Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 18, 2018

A. Choudhary
M. Jethanandani
Cisco Systems
N. Strahle
E. Aries
Juniper Networks
I. Chen
Jabil

December 15, 2017

YANG Model for QoS
draft-asechoud-rtgwg-qos-model-03

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on June 18, 2018.

Copyright Notice

Table of Contents

1. Introduction

This document defines a base YANG [RFC6020] data module for Quality of Service (QoS) configuration parameters. Differentiated Services (DiffServ) module is an augmentation of the base QoS model. Remote Procedure Calls (RPC) or notification definition is currently not part of this document and will be added later if necessary. QoS base modules define a basic building blocks to define a classifier, policy, action and target. The base modules have been augmented to include packet match fields and action parameters to define the DiffServ module. It is left up to individual vendors to stitch some of the actions like queues, random-detect (RED) and vendor specific parameters of the DiffServ policy definitions. Designing the module in this manner allows for a very flexible and extensible module that should fit in with most of the vendor requirements. The DiffServ model is based on DiffServ architecture, and various references have been made to available standard architecture documents.

DiffServ is a preferred approach for network service providers to offer services to different customers based on their network Quality-of-Service (QoS) objectives. The traffic streams are differentiated

based on DiffServ Code Points (DSCP) carried in the IP header of each packet. The DSCP markings are applied by upstream node or by the edge router on entry to the DiffServ network.

2. Terminology

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

3. QoS Model Design

A classifier consists of packets which may be grouped when a logical set of rules are applied on different packet header fields. The grouping may be based on different values or range of values of same packet header field, presence or absence of some values or range of values of a packet field or a combination thereof. The QoS classifier is defined in the ietf-qos-classifier module.

A classifier entry contains one or more packet conditioning functions. A packet conditioning function is typically based on direction of traffic and may drop, mark or delay network packets. A set of classifier entries with corresponding conditioning functions when arranged in order of priority represents a QoS policy. A QoS policy may contain one or more classifier entries. These are defined in ietf-qos-policy module.

Actions are configured in line with respect to the policy module. These include marking, dropping or shaping. Actions are defined in the ietf-qos-action module.

A meter qualifies if the traffic arrival rate is based on agreed upon rate and variability. A meter is modeled based on commonly used alogrithms in industry, Single Rate Tri Color Marking (srTCM) [RFC2697] meter, Two Rate Tri Color Marking (trTCM) [RFC2698] meter, and Single Rate Two Color Marking meter. Different vendors can extend it with other types of meters as well.

4. DiffServ Model Design

DiffServ architecture [RFC3289] and [RFC2475] describe 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).

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 one or more header fields. In the ietf-diffserv module, this is realized by augmenting the QoS classification 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.

5. Modules Tree Structure

This document defines five YANG modules - four QoS base modules and one DiffServ module.

ietf-qos-classifier consists of classifier entries identified by a classifier entry name. Each entry MAY contain a list of filter entries. When no filter entry is present in a classifier entry, it matches all traffic.

```
module: ietf-qos-classifier
   +--rw classifiers
      +--rw classifier-entry* [classifier-entry-name]
         +--rw classifier-entry-name                string
         +--rw classifier-entry-descr?              string
         +--rw classifier-entry-filter-operation?   identityref
         +--rw filter-entry* [filter-type filter-logical-not]
            +--rw filter-type           identityref
            +--rw filter-logical-not    boolean
```

An ietf-qos-policy module contains list of policy objects identified by a policy name and policy type which MUST be provided. 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.

```
module: ietf-qos-policy
   +--rw policies
      +--rw policy-entry* [policy-name policy-type]
         +--rw policy-name         string
         +--rw policy-type         identityref
         +--rw policy-descr?       string
         +--rw classifier-entry* [classifier-entry-name]
            +--rw classifier-entry-name           string
            +--rw classifier-entry-inline?        boolean
            +--rw classifier-entry-filter-oper?   identityref
            +--rw filter-entry* [filter-type filter-logical-not]
                                {policy-inline-classifier-config}?
            |  +--rw filter-type           identityref
            |  +--rw filter-logical-not    boolean
            +--rw classifier-action-entry-cfg* [action-type]
               +--rw action-type    identityref
               +--rw (action-cfg-params)?
```

ietf-qos-action module contains grouping of set of QoS actions. These include metering, marking, dropping and shaping. Marking sets DiffServ codepoint value in the classified packet. Color-aware and Color-blind meters are augmented by vendor specific modules based on the parameters defined in action module.

```
module: ietf-qos-action
   +--rw meter-template
      +--rw meter-entry* [meter-name] {meter-template-support}?
         +--rw meter-name                  string
         +--rw (meter-type)?
            +--:(one-rate-two-color-meter-type)
            |  +--rw one-rate-two-color-meter
            |     +--rw meter-rate?       uint64
            |     +--rw meter-burst?      uint64
            |     +--rw conform-action
            |     |  +--rw meter-action-params* [meter-action-type]
            |     |     +--rw meter-action-type    identityref
            |     |     +--rw (meter-action-val)?
            |     +--rw exceed-action
            |        +--rw meter-action-params* [meter-action-type]
            |           +--rw meter-action-type    identityref
            |           +--rw (meter-action-val)?
            +--:(one-rate-tri-color-meter-type)
            |  +--rw one-rate-tri-color-meter
            |     +--rw committed-rate?    uint64
            |     +--rw committed-burst?   uint64
            |     +--rw excess-burst?      uint64
            |     +--rw conform-action
```

```
                |     |  +--rw meter-action-params* [meter-action-type]
                |     |     +--rw meter-action-type    identityref
                |     |     +--rw (meter-action-val)?
                |     +--rw exceed-action
                |     |  +--rw meter-action-params* [meter-action-type]
                |     |     +--rw meter-action-type    identityref
                |     |     +--rw (meter-action-val)?
                |     +--rw violate-action
                |        +--rw meter-action-params* [meter-action-type]
                |           +--rw meter-action-type    identityref
                |           +--rw (meter-action-val)?
                +--:(two-rate-tri-color-meter-type)
                   +--rw two-rate-tri-color-meter
                      +--rw committed-rate?    uint64
                      +--rw committed-burst?   uint64
                      +--rw peak-rate?         uint64
                      +--rw peak-burst?        uint64
                      +--rw conform-action
                      |  +--rw meter-action-params* [meter-action-type]
                      |     +--rw meter-action-type    identityref
                      |     +--rw (meter-action-val)?
                      +--rw exceed-action
                      |  +--rw meter-action-params* [meter-action-type]
                      |     +--rw meter-action-type    identityref
                      |     +--rw (meter-action-val)?
                      +--rw violate-action
                         +--rw meter-action-params* [meter-action-type]
                            +--rw meter-action-type    identityref
                            +--rw (meter-action-val)?
```

ietf-qos-target module contains reference of qos-policy and augments ietf-interfaces [RFC7223] module. A single policy of a particular policy-type can be applied on an interface in each direction of traffic. Policy-type is of type identity and is populated in a vendor specific manner. This way it provides greater flexibility for each vendor to define different policy types each with its own capabilities and restrictions.

Classifier, metering and queuing counters are associated with a target.

```
module: ietf-qos-target
augment /if:interfaces/if:interface:
   +--rw qos-target-entry* [direction policy-type]
      +--rw direction      identityref
      +--rw policy-type    identityref
      +--rw policy-name    string
```

Diffserv module augments QoS classifier module. Many of the YANG types defined in [RFC6991] are represented as leafs in the classifier module.

