-counter64
|  |  +--ro conform-rate?    yang:gauge64
|  |  +--ro exceed-pkts?     yang:zero-based-counter64
|  |  +--ro exceed-bytes?    yang:zero-based-counter64
|  |  +--ro exceed-rate?     yang:gauge64
|  |  +--ro violate-pkts?    yang:zero-based-counter64
|  |  +--ro violate-bytes?   yang:zero-based-counter64
|  |  +--ro violate-rate?    yang:gauge64
|  |  +--ro drop-pkts?       yang:zero-based-counter64
|  |  +--ro drop-bytes?      yang:zero-based-counter64
|  +--ro queueing* []
|     +--ro queue-id?                   string
|     +--ro transmit-pkts?
|     |       yang:zero-based-counter64
|     +--ro transmit-bytes?
|     |       yang:zero-based-counter64
|     +--ro queue-current-size-bytes?   yang:gauge64
|     +--ro queue-average-size-bytes?   yang:gauge64
|     +--ro queue-peak-size-bytes?      yang:gauge64
|     +--ro tail-drop-pkts?
|     |       yang:zero-based-counter64
|     +--ro tail-drop-bytes?
|     |       yang:zero-based-counter64
|     +--ro red-statistics
|     |  +--ro drop-pkts?          yang:zero-based-counter64
|     |  +--ro drop-bytes?         yang:zero-based-counter64
|     |  +--ro ecn-marked-pkts?    yang:zero-based-counter64
|     |  +--ro ecn-marked-bytes?   yang:zero-based-counter64
|     |  +--ro wred-stats* []
|     |     +--ro profile?            uint8
|     |     +--ro drop-pkts?
|     |     |       yang:zero-based-counter64
|     |     +--ro drop-bytes?
|     |     |       yang:zero-based-counter64
|     |     +--ro ecn-marked-pkts?
|     |     |       yang:zero-based-counter64
|     |     +--ro ecn-marked-bytes?
|     |             yang:zero-based-counter64
```

```
          |     +--ro codel-statistics
          |        +--ro drop-pkts?          yang:zero-based-counter64
          |        +--ro drop-bytes?         yang:zero-based-counter64
          |        +--ro ecn-marked-pkts?    yang:zero-based-counter64
          |        +--ro ecn-marked-bytes?   yang:zero-based-counter64
          |        +--ro new-flow-count?     yang:zero-based-counter64
          +---x clear
             +---w input
             |  +---w category?     identityref
             |  +---w started-at?   yang:date-and-time
             +--ro output
                +--ro finished-at?   yang:date-and-time
```

Figure 9: QoS Operational Module Tree Diagram

A.5. Scheduler Policy Module Tree Diagram

```
   module: ietf-scheduler-policy

     augment /traffic-policy:policies/traffic-policy:policy
               /traffic-policy:classifier/traffic-policy:inline
               /traffic-policy:filter:
       +--rw (filter-params)?
          +--:(filter-match-all)
             +--rw match-all
                +--rw action?   empty
     augment /traffic-policy:policies/traffic-policy:policy
               /traffic-policy:classifier/traffic-policy:action:
       +--rw (action-params)?
          +--:(scheduler)
          |  +--rw scheduler
          |     +--rw min-rate
          |     |  +--rw value?   uint64
          |     |  +--rw unit?    identityref
          |     +--rw max-rate
          |        +--rw value?         uint64
          |        +--rw unit?          identityref
          |        +--rw burst-value?   uint64
          |        +--rw burst-unit?    identityref
          +--:(queue-policy-name)
             +--rw queue-policy-name
                +--rw queue-policy
                         -> /traffic-policy:policies/policy/name
```

Figure 10: Scheduler Policy Module Tree Diagram

A.6. Queue Policy Module Tree Diagram

