YANG Model for QoS

draft-asechoud-netmod-qos-model-00

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 http://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 October 6, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

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

This document defines a base YANG [RFC6020] data module for Quality of Service (QoS) configuration and operational 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 module defines a basic building blocks to define a classifier, policy, action and target. The base module has been augmented to include packet match fields, action parameters, and statistics data to define the DiffServ module. It is left up to individual vendors to stitch 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 generally modeled with qualifying rate and variability defined as a token bucket. Different vendors can extend it to have other types of generic meters as well. Single Rate Tri Color Marking (srTCM) [RFC2697] meter and Two Rate Tri Color Marking (trTCM) [RFC2698] meter are defined separately in the ietf-qos-action module, along with generic-meter.

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, policing 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 ietf-diffserv module by augmenting the QoS policy module, which includes the QoS action module.

Finally, statistics are realized in the ietf-diffserv module by augmenting the QoS target 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 can also be configured in the action module.

module: ietf-qos-action
  +--rw meter-template {meter-template-support}?
    +--rw meter-entry* [meter-name]
      +--rw meter-name string
      +--:(srtcm)
        +--rw srtcm
          +--rw committed-rate? uint64
          +--rw committed-burst? uint64
          +--rw excess-burst? uint64
          +--rw conform-action
            +--rw meter-action-type? identityref
            +--rw (val)?
              +--:(meter-action-dscp-mark)
                +--rw dscp-cfg
                  +--rw dscp? inet:dscp
                +--:(meter-action-drop)
                  +--rw drop-cfg
                    +--rw drop-action? empty
            +--rw conform-color
              +--rw classifier-entry-name? string
              +--rw classifier-entry-descr? string
              +--rw classifier-entry-filter-operation? identityref
++rw exceed-action
  +++rw meter-action-type? identityref
  +++rw (val)?
    ++-:(meter-action-dscp-mark)
    |  ++-rw dscp-cfg
    |     +++rw dscp? inet:dscp
    ++-:(meter-action-drop)
    +++rw drop-cfg
    +++rw drop-action? empty
++rw exceed-color
  +++rw classifier-entry-name? string
  +++rw classifier-entry-descr? string
  +++rw classifier-entry-filter-operation? identityref
++rw violate-action
  +++rw meter-action-type? identityref
  +++rw (val)?
    ++-:(meter-action-dscp-mark)
    |  ++-rw dscp-cfg
    |     +++rw dscp? inet:dscp
    ++-:(meter-action-drop)
    +++rw drop-cfg
    +++rw drop-action? empty
++rw violate-color
  +++rw classifier-entry-name? string
  +++rw classifier-entry-descr? string
  +++rw classifier-entry-filter-operation? identityref
++-:(trtcn)
  +++rw trtcn
  +++rw committed-rate? uint64
  +++rw committed-burst? uint64
  +++rw peak-rate? uint64
  +++rw peak-burst? uint64
++rw conform-action
  +++rw meter-action-type? identityref
  +++rw (val)?
    ++-:(meter-action-dscp-mark)
    |  ++-rw dscp-cfg
    |     +++rw dscp? inet:dscp
    ++-:(meter-action-drop)
    +++rw drop-cfg
    +++rw drop-action? empty
++rw conform-color
  +++rw classifier-entry-name? string
  +++rw classifier-entry-descr? string
  +++rw classifier-entry-filter-operation? identityref
```yang
to be read naturally
```

```
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
    +--ro qos-target-classifier-statistics* string
      +--ro classifier-entry-name? string
      +--ro classifier-entry-statistics
        +--ro classified-pkts? uint64
        +--ro classified-bytes? uint64
        +--ro classified-rate? uint64
```