Metering and marking actions are realized by augmenting the QoS policy-module. Any queuing, AQM and scheduling actions are part of vendor specific augmentation. Statistics are realized by augmenting the QoS target module.

```
module: ietf-diffserv
augment "/classifier:classifiers/classifier:classifier-entry" +
        "/classifier:filter-entry":
   +--rw (filter-param)?
      +--:(dscp)
      |  +--rw dscp-cfg* [dscp-min dscp-max]
      |     +--rw dscp-min    inet:dscp
      |     +--rw dscp-max    inet:dscp
      +--:(source-ip-address)
      |  +--rw source-ip-address-cfg* [source-ip-addr]
      |     +--rw source-ip-addr    inet:ip-prefix
      +--:(destination-ip-address)
      |  +--rw destination-ip-address-cfg* [destination-ip-addr]
      |     +--rw destination-ip-addr    inet:ip-prefix
      +--:(source-port)
      |  +--rw source-port-cfg* [source-port-min source-port-max]
      |     +--rw source-port-min    inet:port-number
      |     +--rw source-port-max    inet:port-number
      +--:(destination-port)
      |  +--rw destination-port-cfg*
                      [destination-port-min destination-port-max]
      |     +--rw destination-port-min    inet:port-number
      |     +--rw destination-port-max    inet:port-number
      +--:(protocol)
         +--rw protocol-cfg* [protocol-min protocol-max]
            +--rw protocol-min    uint8
            +--rw protocol-max    uint8
augment "/policy:policies/policy:policy-entry" +
        "/policy:classifier-entry/policy:filter-entry":
   +--rw (filter-params)?
      +--:(dscp)
      |  +--rw dscp-cfg* [dscp-min dscp-max]
      |     +--rw dscp-min    inet:dscp
      |     +--rw dscp-max    inet:dscp
      +--:(source-ip-address)
      |  +--rw source-ip-address-cfg* [source-ip-addr]
      |     +--rw source-ip-addr    inet:ip-prefix
      +--:(destination-ip-address)
      |  +--rw destination-ip-address-cfg* [destination-ip-addr]
```

```
         |     +--rw destination-ip-addr    inet:ip-prefix
         +--:(source-port)
         |  +--rw source-port-cfg* [source-port-min source-port-max]
         |     +--rw source-port-min    inet:port-number
         |     +--rw source-port-max    inet:port-number
         +--:(destination-port)
         |  +--rw destination-port-cfg*
                        [destination-port-min destination-port-max]
         |     +--rw destination-port-min    inet:port-number
         |     +--rw destination-port-max    inet:port-number
         +--:(protocol)
            +--rw protocol-cfg* [protocol-min protocol-max]
               +--rw protocol-min    uint8
               +--rw protocol-max    uint8
   augment "/policy:policies/policy:policy-entry" +
           "/policy:classifier-entry" +
           "/policy:classifier-action-entry-cfg":
      +--rw (action-cfg-params)?
         +--:(dscp-marking)
            +--rw dscp-cfg
               +--rw dscp?   inet:dscp
```

6. Modules

6.1. IETF-QOS-CLASSIFIER

```
   <CODE BEGINS>file "ietf-qos-classifier@2016-03-03.yang"
   module ietf-qos-classifier {
     yang-version 1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-qos-classifier";
     prefix classifier;
     import ietf-inet-types {
       prefix inet;
     }
     organization "IETF RTG (Routing Area) Working Group";
     contact
       "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
        WG List:  <mailto:rtgwg@ietf.org>
        WG Chair: Chris Bowers
                  <mailto:cbowers@juniper.net>
        WG Chair: Jeff Tantsura
                  <mailto:jefftant.ietf@gmail.com>
        Editor:   Aseem Choudhary
                  <mailto:asechoud@cisco.com>
        Editor:   Mahesh Jethanandani
                  <mailto:mjethanandani@gmail.com>
        Editor:   Norm Strahle
                  <mailto:nstrahle@juniper.net>";
```

```
     description
       "This module contains a collection of YANG definitions for
        configuring qos specification implementations.
        Copyright (c) 2014 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 2016-03-03 {
       description
         "Latest revision of qos base classifier module";
       reference "RFC XXXX";
     }
     feature policy-inline-classifier-config {
       description
         " This feature allows classifier configuration
           directly under policy.";
     }
     identity filter-type {
       description
         "This is identity of base filter-type";
     }
     identity dscp {
       base filter-type;
       description
         "Differentiated services code point filter-type";
     }
     identity source-ip-address {
       base filter-type;
       description
         "source ipv4 and ipv6 address filter-type";
     }
     identity destination-ip-address {
       base filter-type;
       description
         "destination ipv4 and ipv6 address filter-type";
     }
     identity source-port {
       base filter-type;
       description
         "source port filter-type";
     }
     identity destination-port {
```

```
      base filter-type;
      description
        "destination port filter-type";
    }
    identity protocol {
      base filter-type;
      description
        "protocol type filter-type";
    }
    identity classifier-entry-filter-operation-type {
      description
        "Classifier entry filter logical operation";
    }
    identity match-any-filter {
      base classifier-entry-filter-operation-type;
      description
        "Classifier entry filter logical OR operation";
    }
    identity match-all-filter {
      base classifier-entry-filter-operation-type;
      description
        "Classifier entry filter logical AND operation";
    }
    grouping dscp-cfg {
      list dscp-cfg {
        key "dscp-min dscp-max";
        description
          "list of dscp ranges";
        leaf dscp-min {
          type inet:dscp;
          description
            "Minimum value of dscp min-max range";
        }
        leaf dscp-max {
          type inet:dscp;
          description
            "maximum value of dscp min-max range";
        }
      }
      description
        "Filter grouping containing list of dscp ranges";
    }
    grouping source-ip-address-cfg {
      list source-ip-address-cfg {
        key "source-ip-addr";
        description
          "list of source ipv4 or ipv6 address";
        leaf source-ip-addr {
```

```
            type inet:ip-prefix;
            description
              "source ipv4 or ipv6 prefix";
          }
        }
        description
          "Filter grouping containing list of source ip addresses";
      }
      grouping destination-ip-address-cfg {
        list destination-ip-address-cfg {
          key "destination-ip-addr";
          description
            "list of destination ipv4 or ipv6 address";
          leaf destination-ip-addr {
            type inet:ip-prefix;
            description
              "destination ipv4 or ipv6 prefix";
          }
        }
        description
          "Filter grouping containing list of destination ip address";
      }
      grouping source-port-cfg {
        list source-port-cfg {
          key "source-port-min source-port-max";
          description
            "list of ranges of source port";
          leaf source-port-min {
            type inet:port-number;
            description
              "minimum value of source port range";
          }
          leaf source-port-max {
            type inet:port-number;
            description
              "maximum value of source port range";
          }
        }
        description
          "Filter grouping containing list of source port ranges";
      }
      grouping destination-port-cfg {
        list destination-port-cfg {
          key "destination-port-min destination-port-max";
          description
            "list of ranges of destination port";
          leaf destination-port-min {
            type inet:port-number;
```

```
            description
              "minimum value of destination port range";
          }
          leaf destination-port-max {
            type inet:port-number;
            description
              "maximum value of destination port range";
          }
        }
        description
          "Filter grouping containing list of destination port ranges";
      }
      grouping protocol-cfg {
        list protocol-cfg {
          key "protocol-min protocol-max";
          description
            "list of ranges of protocol values";
          leaf protocol-min {
            type uint8 {
              range "0..255";
            }
            description
              "minimum value of protocol range";
          }
          leaf protocol-max {
            type uint8 {
              range "0..255";
            }
            description
              "maximum value of protocol range";
          }
        }
        description
          "Filter grouping containing list of Protocol ranges";
      }
      grouping filters {
        description
          "Filters types in a Classifier entry";
        leaf filter-type {
          type identityref {
            base filter-type;
          }
          description
            "This leaf defines type of the filter";
        }
        leaf filter-logical-not {
          type boolean;
          description
```

```
            "
             This is logical-not operator for a filter. When true, it
             indicates filter looks for absence of a pattern defined
             by the filter
            ";
        }
      }
      grouping classifier-entry-generic-attr {
        description
          "
           Classifier generic attributes like name,
           description, operation type
          ";
        leaf classifier-entry-name {
          type string;
          description
            "classifier entry name";
        }
        leaf classifier-entry-descr {
          type string;
          description
            "classifier entry description statement";
        }
        leaf classifier-entry-filter-operation {
          type identityref {
            base classifier-entry-filter-operation-type;
          }
          default "match-any-filter";
          description
            "Filters are applicable as match-any or match-all filters";
        }
      }
      grouping classifier-entry-inline-attr {
        description
          "attributes of inline classifier in a policy";
        leaf classifier-entry-inline {
          type boolean;
          default "false";
          description
            "Indication of inline classifier entry";
        }
        leaf classifier-entry-filter-oper {
          type identityref {
            base classifier-entry-filter-operation-type;
          }
          default "match-all-filter";
          description
            "Filters are applicable as match-any or match-all filters";
```

```
      }
      list filter-entry {
        if-feature policy-inline-classifier-config;
        must " ../classifier-entry-inline = 'true' " {
          description
            "For inline filter configuration, inline attribute" +
            "must be true";
        }
        key "filter-type filter-logical-not";
        uses filters;
        description
          "Filters configured inline in a policy";
      }
    }
    container classifiers {
      description
        "list of classifier entry";
      list classifier-entry {
        key "classifier-entry-name";
        description
          "each classifier entry contains a list of filters";
        uses classifier-entry-generic-attr;
        list filter-entry {
          key "filter-type filter-logical-not";
          uses filters;
          description
            "Filter entry configuration";
        }
      }
    }
  }
  <CODE ENDS>
```