```
   module: ietf-queue-policy

     augment /traffic-policy:policies/traffic-policy:policy
               /traffic-policy:classifier/traffic-policy:inline
               /traffic-policy:filter:
       +--rw (filter-params)?
          +--:(traffic-group)
          |  +--rw traffic-group
          |     +--rw name?   string
          +--:(filter-match-all)
             +--rw match-all
                +--rw action?   empty
     augment /traffic-policy:policies/traffic-policy:policy
               /traffic-policy:classifier/traffic-policy:action:
       +--rw (action-params)?
          +--:(queue-template-name)
          |  +--rw queue-reference
          |     +--rw queue-name    -> /qos-action:queues/queue/name
          +--:(queue-inline)
             +--rw queue
                +--rw priority
                |  +--rw level?   uint8
                +--rw min-rate
                |  +--rw value?   uint64
                |  +--rw unit?    identityref
                +--rw max-rate
                |  +--rw value?         uint64
                |  +--rw unit?          identityref
                |  +--rw burst-value?   uint64
                |  +--rw burst-unit?    identityref
                +--rw algorithmic-drop
                   +--rw (drop-algorithm)?
                      +--:(tail-drop)
                      |  +--rw tail-drop
                      |     +--rw tail-drop?   empty
                      +--:(red)
                      |  +--rw red
                      |     +--rw min-threshold-val?      uint64
                      |     +--rw min-threshold-unit?     identityref
                      |     +--rw max-threshold-val?      uint64
                      |     +--rw max-threshold-unit?     identityref
                      |     +--rw weight?                 uint8
                      |     +--rw max-probability-val?    uint64
                      |     +--rw max-probability-unit?   identityref
                      |     +--rw ecn-enabled?            boolean
                      +--:(wred)
```

```
|  +--rw wred
|     +--rw wred* [profile]

|     |  +--rw profile                 uint8
|     |  +--rw color-type?             identityref
|     |  +--rw color-val* [min max]
|     |  |  +--rw min    uint8
|     |  |  +--rw max    uint8
|     |  +--rw min-threshold-val?      uint64
|     |  +--rw min-threshold-unit?     identityref
|     |  +--rw max-threshold-val?      uint64
|     |  +--rw max-threshold-unit?     identityref
|     |  +--rw weight?                 uint8
|     |  +--rw max-probability-val?    uint64
|     |  +--rw max-probability-unit?   identityref
|     +--rw ecn-enabled?   boolean
+--:(codel)
|  +--rw codel
|     +--rw target?        uint64
|     +--rw interval?      uint64
|     +--rw ecn-enabled?   boolean
+--:(fq-codel)
   +--rw fq-codel
      +--rw target?        uint64
      +--rw interval?      uint64
      +--rw flows?         uint64
      +--rw ecn-enabled?   boolean
```

Figure 11: Queue Policy Module Tree Diagram

Appendix B. Company A and Company B examples

Company A and Company B Diffserv modules augments all the filter types of the QoS Classifier module as well as the QoS Policy module that allow it to define marking, metering, min-rate, max-rate actions. Queueing and metering counters are realized by augmenting of the QoS Target module.

B.1. Example of Company A Diffserv Model

The following Company A vendor example augments the QoS and Diffserv model, demonstrating some of the following functionality:

* use of template based classifier definitions

- use of single policy type modelling queue, scheduler policy, and a filter policy. All of these policies either augment the QoS policy or the Diffserv modules

- use of inline actions in a policy

- flexibility in marking dscp or metadata at ingress and/or egress.

<CODE BEGINS> file "example-compa-diffserv@2026-03-02.yang"

```
module example-compa-diffserv {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:example-compa-diffserv";
  prefix example;

  import ietf-qos-types {
    prefix qos-types;
    reference
      "RFC XXXX: YANG Models for Quality of Service (QoS).";
  }
  import ietf-traffic-policy {
    prefix traffic-policy;
    reference
      "RFC XXXX: YANG Models for Quality of Service (QoS).";
  }
  import ietf-qos-action {
    prefix qos-action;
    reference
      "RFC XXXX: YANG Models for Quality of Service (QoS).";
  }
  import ietf-diffserv {
    prefix diffserv;
    reference
      "RFC XXXX: YANG Models for Quality of Service (QoS).";
  }
  organization "Company A";
  contact
    "Editor:   XYZ
               <mailto:xyz@compa.com>";
  description
    "This module contains a collection of YANG definitions of
     companyA diffserv specification extension.";
  revision 2026-03-02 {
    description
      "Initial revision for diffserv actions on network packets.";
    reference
      "RFC XXXX: A YANG Data Model for Quality of Service (QoS).";
  }

  identity default-policy-type {
    base qos-types:policy-type;
    description
      "This defines default policy-type.";
```