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:classifier-entry
   /classifier:classifier-entry:
   +++- (filter-param)?
      +++- (dscp)
         +++- dscp-cfg* [dscp-min dscp-max]
         |+++- dscp-min      inet:dscp
         |+++- dscp-max      inet:dscp
      +++- (source-ip-address)
         +++- source-ip-address-cfg* [source-ip-addr]
         |+++- source-ip-addr      inet:ip-prefix
      +++- (destination-ip-address)
         +++- destination-ip-address-cfg* [destination-ip-addr]
         |+++- destination-ip-addr      inet:ip-prefix
      +++- (source-port)
         +++- source-port-cfg* [source-port-min source-port-max]
         |+++- source-port-min      inet:port-number
         |+++- source-port-max      inet:port-number
      +++- (protocol)
         +++- protocol-cfg* [protocol-min protocol-max]
         |+++- protocol-min      uint8
         |+++- protocol-max      uint8
augment /policy:policy-entry/policy:policy-entry
   /policy:policy-entry:
   +++- (action-cfg-params)?
      +++- (dscp-marking)
         +++- dscp-cfg
      |+++- dscp?      inet:dscp
      +++- (meter)
      |+++- (srtcm)
         +++- srtcm
         |+++- committed-rate?      uint64
         |+++- committed-burst?      uint64
         |+++- excess-burst?      uint64
         |+++- conform-action
         |   +++- meter-action-type?      identityref
         |   |+++- (val)?
         |   |   |+++- (meter-action-dscp-mark)
         |   |   |   +++- dscp-cfg
         |   |   |   |+++- dscp?      inet:dscp
         |   |+++- (meter-action-drop)
         |   |   +++- drop-cfg
         |   |   |+++- drop-action?      empty
++--rw conform-color
    ++--rw classifier-entry-name?  string
    ++--rw classifier-entry-descr?  string
    ++--rw classifier-entry-filter-operation?
        identityref
++--rw exceed-action
    ++--rw meter-action-type?  identityref
    ++--rw (val)?
        +--:(meter-action-dscp-mark)
        |  ++--rw dscp-cfg
        |     +--rw dscp?  inet:dscp
        +--:(meter-action-drop)
        |  ++--rw drop-cfg
        |     +--rw drop-action?  empty
++--rw exceed-color
    ++--rw classifier-entry-name?  string
    ++--rw classifier-entry-descr?  string
    ++--rw classifier-entry-filter-operation?
        identityref
++--rw violate-action
    ++--rw meter-action-type?  identityref
    ++--rw (val)?
        +--:(meter-action-dscp-mark)
        |  ++--rw dscp-cfg
        |     +--rw dscp?  inet:dscp
        +--:(meter-action-drop)
        |  ++--rw drop-cfg
        |     +--rw drop-action?  empty
++--rw violate-color
    ++--rw classifier-entry-name?  string
    ++--rw classifier-entry-descr?  string
    ++--rw classifier-entry-filter-operation?
        identityref
++:(generic-meter)
    ++--rw generic-meter-cfg
        ++--rw meter-list* [meter-id]
            ++--rw meter-id  uint16
            ++--rw rate-value?  uint64
            ++--rw rate-unit?  identityref
            ++--rw (burst-type)?
                +--:(size)
                |  ++--rw burst-size?  uint64
                +--:(interval)
                |  ++--rw burst-interval?  uint64
    ++--rw color
        |  ++--rw classifier-entry-name?  string
        |  ++--rw classifier-entry-descr?  string
        |  ++--rw classifier-entry-filter-operation?
augment /if:interfaces/if:interface/target:qos-target-entry 
/target:qos-target-classifier-statistics:
  +--ro diffserv-action-statistics
  |  +--ro generic-meter-statistics* [meter-id]
  |     |  +--ro meter-id         uint16
  |     |  +--ro metered-pkts?    uint64
  |     |  +--ro metered-bytes?   uint64
  |     |  +--ro metered-rate?    uint64
  |  +--ro srtcm-meter-statistics
  |     |  +--ro conform-pkts?    uint64
  |     |  +--ro conform-bytes?   uint64
  |     |  +--ro conform-rate?    uint64
  |     |  +--ro exceed-pkts?     uint64
  |     |  +--ro exceed-bytes?    uint64
  |     |  +--ro exceed-rate?     uint64
  |     |  +--ro violate-pkts?    uint64
  |     |  +--ro violate-bytes?   uint64
  |     |  +--ro violate-rate?    uint64
  |  +--ro trtcn-meter-statistics
  |     |  +--ro conform-pkts?    uint64
  |     |  +--ro conform-bytes?   uint64
  |     |  +--ro conform-rate?    uint64
  |     |  +--ro exceed-pkts?     uint64
  |     |  +--ro exceed-bytes?    uint64
  |     |  +--ro exceed-rate?     uint64
  |     |  +--ro violate-pkts?    uint64
  |     |  +--ro violate-bytes?   uint64
  |  +--ro fail-action
  |     |  +--ro meter-action-type? identityref
  |     |  +--ro (val)?
  |     |     |  +--:(meter-action-dscp-mark)
  |     |     |     |  +--ro dscp-cfg
  |     |     |     |     |  +--ro dscp? inet:dscp
  |     |     |     |  +--:(meter-action-drop)
  |     |     |     |     |  +--ro drop-cfg
  |     |     |     |     |     |  +--ro drop-action? empty
  |     |     |  +--ro next-meter-id?   uint16
6. Modules