6.2. IETF-QOS-POLICY

```
  <CODE BEGINS>file "ietf-qos-policy@2016-03-03.yang"
  module ietf-qos-policy {
    yang-version 1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-qos-policy";
    prefix policy;
    import ietf-qos-classifier {
      prefix classifier;
    }
    organization "IETF RTG (Routing Area) Working Group";
    contact
      "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
       WG List:  <mailto:rtgwg@ietf.org>
       WG Chair: Chris Bowers
```

```
                <mailto:cbowers@juniper.net>
        WG Chair: Jeff Tantsura
                <mailto:jefftant.ietf@gmail.com>
        Editor:   Aseem Choudhary
                <mailto:asechoud@cisco.com>
        Editor:   Mahesh Jethanandani
                <mailto:mjethanandani@gmail.com>
        Editor:   Norm Strahle
                <mailto:nstrahle@juniper.net>";
    description
      "This module contains a collection of YANG definitions for
       configuring qos specification implementations.
       Copyright (c) 2014 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 2016-03-03 {
      description
        "Latest revision of qos policy";
      reference "RFC XXXX";
    }
    identity policy-type {
      description
        "This base identity type defines policy-types";
    }
    grouping policy-generic-attr {
      description
        "Policy Attributes";
      leaf policy-name {
        type string;
        description
          "policy name";
      }
      leaf policy-type {
        type identityref {
          base policy-type;
        }
        description
          "policy type";
      }
      leaf policy-descr {
        type string;
```

```
        description
          "policy description";
      }
    }
    identity action-type {
      description
        "This base identity type defines action-types";
    }
    grouping classifier-action-entry-cfg {
      description
        "List of Configuration of classifier & associated actions";
      list classifier-action-entry-cfg {
        key "action-type";
        ordered-by user;
        description
          "Configuration of classifier & associated actions";
        leaf action-type {
          type identityref {
            base action-type;
          }
          description
            "This defines action type ";
        }
        choice action-cfg-params {
          description
            "Choice of action types";
        }
      }
    }
    container policies {
      description
        "list of policy templates";
      list policy-entry {
        key "policy-name policy-type";
        description
          "policy template";
        uses policy-generic-attr;
        list classifier-entry {
          key "classifier-entry-name";
          ordered-by user;
          description
            "Classifier entry configuration in a policy";
          leaf classifier-entry-name {
            type string;
            description
              "classifier entry name";
          }
          uses classifier:classifier-entry-inline-attr;
```

```
          uses classifier-action-entry-cfg;
        }
      }
    }
  }
  <CODE ENDS>
```

6.3. IETF-QOS-ACTION

```
  <CODE BEGINS>file "ietf-qos-action@2016-06-15.yang"
  module ietf-qos-action {
    namespace "urn:ietf:params:xml:ns:yang:ietf-qos-action";
    prefix action;
    import ietf-inet-types {
      prefix inet;
    }
    import ietf-qos-policy {
      prefix policy;
    }
    organization "IETF RTG (Routing Area) Working Group";
    contact
      "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
       WG List:  <mailto:rtgwg@ietf.org>
       WG Chair: Chris Bowers
                 <mailto:cbowers@juniper.net>
       WG Chair: Jeff Tantsura
                 <mailto:jefftant.ietf@gmail.com>
       Editor:   Aseem Choudhary
                 <mailto:asechoud@cisco.com>
       Editor:   Mahesh Jethanandani
                 <mailto:mjethanandani@gmail.com>
       Editor:   Norm Strahle
                 <mailto:nstrahle@juniper.net>";
    description
      "This module contains a collection of YANG definitions for
       configuring qos specification implementations.
       Copyright (c) 2014 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 2016-06-15 {
      description
```

```
        "Latest revision for qos actions";
      reference "RFC XXXX";
    }
    feature meter-template-support {
      description
        " This feature allows support of meter-template.";
    }

    identity rate-unit-type {
      description
        "base rate-unit type";
    }
    identity bits-per-second {
      base rate-unit-type;
      description
        "bits per second identity";
    }
    identity kilo-bits-per-second {
      base rate-unit-type;
      description
        "kilo bits per second identity";
    }
    identity mega-bits-per-second {
      base rate-unit-type;
      description
        "mega bits per second identity";
    }
    identity giga-bits-per-second {
      base rate-unit-type;
      description
        "mega bits per second identity";
    }
    identity percent {
      base rate-unit-type;
      description
        "percentage";
    }

    identity dscp-marking {
      base policy:action-type;
      description
        "dscp marking action type";
    }
    identity meter-inline {
      base policy:action-type;
      description
        "meter-inline action type";
    }
```

```
identity meter-reference {
  base policy:action-type;
  description
    "meter reference action type";
}
identity min-rate {
  base policy:action-type;
  description
    "min-rate action type";
}
identity max-rate {
  base policy:action-type;
  description
    "max-rate action type";
}
identity queue {
  base policy:action-type;
  description
    "queue action type";
}
identity schedular {
  base policy:action-type;
  description
    "schedular action type";
}
identity discard {
  base policy:action-type;
  description
    "discard action type";
}
identity child-policy {
  base policy:action-type;
  description
    "child-policy action type";
}
identity count {
  base policy:action-type;
  description
    "discard action type";
}

identity meter-type {
  description
    "This base identity type defines meter types";
}
identity one-rate-two-color-meter-type {
  base meter-type;
  description
```

```
        "one rate two color meter type";
    }
    identity one-rate-tri-color-meter-type {
      base meter-type;
      description
        "one rate three color meter type";
    }
    identity two-rate-tri-color-meter-type {
      base meter-type;
      description
        "two rate three color meter action type";
    }

    identity drop-type {
      description
        "drop algorithm";
    }
    identity tail-drop {
      base drop-type;
      description
        "tail drop algorithm";
    }
    identity random-detect {
      base drop-type;
      description
        "random detect algorithm";
    }

    identity meter-action-type {
      description
        "action type in a meter";
    }
    identity meter-action-drop {
      base meter-action-type;
      description
        "drop action type in a meter";
    }
    identity meter-action-mark-dscp {
      base meter-action-type;
      description
        "dscp mark action type in a meter";
    }

    grouping rate-value-unit {
      leaf rate-value {
        type uint64;
        description
          "rate value";
```

```
      }
      leaf rate-unit {
        type identityref {
          base rate-unit-type;
        }
        description
          "rate unit";
      }
      description
        "rate value and unit grouping";
    }
    grouping burst {
      description
        "burst size or interval configuration";
      choice burst-type {
        case size {
          leaf burst-size {
            type uint64;
            units "bytes";
            description
              "burst size";
          }
        }
        case interval {
          leaf burst-interval {
            type uint64;
            units "microsecond";
            description
              "burst interval";
          }
        }
        description
          "Choice of burst type";
      }
    }

    grouping threshold {
      description
        "Threshold Parameters";
      container threshold {
        description
          "threshold";
        choice threshold-type {
          case size {
            leaf threshold-size {
              type uint64;
              units "bytes";
              description
```

```
                "Threshold size";
            }
          }
          case interval {
            leaf threshold-interval {
              type uint64;
              units "microsecond";
              description
                "Threshold interval";
            }
          }
          description
            "Choice of threshold type";
        }
      }
    }

    grouping drop {
      container drop-cfg {
        leaf drop-action {
          type empty;
          description
            "always drop algorithm";
        }
        description
          "the drop action";
      }
      description
        "always drop grouping";
    }

    grouping queuelimit {
      container qlimit-thresh {
        uses threshold;
        description
          "the queue limit";
      }
      description
        "the queue limit beyond which queue will not hold any packet";
    }

    grouping meter-action-params {
      description
        "meter action parameters";
      list meter-action-params {
        key "meter-action-type";
        ordered-by user;
        description
```

```
          "Configuration of basic-meter & associated actions";
        leaf meter-action-type {
          type identityref {
            base meter-action-type;
          }
          description
            "meter action type";
        }
        choice meter-action-val {
          description
            " meter action based on choice of meter action type";
        }
      }
    }

    grouping one-rate-two-color-meter {
      container one-rate-two-color-meter {
        description
          "single rate two color marker meter";
        leaf meter-rate {
          type uint64;
          units "bits-per-second";
          description
            "meter rate";
        }
        leaf meter-burst {
          type uint64;
          units "byes";
          description
            "burst size";
        }
        container conform-action {
          uses meter-action-params;
          description
            "conform action";
        }
        container exceed-action {
          uses meter-action-params;
          description
            "exceed action";
        }
      }
      description
        "single rate two color marker meter attributes";
    }

    grouping one-rate-tri-color-meter {
      container one-rate-tri-color-meter {
```

```
        description
          "single rate three color meter";
      leaf committed-rate {
        type uint64;
        units "bits-per-second";
        description
          "meter rate";
      }
      leaf committed-burst {
        type uint64;
        units "byes";
        description
          "commited burst size";
      }
      leaf excess-burst {
        type uint64;
        units "byes";
        description
          "excess burst size";
      }
      container conform-action {
        uses meter-action-params;
        description
          "conform, or green action";
      }
      container exceed-action {
        uses meter-action-params;
        description
          "exceed, or yellow action";
      }
      container violate-action {
        uses meter-action-params;
        description
          "violate, or red action";
      }
    }
      description
        "one-rate-tri-color-meter attributes";
    }

    grouping two-rate-tri-color-meter {
      container two-rate-tri-color-meter {
        description
          "two rate three color meter";
      leaf committed-rate {
        type uint64;
        units "bits-per-second";
        description
```

```
            "meter rate";
        }
        leaf committed-burst {
          type uint64;
          units "byes";
          description
            "commited burst size";
        }
        leaf peak-rate {
          type uint64;
          units "bits-per-second";
          description
            "meter rate";
        }
        leaf peak-burst {
          type uint64;
          units "byes";
          description
            "commited burst size";
        }
        container conform-action {
          uses meter-action-params;
          description
            "conform, or green action";
        }
        container exceed-action {
          uses meter-action-params;
          description
            "exceed, or yellow action";
        }
        container violate-action {
          uses meter-action-params;
          description
            "exceed, or red action";
        }
      }
        description
          "two-rate-tri-color-meter attributes";
      }

      grouping meter {
        choice meter-type {
          case one-rate-two-color-meter-type {
            uses one-rate-two-color-meter;
            description
              "basic meter";
          }
          case one-rate-tri-color-meter-type {
```

```
          uses one-rate-tri-color-meter;
          description
            "one rate tri-color meter";
        }
        case two-rate-tri-color-meter-type {
          uses two-rate-tri-color-meter;
          description
            "two rate tri-color meter";
        }
        description
          " meter action based on choice of meter action type";
      }
      description
        "meter attributes";
    }

    container meter-template {
      description
        "list of meter templates";
      list meter-entry {
        if-feature meter-template-support;
        key "meter-name";
        description
          "meter entry template";
        leaf meter-name {
          type string;
          description
            "meter identifier";
        }
        uses meter;
      }
    }

    grouping meter-reference {
      container meter-reference-cfg {
        leaf meter-type {
          type identityref {
            base meter-type;
          }
          description
            "This leaf defines type of the filter";
        }
        description
          "meter reference";
      }
      description
        "meter reference";
    }
```

```
     grouping count {
       container count-cfg {
         leaf count-action {
           type empty;
           description
             "count action";
         }
         description
           "the count action";
       }
       description
         "the count action grouping";
     }

     grouping discard {
       container discard-cfg {
         leaf discard {
           type empty;
           description
             "discard action";
         }
         description
           "discard action";
       }
       description
         "discard grouping";
     }
     grouping priority {
       container priority-cfg {
         leaf priority-level {
           type uint8;
           description
             "priority level";
         }
         description
           "priority attributes";
       }
       description
         "priority attributes grouping";
     }
     grouping min-rate {
       container min-rate-cfg {
         uses rate-value-unit;
         description
           "min guaranteed bandwidth";
       }
       description
         "minimum rate grouping";
```

```
    }
    grouping dscp-marking {
      container dscp-cfg {
        leaf dscp {
          type inet:dscp;
          description
            "dscp marking";
        }
        description
          "dscp marking container";
      }
      description
        "dscp marking grouping";
    }
    grouping max-rate {
      container max-rate-cfg {
        uses rate-value-unit;
        uses burst;
        description
          "maximum rate attributes container";
      }
      description
        "maximum rate attributes";
    }
    grouping queue {
      container queue-cfg {
        uses priority;
        uses min-rate;
        uses max-rate;
        container algorithmic-drop-cfg {
          choice drop-algorithm {
            case tail-drop {
              container tail-drop-cfg {
                leaf tail-drop-alg {
                  type empty;
                  description
                   "tail drop algorithm";
                }
                description
                  "Tail Drop configuration container";
              }
              description
                "Tail Drop choice";
            }
            description
              "Choice of Drop Algorithm";
          }
          description
```

```
              "Algorithmic Drop configuration container";
          }
          description
            "Queue configuration container";
        }
        description
          "Queue grouping";
      }
      grouping schedular {
        container schedular-cfg {
          uses min-rate;
          uses max-rate;
          description
            "Schedular configuration container";
        }
        description
          "Schedular configuration grouping";
      }
    }
    <CODE ENDS>
```