```
     }

     identity qos-group {
       base qos-types:filter-type;
       description
         "Filter-type to match on  packet metadata. This metadata
          can be used for packet marking, queuing for traffic in
          egress direction.";
     }

     grouping qos-group {
       list qos-group {
         key "min max";
         description
           "List of qos-groups to match on. A QoS group is a set of
            packets tagged as group.";
         leaf min {
           type uint8;
           description
             "Minimum value of qos-group range.";
         }
         leaf max {
           type uint8;
           must ". >= ../min" {
             error-message
             "The qos-group max must be
              greater than or equal to qos-group min.";
           }
           description
             "Maximum value of qos-group range.";
         }
       }
       description
         "Filter containing list of qos-group ranges.";
     }
     grouping bw-remaining-percent {
       container bw-remaining-percent {
         leaf value {
           type uint8;
           units "percentage";
           description
             "Percentage of the remaining bandwidth present.";
         }
         description
           "Remaining bandwidth of the link in percent.";
       }
       description
         "Grouping for bandwidth remaining after
```

```
        reserved and priority bandwidth.";
    }

    grouping qos-group-marking {
      container qos-group {
        leaf qos-group {
          type uint8;
          description
            "Mark metadata information in the network packet";
        }
        description
          "Container for QoS-Group marking.";
      }
      description
        "Grouping for QoS-Group marking.";
    }

    augment "/traffic-policy:classifiers/" +
      "traffic-policy:classifier/" +
      "traffic-policy:filter/diffserv:filter-param" {
      case qos-group {
        uses qos-group;
        description
          "Filter containing list of qos-group ranges.
          QoS-group represent packet metadata information
          in a network packet.";
      }
      description
        "Augmentation of classifier filters.";
    }

    augment "/traffic-policy:policies/traffic-policy:policy/" +
      "traffic-policy:classifier/" +
      "traffic-policy:action" {
      choice action-params {
        case priority {
          uses qos-action:priority;
        }
        case min-rate {
          uses qos-action:min-rate;
        }
        case max-rate {
          uses qos-action:max-rate;
        }
        case bw-remaining {
          uses bw-remaining-percent;
        }
        case mark-qos-group {
```

```
          uses qos-group-marking;
        }
        description

          "Choice of action parameters.";
      }
      description
        "Augments policy entry to add action parameters.";
    }

    augment "/traffic-policy:policies" +
      "/traffic-policy:policy" +
      "/traffic-policy:classifier" +
      "/traffic-policy:action" +
      "/diffserv:action-params" +
      "/diffserv:meter-inline" +
      "/diffserv:meter-types" +
      "/diffserv:single-rate-two-color-meter-type" +
      "/diffserv:single-rate-two-color-meter" {
      description
        "Augment the single-rate-two-color meter to add" +
        "color classifiers.";
      container conform-color {
        uses traffic-policy:generic-classifier-attr;
        description
          "Conform color classifier container.";
      }
      container exceed-color {
        uses traffic-policy:generic-classifier-attr;
        description
          "Exceed color classifier container.";
      }
    }

    augment "/traffic-policy:policies" +
      "/traffic-policy:policy" +
      "/traffic-policy:classifier" +
      "/traffic-policy:action" +
      "/diffserv:action-params" +
      "/diffserv:meter-inline" +
      "/diffserv:meter-types" +
      "/diffserv:single-rate-three-color-meter-type" +
      "/diffserv:single-rate-three-color-meter" {
      description
        "Augment the one-rate-tri-color meter to add" +
        "color classifiers.";
      container conform-color {
        uses traffic-policy:generic-classifier-attr;
```

```
        description
          "Conform color classifier container.";
      }
      container exceed-color {

        uses traffic-policy:generic-classifier-attr;
        description
          "Exceed color classifier container.";
      }
      container violate-color {
        uses traffic-policy:generic-classifier-attr;
        description
          "Violate color classifier container.";
      }
    }