6.1. IETF-QOS-CLASSIFIER

<CODE BEGINS>file "ietf-qos-classifier@2016-03-03.yang"

module ietf-qos-classifier {
  yang-version 1;
  prefix classifier;

  import ietf-inet-types {
    prefix inet;
  }

  organization "IETF NETMOD (Netmod Working Group) Working Group";
  contact
    "WG Web:   <http://tools.ietf.org/wg/netmod/>
    WG List:  <mailto:netmod@ietf.org>
    WG Chair: Jurgen Schoenwalder
              <mailto:j.schoenwaelder@jacobs-university.de>
    WG Chair: Lou Berger
              <mailto:lberger@labn.net>
    WG Chair: Kent Watsen
              <mailto:kwatsen@juniper.net>
    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

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;
    prefix policy;

    import ietf-qos-classifier {
        prefix classifier;
    }

    organization "IETF NETMOD (Netmod Working Group) Working Group";
    contact
        "WG Web:  <http://tools.ietf.org/wg/netmod/>
        WG List:  <mailto:netmod@ietf.org>
        WG Chair: Jurgen Schonwalder
            <mailto:j.schoenwaelder@jacobs-university.de>
        WG Chair: Lou Berger
            <mailto:lberger@labn.net>
        WG Chair: Kent Watsen
            <mailto:kwatsen@juniper.net>
        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 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;
        }
    }
}

6.3. IETF-QOS-ACTION

<CODE BEGINS>file "ietf-qos-action@2016-03-03.yang"
module ietf-qos-action {
    prefix action;

    import ietf-inet-types {
        prefix inet;
    }
    import ietf-qos-classifier {
        prefix classifier;
    }
    import ietf-qos-policy {
        prefix policy;
    }

    organization "IETF NETMOD (Netmod Working Group) Working Group";
    contact
        "WG Web:  <http://tools.ietf.org/wg/netmod/>
        WG List:  <mailto:netmod@ietf.org>

        WG Chair: Jurgen Schonwalder
        <mailto:j.schoenwaelder@jacobs-university.de>

        WG Chair: Lou Berger
        <mailto:lberger@labn.net>

        WG Chair: Kent Watsen
        <mailto:kwatsen@juniper.net>

        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)."
revision 2016-03-03 {
  description
    "Latest revision for qos actions";
  reference "RFC XXXX";
}

feature meter-template-support {
  description
    " This feature allows support of meter-template.";
}

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";
}
}

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";
}

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";
}

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

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

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

identity srtcm-meter {
    base policy:action-type;
    description "single rate three color meter action type";
}

identity trtcn-meter {

}
base policy:action-type;
description
  "two rate three color meter action type";
}

identity scheduler {
  base policy:action-type;
description
  "schedular action type";
}

identity always-drop {
  base policy:action-type;
description
  "always drop algorithm";
}

identity tail-drop {
  base policy:action-type;
description
  "tail drop algorithm";
}

identity random-detect {
  base policy:action-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-dscp-set {
  base meter-action-type;
description
  "dscp mark action type in a meter";
}