6.4. IETF-QOS-TARGET

```
    <CODE BEGINS>file "ietf-qos-target@2017-12-12.yang"
    module ietf-qos-target {
      yang-version 1;
      namespace "urn:ietf:params:xml:ns:yang:ietf-qos-target";
      prefix target;
      import ietf-interfaces {
        prefix if;
      }
      import ietf-qos-policy {
        prefix policy;
      }
      organization "IETF RTG (Routing Area) Working Group";
      contact
        "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
         WG List:  <mailto:rtgwg@ietf.org>
         WG Chair: Chris Bowers
                   <mailto:cbowers@juniper.net>
         WG Chair: Jeff Tantsura
                   <mailto:jefftant.ietf@gmail.com>
         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 specification implementations.
        Copyright (c) 2014 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 2017-12-12 {
      description
        "Latest revision qos based policy applied to a target";
      reference "RFC XXXX";
    }
    identity direction {
      description
        "This is identity of traffic direction";
    }
    identity inbound {
      base direction;
      description
        "Direction of traffic coming into the network entry";
    }
    identity outbound {
      base direction;
      description
        "Direction of traffic going out of the network entry";
    }
    augment "/if:interfaces/if:interface" {
      description
        "Augments Diffserv Target Entry to Interface module";
      list qos-target-entry {
        key "direction policy-type";
        description
          "policy target for inbound or outbound direction";
        leaf direction {
          type identityref {
            base direction;
          }
          description
            "Direction fo the traffic flow either inbound or outbound";
        }
        leaf policy-type {
          type identityref {
            base policy:policy-type;
          }
```

```
            description
              "Policy entry type";
          }
          leaf policy-name {
            type string;
            mandatory true;
            description
              "Policy entry name";
          }
        }
      }
    }
    <CODE ENDS>
```

6.5. IETF-DIFFSERV

```
    <CODE BEGINS>file "ietf-diffserv@2017-12-12.yang"
    module ietf-diffserv {
      yang-version 1;
      namespace "urn:ietf:params:xml:ns:yang:ietf-diffserv";
      prefix diffserv;

      import ietf-qos-classifier {
        prefix classifier;
      }
      import ietf-qos-policy {
        prefix policy;
      }
      import ietf-qos-action {
        prefix action;
      }

      organization "IETF RTG (Routing Area) Working Group";
      contact
        "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
         WG List:  <mailto:rtgwg@ietf.org>
         WG Chair: Chris Bowers
                   <mailto:cbowers@juniper.net>
         WG Chair: Jeff Tantsura
                   <mailto:jefftant.ietf@gmail.com>
         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) 2014 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 2017-12-12 {
      description
        "Latest revision of diffserv based classifier";
      reference "RFC XXXX";
    }

    augment "/classifier:classifiers/classifier:classifier-entry" +
            "/classifier:filter-entry" {
      choice filter-param {
        description
          "Choice of filter types";
        case dscp {
          uses classifier:dscp-cfg;
          description
            "Filter containing list of dscp ranges";
        }
        case source-ip-address {
          uses classifier:source-ip-address-cfg;
          description
            "Filter containing list of source ip addresses";
        }
        case destination-ip-address {
          uses classifier:destination-ip-address-cfg;
          description
            "Filter containing list of destination ip address";
        }
        case source-port {
          uses classifier:source-port-cfg;
          description
            "Filter containing list of source-port ranges";
        }
        case destination-port {
          uses classifier:destination-port-cfg;
          description
            "Filter containing list of destination-port ranges";
        }
        case protocol {
          uses classifier:protocol-cfg;
```

```
            description
              "Filter Type Protocol";
          }
        }
        description
          "augments diffserv filters to qos classifier";
      }
      augment "/policy:policies/policy:policy-entry" +
              "/policy:classifier-entry/policy:filter-entry" {
        choice filter-params {
          description
            "Choice of action types";
          case dscp {
            uses classifier:dscp-cfg;
            description
              "Filter containing list of dscp ranges";
          }
          case source-ip-address {
            uses classifier:source-ip-address-cfg;
            description
              "Filter containing list of source ip addresses";
          }
          case destination-ip-address {
            uses classifier:destination-ip-address-cfg;
            description
              "Filter containing list of destination ip address";
          }
          case source-port {
            uses classifier:source-port-cfg;
            description
              "Filter containing list of source-port ranges";
          }
          case destination-port {
            uses classifier:destination-port-cfg;
            description
              "Filter containing list of destination-port ranges";
          }
          case protocol {
            uses classifier:protocol-cfg;
            description
              "Filter Type Protocol";
          }
        }
        description
          "Augments Diffserv Classifier with common filter types";
      }
      augment "/policy:policies/policy:policy-entry" +
              "/policy:classifier-entry/" +
```

```
            "policy:classifier-action-entry-cfg" {
       choice action-cfg-params {
         description
           "Choice of action types";
         case dscp-marking {
           uses action:dscp-marking;
         }
       }
       description
         "augments dscp-marking and meter to qos policy";
     }
   }
   <CODE ENDS>
```

7. Security Considerations

8. Acknowledgement

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>.