    augment "/traffic-policy:policies" +
      "/traffic-policy:policy" +
      "/traffic-policy:classifier" +
      "/traffic-policy:action" +
      "/diffserv:action-params" +
      "/diffserv:meter-inline" +
      "/diffserv:meter-types" +
      "/diffserv:two-rate-three-color-meter-type" +
      "/diffserv:two-rate-three-color-meter" {
      description
        "Augment the two-rate-tri-color meter to add" +
        "color classifiers.";
      container conform-color {
        uses traffic-policy:generic-classifier-attr;
        description
          "Conform color classifier container.";
      }
      container exceed-color {
        uses traffic-policy:generic-classifier-attr;
        description
          "Exceed color classifier container.";
      }
      container violate-color {
        uses traffic-policy:generic-classifier-attr;
        description
          "Violate color classifier container.";
      }
    }
  }
  <CODE ENDS>
```

B.2. Example of Company B Diffserv Model

The following vendor example augments the QoS and Diffserv model, demonstrating some of the following functionality:

- use of inline classifier definitions (defined inline in the policy vs referencing an externally defined classifier)

- use of mulitple policy types, e.g. a queue policy, a scheduler policy, and a filter policy. All of these policies either augment the QoS policy or the Diffserv modules

- use of a queue module, which uses and extends the queue grouping from the QoS Action module

- use of meter templates (v.s. meter inline)

- use of internal meta data for classification and marking

<CODE BEGINS>
file "example-compb-diffserv-filter-policy@2026-03-02.yang"

```
module example-compb-diffserv-filter-policy {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:" +
            "example-compb-diffserv-filter-policy";
  prefix compb-filter-policy;

  import ietf-qos-types {
    prefix qos-types;
    reference
      "RFC XXXX: YANG Models for Quality of Service (QoS).";
  }
  import ietf-traffic-policy {
    prefix traffic-policy;
  }
  import ietf-qos-action {
    prefix qos-action;
  }
  import ietf-diffserv {
    prefix diffserv;
  }

  organization
    "Company B";

  contact
    "Editor:   XYZ
               <mailto:xyz@compb.com>";
```

```
     description
       "This module contains a collection of YANG definitions for
        configuring diffserv specification implementations.

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

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Revised BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (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 2026-03-02 {
       description
         "Initial revision of diffserv policy for Company B.";
       reference
         "RFC XXXX: YANG Data Model for QoS.";
     }

     /************************************************
      * Classification types
      ************************************************/

     identity internal-loss-priority  {
        base qos-types:filter-type;
        description
           "Internal loss priority filter type.";
     }

     grouping loss-priority {
       list loss-priority {
         key "loss-priority";
         description
           "list of loss-priorities";
         leaf loss-priority {
           type enumeration {
             enum high {
              description "High loss Priority.";
             }
             enum medium-high {
              description "Medium to high loss priority.";
             }
             enum medium-low {
```

```
             description "Medium to low loss priority.";
            }
            enum low {

             description "Low loss priority.";
            }
          }
          description
            "Loss priority.";
        }
      }
      description
        "Filter containing list of loss priorities";
    }

    augment "/traffic-policy:policies" +
            "/traffic-policy:policy" +
            "/traffic-policy:classifier" +
            "/traffic-policy:inline/traffic-policy:filter" +
            "/diffserv:filter-params" {
      case internal-loss-priority {
          uses loss-priority;
        description
          "Filter Type Internal-loss-priority";
      }
      description
        "Augments Diffserv Classifier with vendor " +
        "specific types.";
    }

    /***********************************************
     * Actions
     ***********************************************/

    identity mark-loss-priority {
      base qos-types:action-type;
      description
        "Mark loss-priority action type.";
    }

    grouping mark-loss-priority {
      container mark-loss-priority {
        leaf loss-priority {
          type enumeration {
            enum high {
             description "High loss Priority.";
            }
            enum medium-high {
```

```
              description "Medium to high loss priority.";
             }
             enum medium-low {
              description "Medium to low loss priority.";
             }
             enum low {
              description "Low loss priority.";
             }
           }
           description
             "Loss priority.";
       }
       description
         "Mark loss priority container.";
     }
     description
       "Mark loss priority grouping.";
   }

   augment "/traffic-policy:policies" +
           "/traffic-policy:policy" +
           "/traffic-policy:classifier" +
           "/traffic-policy:action" +
           "/diffserv:action-params" {
     case traffic-group-marking {
       uses qos-action:traffic-group-marking;
       description
         "Mark traffic group in the packet.";
     }
     case mark-loss-priority {
         uses mark-loss-priority;
       description
         "Mark loss priority in the packet.";
     }
     case meter-reference {
         uses qos-action:meter-reference;
       description
         "Assign a meter as an action.";
     }
     case discard {
         uses qos-action:discard;
       description
         "Discard action.";
     }
     case qos-count {
         uses qos-action:qos-count;
       description
         "Traffic-count action - explicit qos-count configuration.";
```