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 {
    leaf meter-action-type {
        type identityref {
            base meter-action-type;
        } description
            "meter action type";
    } choice val {
        case meter-action-dscp-mark {
            uses dscp-marking;
            description
                "meter action: mark";
        } case meter-action-drop {
            description
                "meter action: drop";
            uses drop;
        } description
            "meter action based on choice of meter action type";
    } description
        "meter action parameters";
    }
grouping generic-meter {
  leaf meter-id {
    type uint16;
    description
    "meter identifier";
  }
  uses rate-value-unit;
  uses burst;
  container color {
    uses classifier:classifier-entry-generic-attr;
    description
    "color aware & color blind attributes container";
  }
  container succeed-action {
    uses meter-action-params;
    leaf next-meter-id {
      type uint16;
      description
      "next meter identifier";
    }
    description
    "confirm action";
  }
  container fail-action {
    uses meter-action-params;
    leaf next-meter-id {
      type uint16;
      description
      "next meter identifier";
    }
    description
    "exceed action";
  }
  description
  "meter attributes";
}

grouping one-rate-tri-color-meter {
  container srtcm {
    leaf committed-rate {
      type uint64;
      units "bits-per-second";
      description
      "committed meter rate";
    }
    leaf committed-burst {
      type uint64;
      units "byes";
    }
description
"committed burst size";
}
leaf excess-burst {
  type uint64;
  units "byes";
  description
  "excess burst size";
}
container conform-action {
  uses meter-action-params;
  description
  "conform action";
}
container conform-color {
  uses classifier:classifier-entry-generic-attr;
  description
  "color aware & color blind attributes container";
}
container exceed-action {
  uses meter-action-params;
  description
  "exceed action";
}
container exceed-color {
  uses classifier:classifier-entry-generic-attr;
  description
  "color aware & color blind attributes container";
}
container violate-action {
  uses meter-action-params;
  description
  "violate action";
}
container violate-color {
  uses classifier:classifier-entry-generic-attr;
  description
  "color aware & color blind attributes container";
}
description
"one-rate-tri-color-meter attributes";
}
description
"one-rate-tri-color-meter grouping";
}
grouping two-rate-tri-color-meter {
  container trtcmt {

leaf committed-rate {
  type uint64;
  units "bits-per-second";
  description
    "committed meter rate";
}
leaf committed-burst {
  type uint64;
  units "bytes";
  description
    "committed burst size";
}
leaf peak-rate {
  type uint64;
  units "bits-per-second";
  description
    "peak meter rate";
}
leaf peak-burst {
  type uint64;
  units "bytes";
  description
    "peak burst size";
}
container conform-action {
  uses meter-action-params;
  description
    "conform action";
}
container conform-color {
  uses classifier:classifier-entry-generic-attr;
  description
    "color aware & color blind attributes container";
}
container exceed-action {
  uses meter-action-params;
  description
    "exceed action";
}
container exceed-color {
  uses classifier:classifier-entry-generic-attr;
  description
    "color aware & color blind attributes container";
}
container violate-action {
  uses meter-action-params;
  description
    "violate action";
container violate-color {
  uses classifier:classifier-entry-generic-attr;
  description
    "color aware & color blind attributes container";
}

description
  "two-rate-tri-color-meter attributes";
}
description
  "two-rate-tri-color-meter 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 always-drop {
          uses drop;
          description
            "Always Drop configuration container";
        }
        case tail-drop {
          container tail-drop-cfg {
            uses queue-limit;
            description
              "Tail Drop configuration container";
          }
        }
        description
          "Choice of Drop Algorithm";
      }
      description
        "Algorithmic Drop configuration container";
    }
    description
      "Queue configuration container";
  }
  description
    "Queue configuration grouping";
}
grouping scheduler {
  container scheduler-cfg {