[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>.

[RFC3289] Baker, F., Chan, K., and A. Smith, "Management Information Base for the Differentiated Services Architecture", RFC 3289, DOI 10.17487/RFC3289, May 2002, <https://www.rfc-editor.org/info/rfc3289>.

[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>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, DOI 10.17487/RFC7223, May 2014, <https://www.rfc-editor.org/info/rfc7223>.

9.2. Informative References

[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>.

Appendix A. Company A, Company B and Company C examples

Company A, Company B and Company C 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. Queuing and metering counters are realized by augmenting of the QoS target module.

A.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

- support of hierarchial policy.

- use of inline actions in a policy

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

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

  import ietf-interfaces {
    prefix if;
  }
  import ietf-qos-classifier {
    prefix classifier;
  }
  import ietf-qos-policy {
    prefix policy;
```

```
        }
        import ietf-qos-action {
          prefix action;
        }
        import ietf-qos-target {
          prefix target;
        }
        import ietf-diffserv {
          prefix diffserv;
        }

        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 2016-06-15 {
          description
            "Initial revision for diffserv actions on network packets";
          reference
            "RFC 6020: YANG - A Data Modeling Language for the
                       Network Configuration Protocol (NETCONF)";
        }

        identity default-policy-type {
          base policy:policy-type;
          description
            "This defines default policy-type";
        }

        identity qos-group {
          base classifier:filter-type;
          description
            "qos-group filter-type";
        }

        grouping qos-group-cfg {
          list qos-group-cfg {
            key "qos-group-min qos-group-max";
            description
              "list of dscp ranges";
            leaf qos-group-min {
              type uint8;
              description
                "Minimum value of qos-group range";
            }
```

```
            leaf qos-group-max {
              type uint8;
              description
                "maximum value of qos-group range";
            }
          }
          description
            "Filter containing list of qos-group ranges";
        }

        grouping wred-threshold {
          container wred-min-thresh {
            uses action:threshold;
            description
              "Minimum threshold";
          }
          container wred-max-thresh {
            uses action:threshold;
            description
              "Maximum threshold";
          }
          leaf mark-probability {
            type uint32 {
              range "1..1000";
            }
            description
              "Mark probability";
          }
          description
            "WRED threshold attributes";
        }

        grouping randomdetect {
          leaf exp-weighting-const {
            type uint32;
            description
              "Exponential weighting constant factor for wred profile";
          }
          uses wred-threshold;
          description
            "Random detect attributes";
        }

      /************************************************
       * Augmentation to Classifier Module
       ************************************************/

        augment "/classifier:classifiers/" +
```

```
               "classifier:classifier-entry/" +
               "classifier:filter-entry/diffserv:filter-param" {
          case qos-group {
            uses qos-group-cfg;
            description
              "Filter containing list of qos-group ranges.
               Qos-group represent packet metadata information
               in a device. ";
          }
          description
            "augmentation of classifier filters";
        }

      /***********************************************
       * Augmentation to Policy Module
       ***********************************************/

        augment "/policy:policies/policy:policy-entry/" +
                "policy:classifier-entry/" +
                "policy:classifier-action-entry-cfg/" +
                "policy:action-cfg-params" {
          case priority {
            uses action:priority;
          }
          case min-rate {
            uses action:min-rate;
          }
          case max-rate {
            uses action:max-rate;
          }
          case random-detect {
            uses randomdetect;
          }
          case meter-inline {
            uses action:meter;
          }
          case child-policy {
            leaf child-policy {
              type leafref {
                path "/policy:policies/policy:policy-entry/" +
                     "policy:policy-name";
              }
              description
                "Child Policy in the hierarchial configuration";
            }
          }
          description
            "Augment the actions to policy entry";
```

```
        }

        augment "/policy:policies" +
                "/policy:policy-entry" +
                "/policy:classifier-entry" +
                "/policy:classifier-action-entry-cfg" +
                "/policy:action-cfg-params" +
                "/example:meter-inline" +
                "/example:meter-type" +
                "/example:one-rate-two-color-meter-type" +
                "/example:one-rate-two-color-meter" +
                "/example:conform-action" +
                "/example:meter-action-params" +
                "/example:meter-action-val" {

          description
                "augment the one-rate-two-color meter conform
                 with actions";
          case meter-action-drop {
            description
                "meter drop";
                 uses action:drop;
          }
          case meter-action-mark-dscp {
            description
                "meter action dscp marking";
                 uses action:dscp-marking;
          }
        }
        augment "/policy:policies" +
                "/policy:policy-entry" +
                "/policy:classifier-entry" +
                "/policy:classifier-action-entry-cfg" +
                "/policy:action-cfg-params" +
                "/example:meter-inline" +
                "/example:meter-type" +
                "/example:one-rate-two-color-meter-type" +
                "/example:one-rate-two-color-meter" +
                "/example:exceed-action" +
                "/example:meter-action-params" +
                "/example:meter-action-val" {

          description
                "augment the one-rate-two-color meter exceed
                 with actions";
          case meter-action-drop {
            description
                "meter drop";
```

```
                uses action:drop;
          }
          case meter-action-mark-dscp {
            description
                "meter action dscp marking";
                uses action:dscp-marking;
          }
        }
        augment "/policy:policies" +
                "/policy:policy-entry" +
                "/policy:classifier-entry" +
                "/policy:classifier-action-entry-cfg" +
                "/policy:action-cfg-params" +
                "/example:meter-inline" +
                "/example:meter-type" +
                "/example:one-rate-tri-color-meter-type" +
                "/example:one-rate-tri-color-meter" +
                "/example:conform-action" +
                "/example:meter-action-params" +
                "/example:meter-action-val" {

          description
                "augment the one-rate-tri-color meter conform
                 with actions";
          case meter-action-drop {
            description
                "meter drop";
                uses action:drop;
          }
          case meter-action-mark-dscp {
            description
                "meter action dscp marking";
                uses action:dscp-marking;
          }
        }
        augment "/policy:policies" +
                "/policy:policy-entry" +
                "/policy:classifier-entry" +
                "/policy:classifier-action-entry-cfg" +
                "/policy:action-cfg-params" +
                "/example:meter-inline" +
                "/example:meter-type" +
                "/example:one-rate-tri-color-meter-type" +
                "/example:one-rate-tri-color-meter" +
                "/example:exceed-action" +
                "/example:meter-action-params" +
                "/example:meter-action-val" {
```

```
          description
                "augment the one-rate-tri-color meter exceed
                 with actions";
          case meter-action-drop {
            description
                "meter drop";
                 uses action:drop;
          }
          case meter-action-mark-dscp {
            description
                "meter action dscp marking";
                 uses action:dscp-marking;
          }
        }
        augment "/policy:policies" +
                "/policy:policy-entry" +
                "/policy:classifier-entry" +
                "/policy:classifier-action-entry-cfg" +
                "/policy:action-cfg-params" +
                "/example:meter-inline" +
                "/example:meter-type" +
                "/example:one-rate-tri-color-meter-type" +
                "/example:one-rate-tri-color-meter" +
                "/example:violate-action" +
                "/example:meter-action-params" +
                "/example:meter-action-val" {
          description
                "augment the one-rate-tri-color meter conform
                 with actions";
          case meter-action-drop {
            description
                "meter drop";
                 uses action:drop;
          }
          case meter-action-mark-dscp {
            description
                "meter action dscp marking";
                 uses action:dscp-marking;
          }
        }

        augment "/policy:policies" +
                "/policy:policy-entry" +
                "/policy:classifier-entry" +
                "/policy:classifier-action-entry-cfg" +
                "/policy:action-cfg-params" +
                "/example:meter-inline" +
                "/example:meter-type" +
```