```
      }

      description
        "Augments common diffserv policy actions";
    }

    augment "/qos-action:meters" +
            "/qos-action:meter" +
            "/qos-action:meter-types" +
            "/qos-action:single-rate-three-color-meter-type" +
            "/qos-action:single-rate-three-color-meter" {
     leaf one-rate-color-aware {
      type boolean;
      description
        "This defines if the meter is color-aware.";
      }
      description
        "Augmentation of color-aware flag.";
    }
    augment "/qos-action:meters" +
            "/qos-action:meter" +
            "/qos-action:meter-types" +
            "/qos-action:two-rate-three-color-meter-type" +
            "/qos-action:two-rate-three-color-meter" {
     leaf two-rate-color-aware {
      type boolean;
      description
        "This defines if the meter is color-aware.";
      }
      description
        "Augmentation of color-aware flag.";
    }

  }
  <CODE ENDS>
```

Appendix C. Configuration examples

This section describes several examples of how the models defined in this draft can be used for configuration.

C.1. Configuration example for QoS Classifier

```
   <!--
       This example shows a QoS classifier configuration.
       This classifier will match on any packet which has
       dscp value not in the range of 11-13 and source-port
       value which is in the range of either 10000-10300 or
       17540-19800. In other words, packets with dscp value
       0-10 or 14-63 and source-port value 10000-10300 or
       17540-19800
   -->
   <?xml version="1.0" encoding="UTF-8"?>
   <classifiers
       xmlns="urn:ietf:params:xml:ns:yang:ietf-traffic-policy"
       xmlns:qt="urn:ietf:params:xml:ns:yang:ietf-qos-types">
     <classifier>
       <name>my-classifier</name>
       <filter-operation>match-all</filter-operation>
       <filter>
         <type>qt:dscp</type>
         <logical-not>true</logical-not>
         <dscp
           xmlns="urn:ietf:params:xml:ns:yang:ietf-diffserv">
           <min>11</min>
           <max>13</max>
         </dscp>
       </filter>
       <filter>
         <type>qt:source-port</type>
         <logical-not>false</logical-not>
           <source-port
             xmlns="urn:ietf:params:xml:ns:yang:ietf-diffserv">
             <min>10000</min>
             <max>10300</max>
           </source-port>
           <source-port
             xmlns="urn:ietf:params:xml:ns:yang:ietf-diffserv">
             <min>17540</min>
             <max>19800</max>
           </source-port>
       </filter>
     </classifier>
   </classifiers>
```

Figure 21: Configuration example for QoS Classifier

C.2. Configuration example for QoS Policy

```
   <!--
       This example shows a QoS policy configuration.
```

```
   -->

   <?xml version="1.0" encoding="UTF-8"?>
   <policies
        xmlns="urn:ietf:params:xml:ns:yang:ietf-traffic-policy"
        xmlns:qt="urn:ietf:params:xml:ns:yang:ietf-qos-types">
     <policy>
       <name>my-policy</name>
       <type>qt:diffserv-policy-type</type>
       <classifier>
         <name>my-classifier</name>
         <inline>
           <filter>
                 <type>qt:dscp</type>
                 <logical-not>false</logical-not>
               <dscp
                 xmlns="urn:ietf:params:xml:ns:yang:ietf-diffserv">
                 <min>21</min>
                 <max>22</max>
               </dscp>
           </filter>
         </inline>
         <action>
           <type>qt:dscp-marking</type>
           <dscp
             xmlns="urn:ietf:params:xml:ns:yang:ietf-diffserv">
             <dscp>23</dscp>
           </dscp>
         </action>
       </classifier>
     </policy>
   </policies>
```

Figure 12: Configuration example for QoS Policy

C.3. Configuration example for QoS Policing

Authors' Addresses

A. Choudhary
Cisco Systems

M. Jethanandani
Kloud Services

E. Aries
Juniper Networks

I. Chen
The MITRE Corporation