uses min-rate;
uses max-rate;
description
  "Schedular configuration container";
}
description
  "Schedular configuration grouping";
}
grouping generic-metering {
  container generic-meter-cfg {
    list meter-list {
      key "meter-id";
      uses generic-meter;
      description
        "Generic Meter configuration";
    }
    description
      "Generic Meter list configuration container";
    }
    description
      "Generic Meter configuration container grouping";
  }
grouping metering {
  choice meter-type {
    case srtcm {
      uses one-rate-tri-color-meter;
      description
        "one rate tri-color meter";
    }
    case trtcm {
      uses two-rate-tri-color-meter;
      description
        "two rate tri-color meter";
    }
    case generic-meter {
      uses generic-metering;
      description
        "generic meter";
    }
    description
      " meter action based on choice of meter action type";
    }
    description
      "meter attributes";
  }
container meter-template {
  if-feature meter-template-support;
  description
    "list of meter-entry templates";
  list meter-entry {
    key "meter-name";
    description
      "meter entry template";
    leaf meter-name {
      type string;
      description
        "basic meter identifier";
    }
    uses metering;
  }
}

6.4. IETF-QOS-TARGET

<CODE BEGINS>file "ietf-qos-target@2016-03-03.yang"

module ietf-qos-target {
  yang-version 1;
  prefix target;

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

  organization "IETF NETMOD (Netmod Working Group) Working Group";
  contact
    "WG Web:  <http://tools.ietf.org/wg/netmod/>"
    "WG List:  <mailto:netmod@ietf.org>"
    "WG Chair:  Jurgen Schonwalder"
                "<mailto:j.schoenwaelder@jacobs-university.de>"
    "WG Chair:  Lou Berger"
                "<mailto:lberger@labn.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 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";
}
feature target-inline-policy-config {
    description "This feature allows the policy configuration directly under a target.";
}

grouping wred-stats {
    description "WRED Counters";
    leaf early-drop-pkts {
        type uint64;
        description "Early drop packets ";
    }
    leaf early-drop-bytes {
        type uint64;
        description "Early drop bytes ";
    }
}

grouping classifier-entry-stats {
    description "Classifier Counters";
    container classifier-entry-statistics {
        config false;
        description "This group defines the classifier filter statistics of each classifier entry";
        leaf classified-pkts {
            type uint64;
            description "Number of total packets which filtered to the classifier-entry";
        }
        leaf classified-bytes {
            type uint64;
            description "Number of total bytes which filtered to the classifier-entry";
        }
        leaf classified-rate {
            type uint64;
            units "bits-per-second";
            description
        }
    }
}
"Rate of average data flow through the classifier-entry";
}
}

grouping queuing-stats {
  description
  "Queuing Counters";
  container queuing-statistics {
    description
    "queue related statistics ";
    leaf output-pkts {
      type uint64;
      description
      "Number of packets transmitted from queue ";
    }
    leaf output-bytes {
      type uint64;
      description
      "Number of bytes transmitted from queue ";
    }
    leaf queue-size-pkts {
      type uint64;
      description
      "Number of packets currently buffered ";
    }
    leaf queue-size-bytes {
      type uint64;
      description
      "Number of bytes currently buffered ";
    }
    leaf drop-pkts {
      type uint64;
      description
      "Total number of packets dropped ";
    }
    leaf drop-bytes {
      type uint64;
      description
      "Total number of bytes dropped ";
    }
    container wred-stats {
      uses wred-stats;
      description
      "Container for WRED statistics";
    }
  }
}
grouping tri-color-marking-meter-stats {