```
                "/example:two-rate-tri-color-meter-type" +
                "/example:two-rate-tri-color-meter" +
                "/example:conform-action" +
                "/example:meter-action-params" +
                "/example:meter-action-val" {

          description
                "augment the one-rate-tri-color meter conform
                 with actions";
          case meter-action-drop {
            description
                "meter drop";
                 uses action:drop;
          }
          case meter-action-mark-dscp {
            description
                "meter action dscp marking";
                 uses action:dscp-marking;
          }
        }
        augment "/policy:policies" +
                "/policy:policy-entry" +
                "/policy:classifier-entry" +
                "/policy:classifier-action-entry-cfg" +
                "/policy:action-cfg-params" +
                "/example:meter-inline" +
                "/example:meter-type" +
                "/example:two-rate-tri-color-meter-type" +
                "/example:two-rate-tri-color-meter" +
                "/example:exceed-action" +
                "/example:meter-action-params" +
                "/example:meter-action-val" {

          description
                "augment the two-rate-tri-color meter exceed
                 with actions";
          case meter-action-drop {
            description
                "meter drop";
                 uses action:drop;
          }
          case meter-action-mark-dscp {
            description
                "meter action dscp marking";
                 uses action:dscp-marking;
          }
        }
        augment "/policy:policies" +
```

```
              "/policy:policy-entry" +
              "/policy:classifier-entry" +
              "/policy:classifier-action-entry-cfg" +
              "/policy:action-cfg-params" +
              "/example:meter-inline" +
              "/example:meter-type" +
              "/example:two-rate-tri-color-meter-type" +
              "/example:two-rate-tri-color-meter" +
              "/example:violate-action" +
              "/example:meter-action-params" +
              "/example:meter-action-val" {
        description
              "augment the two-rate-tri-color meter violate
               with actions";
        case meter-action-drop {
          description
              "meter drop";
               uses action:drop;
        }
        case meter-action-mark-dscp {
          description
              "meter action dscp marking";
               uses action:dscp-marking;
        }
      }
      augment "/policy:policies" +
              "/policy:policy-entry" +
              "/policy:classifier-entry" +
              "/policy:classifier-action-entry-cfg" +
              "/policy:action-cfg-params" +
              "/example:meter-inline" +
              "/example:meter-type" +
              "/example:one-rate-two-color-meter-type" +
              "/example:one-rate-two-color-meter" {
        description
              "augment the one-rate-two-color meter with" +
              "color classifiers";
          container conform-color {
            uses classifier:classifier-entry-generic-attr;
            description
              "conform color classifier container";
          }
          container exceed-color {
            uses classifier:classifier-entry-generic-attr;
            description
              "exceed color classifier container";
          }
      }
```

```
        augment "/policy:policies" +
                "/policy:policy-entry" +
                "/policy:classifier-entry" +
                "/policy:classifier-action-entry-cfg" +
                "/policy:action-cfg-params" +
                "/example:meter-inline" +
                "/example:meter-type" +
                "/example:one-rate-tri-color-meter-type" +
                "/example:one-rate-tri-color-meter" {
          description
                "augment the one-rate-tri-color meter with" +
                "color classifiers";
            container conform-color {
              uses classifier:classifier-entry-generic-attr;
              description
                "conform color classifier container";
            }
            container exceed-color {
              uses classifier:classifier-entry-generic-attr;
              description
                "exceed color classifier container";
            }
            container violate-color {
              uses classifier:classifier-entry-generic-attr;
              description
                "violate color classifier container";
            }
        }
        augment "/policy:policies" +
                "/policy:policy-entry" +
                "/policy:classifier-entry" +
                "/policy:classifier-action-entry-cfg" +
                "/policy:action-cfg-params" +
                "/example:meter-inline" +
                "/example:meter-type" +
                "/example:two-rate-tri-color-meter-type" +
                "/example:two-rate-tri-color-meter" {
          description
                "augment the two-rate-tri-color meter with" +
                "color classifiers";
            container conform-color {
              uses classifier:classifier-entry-generic-attr;
              description
                "conform color classifier container";
            }
            container exceed-color {
              uses classifier:classifier-entry-generic-attr;
              description
```

```
                "exceed color classifier container";
            }
            container violate-color {
              uses classifier:classifier-entry-generic-attr;
              description
                "violate color classifier container";
            }
        }

      /*************************************************
       * Augmentation to Target Module
       *************************************************/

        augment "/if:interfaces/if:interface/" +
                "target:qos-target-entry/" +
                "target:qos-target-classifier-statistics/" +
                "diffserv:diffserv-action-statistics" {
          uses target:queuing-stats;
          description
            "Augment the statistics to policy entry";
        }
        augment "/if:interfaces/if:interface/" +
                "target:qos-target-entry/" +
                "target:qos-target-classifier-statistics" {
          leaf relative-path {
            type string;
            description
              "Relative Path of the classifier entry in the
               hierarchial policy";
          }
          description
            "Augment the statistics to policy entry";
        }
      }
```

A.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 ietf-qos-action module

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

- use of internal meta data for classification and marking

```
module example-compb-diffserv-filter-policy {
  yang-version 1;
  namespace "urn:ietf:params:xml:ns:yang:" +
            "example-compb-diffserv-filter-policy";
  prefix compb-filter-policy;

  import ietf-qos-classifier {
    prefix classifier;
  }
  import ietf-qos-policy {
    prefix policy;
  }
  import ietf-qos-action {
    prefix 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) 2014 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 2015-04-07 {
    description
      "Latest revision of diffserv policy";
```

```
      reference "RFC XXXX";
    }


    /*
     * The policy must be of either type v4 or v6. Corresponding
     * address types must be used. Enforce with "must" statement?
     */
    identity v4-diffserv-policy-type {
      base policy:policy-type;
      description
        "This defines default policy-type";
    }

    identity v6-diffserv-policy-type {
      base policy:policy-type;
      description
        "This defines default policy-type";
    }

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

    identity forwarding-class {
        base classifier:filter-type;
        description
          "Forwarding class filter type";
    }

    identity internal-loss-priority  {
       base classifier:filter-type;
       description
          "Internal loss priority filter type";
    }

    grouping forwarding-class-cfg {
      list forwarding-class-cfg {
        key "forwarding-class";
        description
          "list of forwarding-classes";
        leaf forwarding-class {
          type string;
          description
            "Forwarding class name";
        }
      }
      description
```

```
        "Filter containing list of forwarding classes";
    }

    grouping loss-priority-cfg {
      list loss-priority-cfg {
        key "loss-priority";
        description
          "list of loss-priorities";
        leaf loss-priority {
          type enumeration {
            enum high {
             description "High Loss Priority";
            }
            enum medium-high {
             description "Medium-high Loss Priority";
            }
            enum medium-low {
             description "Medium-low Loss Priority";
            }
            enum low {
             description "Low Loss Priority";
            }
          }
          description
            "Loss-priority";
        }
      }
      description
        "Filter containing list of loss priorities";
    }

    augment "/policy:policies" +
            "/policy:policy-entry" +
            "/policy:classifier-entry" +
            "/policy:filter-entry" +
            "/diffserv:filter-params" {
      case forwarding-class {
          uses forwarding-class-cfg;
        description
          "Filter Type Internal-loss-priority";
      }
      case internal-loss-priority {
          uses loss-priority-cfg;
        description
          "Filter Type Internal-loss-priority";
      }
      description
        "Augments Diffserv Classifier with vendor" +
```

```
        " specific types";
    }

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

    identity mark-fwd-class {
      base policy:action-type;
      description
        "mark forwarding class action type";
    }

    identity mark-loss-priority {
      base policy:action-type;
      description
        "mark loss-priority action type";
    }

    grouping mark-fwd-class {
      container mark-fwd-class-cfg {
        leaf forwarding-class {
          type string;
          description
            "Forwarding class name";
        }
        description
          "mark-fwd-class container";
      }
      description
        "mark-fwd-class grouping";
    }

    grouping mark-loss-priority {
      container mark-loss-priority-cfg {
        leaf loss-priority {
          type enumeration {
            enum high {
             description "High Loss Priority";
            }
            enum medium-high {
             description "Medium-high Loss Priority";
            }
            enum medium-low {
             description "Medium-low Loss Priority";
            }
            enum low {
             description "Low Loss Priority";
```

```
            }
          }
          description
            "Loss-priority";
        }
        description
          "mark-loss-priority container";
      }
      description
        "mark-loss-priority grouping";
    }

    augment "/policy:policies" +
            "/policy:policy-entry" +
            "/policy:classifier-entry" +
            "/policy:classifier-action-entry-cfg" +
            "/diffserv:action-cfg-params" {
      case mark-fwd-class {
        uses mark-fwd-class;
        description
          "Mark forwarding class in the packet";
      }
      case mark-loss-priority {
          uses mark-loss-priority;
        description
          "Mark loss priority in the packet";
      }
      case meter-reference {
          uses action:meter-reference;
        description
          "Assign a meter as an action";
      }
      case discard {
          uses action:discard;
        description
          "Discard action";
      }
      case count {
          uses action:count;
        description
          "Count action - explicit count configuration";
      }
      description
        "Augments common diffserv policy actions";
    }

    augment "/action:meter-template" +
            "/action:meter-entry" +
```

```
            "/action:meter-type" +
            "/action:one-rate-tri-color-meter-type" +
            "/action:one-rate-tri-color-meter" {
       leaf one-rate-color-aware {
        type boolean;
        description
          "This defines if the meter is color-aware";
      }
    }
    augment "/action:meter-template" +
            "/action:meter-entry" +
            "/action:meter-type" +
            "/action:two-rate-tri-color-meter-type" +
            "/action:two-rate-tri-color-meter" {
       leaf two-rate-color-aware {
        type boolean;
        description
          "This defines if the meter is color-aware";
      }
    }

    /* example of augmenting a meter template with a
    /* vendor specific action */
    augment "/action:meter-template" +
            "/action:meter-entry" +
            "/action:meter-type" +
            "/action:one-rate-two-color-meter-type" +
            "/action:one-rate-two-color-meter" +
            "/action:exceed-action" +
            "/action:meter-action-params" +
            "/action:meter-action-val" {
      case meter-action-drop {
        description
            "meter drop";
             uses action:drop;
      }

      description
        "Augment the actions to basic meter";
    }

  }

  module example-compb-queue-policy {
    yang-version 1;
    namespace "urn:ietf:params:xml:ns:yang:example-compb-queue-policy";
    prefix queue-plcy;
```

```
import ietf-qos-classifier {
  prefix classifier;
}
import ietf-qos-policy {
  prefix policy;
}

organization "Company B";
contact
  "Editor:   XYZ
            <mailto:xyz@compb.com>";

description
  "This module defines a queue policy. The classification
   is based on aforwarding class, and the actions are queues.
   Copyright (c) 2014 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 2015-04-07 {
  description
    "Latest revision of diffserv policy";
  reference "RFC XXXX";
}

identity forwarding-class {
  base classifier:filter-type;
  description
     "Forwarding class filter type";
}

grouping forwarding-class-cfg {
  leaf forwarding-class-cfg {
    type string;
    description
      "forwarding-class name";
  }
  description
    "Forwarding class filter";
}
```

```
augment "/policy:policies" +
        "/policy:policy-entry" +
        "/policy:classifier-entry" +
        "/policy:filter-entry" {
  /* Does NOT support "logical-not" of forwarding class.
     Use "must"? */
  choice filter-params {
    description
      "Choice of filters";
    case forwarding-class-cfg {
      uses forwarding-class-cfg;
      description
        "Filter Type Internal-loss-priority";
    }
  }
  description
    "Augments Diffserv Classifier with fwd class filter";
}

identity compb-queue {
  base policy:action-type;
  description
    "compb-queue action type";
}

grouping compb-queue-name {
  container queue-name {
    leaf name {
      type string;
      description
        "Queue class name";
    }
    description
      "compb queue container";
  }
  description
    "compb-queue grouping";
}

augment "/policy:policies" +
        "/policy:policy-entry" +
        "/policy:classifier-entry" +
        "/policy:classifier-action-entry-cfg" {
  choice action-cfg-params {
    description
      "Choice of action types";
    case compb-queue {
      uses compb-queue-name;
```

```
        }
      }
      description
        "Augment the queue actions to queue policy entry";
    }
  }

  module example-compb-queue {
    yang-version 1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-compb-queue";
    prefix compb-queue;

    import ietf-qos-action {
      prefix action;
    }

    organization "Company B";
    contact
      "Editor:   XYZ
                 <mailto:xyz@compb.com>";

    description
      "This module describes a compb queue module. This is a
       template for a queue within a queue policy, referenced
       by name.

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

    revision 2015-04-07 {
      description
        "Latest revision of diffserv based classifier";
      reference "RFC XXXX";
    }

    container compb-queue {
      description
        "Queue used in compb architecture";
        leaf name {
            type string;
            description
              "A unique name identifying this queue";
        }
        uses action:queue;
        container excess-rate {
          choice excess-rate-type {
             case percent {
               leaf excess-rate-percent {
```

```
                type uint32 {
                  range "1..100";
                }
                description
                  "excess-rate-percent";
              }
            }
            case proportion {
              leaf excess-rate-poroportion {
                type uint32 {
                  range "1..1000";
                }
                description
                  "excess-rate-poroportion";
              }
            }
            description
              "Choice of excess-rate type";
          }
          description
              "Excess rate value";
        }
        leaf excess-priority {
          type enumeration {
            enum high {
             description "High Loss Priority";
            }
            enum medium-high {
             description "Medium-high Loss Priority";
            }
            enum medium-low {
             description "Medium-low Loss Priority";
            }
            enum low {
             description "Low Loss Priority";
            }
            enum none {
             description "No excess priority";
            }
          }
          description
            "Priority of excess (above guaranted rate) traffic";
        }
        container buffer-size {
          choice buffer-size-type {
             case percent {
               leaf buffer-size-percent {
                 type uint32 {
```

```
                  range "1..100";
                }
                description
                  "buffer-size-percent";
              }
            }
            case temporal {
              leaf buffer-size-temporal {
                type uint64;
                units "microsecond";
                description
                  "buffer-size-temporal";
              }
            }
            case remainder {
              leaf buffer-size-remainder {
                type empty;
                description
                  "use remaining of buffer";
              }
            }
            description
              "Choice of buffer size type";
          }
          description
              "Buffer size value";
        }
      }

      augment
        "/compb-queue" +
        "/queue-cfg" +
        "/algorithmic-drop-cfg" +
        "/drop-algorithm" {
        case random-detect {
          list drop-profile-list {
            key "priority";
            description
              "map of priorities to drop-algorithms";
            leaf priority {
              type enumeration {
                enum any {
                  description "Any priority mapped here";
                }
                enum high {
                  description "High Priority Packet";
                }
                enum medium-high {
```

```
                description "Medium-high Priority Packet";
              }
              enum medium-low {
                description "Medium-low Priority Packet";
              }
              enum low {
                description "Low Priority Packet";
              }
            }
            description
              "Priority of guaranteed traffic";
          }
          leaf drop-profile {
            type string;
            description
              "drop profile to use for this priority";
          }
        }
      }
      description
        "compb random detect drop algorithm config";
    }
  }

  module example-compb-scheduler-policy {
       yang-version 1;
       namespace "urn:ietf:params:xml:ns:yang:" +
                 "example-compb-scheduler-policy";
       prefix scheduler-plcy;

    import ietf-qos-action {
      prefix action;
    }

    import ietf-qos-policy {
      prefix policy;
    }

    organization "Company B";
    contact
      "Editor:   XYZ
                 <mailto:xyz@compb.com>";

     description
       "This module defines a scheduler policy. The classification
        is based on classifier-any, and the action is a scheduler.";

     revision 2015-04-07 {
```

```
        description
          "Latest revision of diffserv policy";
        reference "RFC XXXX";
      }

      identity queue-policy {
        base policy:action-type;
        description
          "forwarding-class-queue action type";
      }

     grouping queue-policy-name {
       container compb-queue-policy-name {
         leaf name {
           type string;
           description
             "Queue policy name";
         }
         description
           "compb-queue-policy container";
       }
       description
         "compb-queue policy grouping";
     }

     augment "/policy:policies" +
             "/policy:policy-entry" +
             "/policy:classifier-entry" +
             "/policy:classifier-action-entry-cfg" {
       choice action-cfg-params {
         case schedular {
           uses action:schedular;
         }
         case queue-policy {
           uses queue-policy-name;
         }
         description
           "Augment the scheduler policy with a queue policy";
       }
     }
   }
```

A.3. Example of Company C Diffserv Model

Company C vendor augmentation is based on Ericsson’s implementation differentiated QoS. This implementation first sorts traffic based on a classifier, which can sort traffic into one or more traffic forwarding classes. Then, a policer or meter policy references the

classifier and its traffic forwarding classes to specify different service levels for each traffic forwarding class.

Because each classifier sorts traffic into one or more traffic forwarding classes, this type of classifier does not align with ietf-qos-classifier.yang, which defines one traffic forwarding class per classifier. Additionally, Company C's policing and metering policies relies on the classifier's pre-defined traffic forwarding classes to provide differentiated services, rather than redefining the patterns within a policing or metering policy, as is defined in ietf-diffserv.yang.

Due to these differences, even though Company C uses all the building blocks of classifier and policy, Company C's augmentation does not use ietf-diffserv.yang to provide differentiated service levels. Instead, Company C's augmentation uses the basic building blocks, ietf-qos-policy.yang to provide differentiated services.