description
"Tri-Colur-Marking Metering Counters";
leaf conform-pkts {
  type uint64;
  description
  "Number of conform packets";
}
leaf conform-bytes {
  type uint64;
  description
  "Bytes of conform packets";
}
leaf conform-rate {
  type uint64;
  units "bits-per-second";
  description
  "Traffic Rate measured as conforming";
}
leaf exceed-pkts {
  type uint64;
  description
  "Number of packets counted as exceeding";
}
leaf exceed-bytes {
  type uint64;
  description
  "Bytes of packets counted as exceeding";
}
leaf exceed-rate {
  type uint64;
  units "bits-per-second";
  description
  "Traffic Rate measured as exceeding";
}
leaf violate-pkts {
  type uint64;
  description
  "Number of packets counted as violating";
}
leaf violate-bytes {
  type uint64;
  description
  "Bytes of packets counted as violating";
}
leaf violate-rate {
type uint64;
units "bits-per-second";
description
"Traffic Rate measured as violating";
}
}
grouping meter-stats {
description
"Metering Counters";
list generic-meter-statistics {
key "meter-id";
description
"Meter statistics";
leaf meter-id {
    type uint16;
description
"Meter Identifier";
}
leaf metered-pkts {
    type uint64;
description
"Number of packets counted by the meter";
}
leaf metered-bytes {
    type uint64;
description
"Bytes of packets counted by the meter";
}
leaf metered-rate {
    type uint64;
    units "bits-per-second";
description
"Traffic Rate measured by the meter";
}
}
container srtcm-meter-statistics {
    uses tri-color-marking-meter-stats;
description
"Srtcm meter statistics";
}
container trtcm-meter-statistics {
    uses tri-color-marking-meter-stats;
description
"Trtcm meter statistics";
}
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 of 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";
    }
    list qos-target-classifier-statistics {
      config false;
      description
      "Statistics for each Classifier Entry in a Policy";
      leaf classifier-entry-name {
        type string;
        description
        "Classifier Entry Name";
      }
      uses classifier-entry-stats;
    }
  }
}
6.5. IETF-DIFFSERV

<CODE BEGINS>file "ietf-diffserv@2016-03-03.yang"

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

    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;
    }

    organization "IETF NETMOD (Netmod Working Group) Working Group";
    contact
        "WG Web:   <http://tools.ietf.org/wg/netmod/>
        WG List:  <mailto:netmod@ietf.org>

        WG Chair: Jurgen Schonwalder
            <mailto:j.schoenwaelder@jacobs-university.de>

        WG Chair: Lou Berger
            <mailto:lberger@labn.net>

        WG Chair: Kent Watsen
            <mailto:kwatsen@juniper.net>

        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 2016-03-03 {
  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;
      }
    }
7. Security Considerations
8. Acknowledgement

The editors of this document wishes to thank Fred Baker for
overviewing the document and provide useful comments, Andrew Mao for
the guidance and support, Fred Yip and Aleksandr Zhdankin for helpful
suggestions and contributions.

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,

Marker", RFC 2697, DOI 10.17487/RFC2697, September 1999,

[RFC2698] Heinanen, J. and R. Guerin, "A Two Rate Three Color
Marker", RFC 2698, DOI 10.17487/RFC2698, September 1999,

[RFC3289] Baker, F., Chan, K., and A. Smith, "Management Information
Base for the Differentiated Services Architecture",
RFC 3289, DOI 10.17487/RFC3289, May 2002,

the Network Configuration Protocol (NETCONF)", RFC 6020,
DOI 10.17487/RFC6020, October 2010,

RFC 6991, DOI 10.17487/RFC6991, July 2013,

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface
Management", RFC 7223, DOI 10.17487/RFC7223, May 2014,

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,
Appendix A.  Open Items

The current model represents hierarchical QoS with leaf and non-leaf nodes in a scheduling hierarchy without any restrictions of actions, such as AQM. Normally, that should not be allowed at non-leaf nodes. This will be addressed in a subsequent revision.

Appendix B.  Company A example

Company A Diffserv module 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.

B.1.  EXAMPLE-COMPA-DIFFSERV

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 2016-03-03 {
  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 min-rate {
  base policy:action-type;
  description
    "min-rate action type";
}

identity max-rate {
  base policy:action-type;
  description
    "max-rate action type";
}

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

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";
}
augment "/policy:policies/policy:policy-entry/ + 
"policy:classifier-entry/ + 
"policy:classifier-action-entry-cfg/ + 
"diffserv:action-cfg-params" {

case priority {
  uses action:prioritiy;
}
}
case min-rate {
  uses action:min-rate;
}
}
case max-rate {
  uses action:max-rate;
}
}
case random-detect {
  uses randomdetect;
}
}
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 "/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 hierarchical policy";
}
description
  "Augment the statistics to policy entry";
}
}

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