```
module example-compc-qos-policy {
  yang-version 1.1;
  namespace "urn:example-compc-qos-policy";
  prefix "compcqos";

  import ietf-qos-policy {
    prefix "pol";
  }

  import ietf-qos-action {
    prefix "action";
  }

  organization "";
  contact "";
  description "";

  revision 2016-09-26 {
    description "";
    reference "";
  }

  /* identities */

  identity compc-qos-policy {
    base pol:policy-type;
  }

  identity mdrr-queuing-policy {
    base compc-qos-policy;
```

```
}

identity pwfq-queuing-policy {
  base compc-qos-policy;
}

identity policing-policy {
  base compc-qos-policy;
}

identity metering-policy {
  base compc-qos-policy;
}

identity forwarding-policy {
  base compc-qos-policy;
}

identity overhead-profile-policy {
  base compc-qos-policy;
}

identity resource-profile-policy {
  base compc-qos-policy;
}

identity protocol-rate-limit-policy {
  base compc-qos-policy;
}

identity compc-qos-action {
  base pol:action-type;
}

/* groupings */

grouping redirect-action-grp {
  container redirect {
    /* Redirect options */
  }
}

/* deviations */

deviation "/pol:policies/pol:policy-entry" {
  deviate add {
    must "pol:type = compc-qos-policy" {
      description
```

```
            "Only policy types drived from compc-qos-policy " +
            "are supported";
        }
      }
    }

    deviation "/pol:policies/pol:policy-entry/pol:classifier-entry" {
      deviate add {
        must "../per-class-action = 'true'" {
          description
            "Only policies with per-class actions have classifiers";
        }
        must "((../sub-type != 'mdrr-queuing-policy') and " +
             " (../sub-type != 'pwfq-queuing-policy')) or " +
             "(((../sub-type = 'mdrr-queuing-policy') or " +
             "  (../sub-type = 'pwfq-queueing-policy')) and " +
             " ((classifier-entry-name = '0') or " +
             "  (classifier-entry-name = '1') or " +
             "  (classifier-entry-name = '2') or " +
             "  (classifier-entry-name = '3') or " +
             "  (classifier-entry-name = '4') or " +
             "  (classifier-entry-name = '5') or " +
             "  (classifier-entry-name = '6') or " +
             "  (classifier-entry-name = '7') or " +
             "  (classifier-entry-name = '8')))" {
          description
            "MDRR queuing policy's or PWFQ queuing policy's " +
            "classifier-entry-name is limited to the listed values";
        }
      }
    }

    deviation "/pol:policies/pol:policy-entry/pol:classifier-entry" +
              "/pol:classifier-action-entry-cfg" {
      deviate add {
        max-elements 1;
        must "action-type = 'compc-qos-action'" {
          description
            "Only compc-qos-action is allowed";
        }
      }
    }

    /* augments */

    augment "/pol:policies/pol:policy-entry" {
      when "pol:type = 'compc-qos-policy')" {
        description
```

```
          "Additional nodes only for diffserv-policy";
      }
      leaf sub-type {
        type identityref {
          base compc-qos-policy;
        }
        mandatory true;
        /* The value of this leaf must not change once configured */
      }
      leaf per-class-action {
        mandatory true;
        type boolean;
        must "(((. = 'true') and " +
             "  ((../sub-type = 'policing-policy') or " +
             "   (../sub-type = 'metering-policy') or " +
             "   (../sub-type = 'mdrr-queuing-policy') or " +
             "   (../sub-type = 'pwfq-queuing-policy') or " +
             "   (../sub-type = 'forwarding-policy'))) or " +
             " ((. = 'false') and " +
             "  ((../sub-type = 'overhead-profile-policy') or " +
             "   (../sub-type = 'resource-profile-policy') or " +
             "   (../sub-type = 'protocol-rate-limit-policy')))" {
          description
            "Only certain policies have per-class action";
        }
      }
      container traffic-classifier {
        presence true;
        when "../sub-type = 'policing-policy' or " +
             "../sub-type = 'metering-policy' or " +
             "../sub-type = 'forwarding-policy'" {
          description
            "A classifier for policing-policy or metering-policy";
        }
        leaf name {
          type string;
          mandatory true;
          description
            "Traffic classifier name";
        }
        leaf type {
          type enumeration {
            enum 'internal-dscp-only-classifier' {
              value 0;
              description
                "Classify traffic based on (internal) dscp only";
            }
            enum 'ipv4-header-based-classifier' {
```

```
              value 1;
              description
                "Classify traffic based on IPv4 packet header fields";
            }
            enum 'ipv6-header-based-classifier' {
              value 2;
              description
                "Classify traffic based on IPv6 packet header fields";
            }
          }
          mandatory true;
          description
            "Traffic classifier type";
        }
      }
      container traffic-queue {
        when "(../sub-type = 'mdrr-queuing-policy') or " +
             "(../sub-type = 'pwfq-queuing-policy')" {
          description
            "Queuing policy properties";
        }
        leaf queue-map {
          type string;
          description
            "Traffic queue map for queuing policy";
        }
      }
      container overhead-profile {
        when "../sub-type = 'overhead-profile-policy'" {
          description
            "Overhead profile policy properties";
        }
      }
      container resource-profile {
        when "../sub-type = 'resource-profile-policy'" {
          description
            "Resource profile policy properties";
        }
      }
      container protocol-rate-limit {
        when "../sub-type = 'protocol-rate-limit-policy'" {
          description
            "Protocol rate limit policy properties";
        }
      }
    }

    augment "/pol:policies/pol:policy-entry/pol:classifier-entry" +
```

```
          "/pol:classifier-action-entry-cfg/pol:action-cfg-params" {
        when "../../../pol:type = 'compc-qos-policy')" {
          description
            "Configurations for a classifier-policy-type policy";
        }
        case metering-or-policing-policy {
          when "../../../sub-type = 'policing-policy' or "
             + "../../../sub-type = 'metering-policy'" {
          }
          container dscp-marking {
            uses action:dscp-marking;
          }
          container precedence-marking {
            uses action:dscp-marking;
          }
          container priority-marking {
            uses action:priority;
          }
          container rate-limiting {
            uses action:one-rate-two-color-meter;
          }
        }
        case mdrr-queuing-policy {
          when "../../../sub-type = 'mdrr-queuing-policy'" {
            description
              "MDRR queue handling properties for the traffic " +
              "classified into current queue";
          }
          leaf mdrr-queue-weight {
            type uint8 {
              range "20..100";
            }
            units percentage;
          }
        }
        case pwfq-queuing-policy {
          when "../../../sub-type = 'pwfq-queuing-policy'" {
            description
              "PWFQ queue handling properties for traffic " +
              "classified into current queue";
          }
          leaf pwfq-queue-weight {
            type uint8 {
              range "20..100";
            }
            units percentage;
          }
          leaf pwfq-queue-priority {
```

```
            type uint8;
          }
          leaf pwfq-queue-rate {
            type uint8;
          }
        }
        case forwarding-policy {
          when "../../../sub-type = 'forwarding-policy'" {
            description
              "Forward policy handling properties for traffic " +
              "in this classifier";
          }
          uses redirect-action-grp;
        }
        description
          "Add the classify action configuration";
      }

    }
```

Authors' Addresses

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

Email: asechoud@cisco.com

Mahesh Jethanandani
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95134
US

Email: mjethanandani@gmail.com

Norm Strahle
Juniper Networks
1194 North Mathilda Avenue
Sunnyvale, CA 94089
US

Email: nstrahle@juniper.net

Ebben Aries
Juniper Networks
1194 North Mathilda Avenue
Sunnyvale, CA 94089
US

Email: exa@juniper.net

Ing-Wher Chen
Jabil

Email: ing-wher_chen@jabil.com