CATS Working Group
Internet Draft
Intended status: Standards Track
Expires: December 30, 2025

H. Yao
China Mobile
C. Lin
New H3C Technologies
Z. Li
China Mobile
Q. Xiong
ZTE Corporation

June 30, 2025

Data Model for Computing-Aware Traffic Steering (CATS)
draft-yl-cats-data-model-03

Abstract

This document defines a YANG data model for the configuration and management of Computing-Aware Traffic Steering (CATS) framework.

The YANG module defined in this document conforms to the Network Management Datastore Architecture (NMDA).

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 December 30, 2025.

Table of Contents

1. Introduction

[I-D. draft-ietf-cats-framework] introduces the framework definition of CATS. This document defines a YANG data model for CATS that can be used to configure and manage the CATS framework. This model imports and augments ietf-routing YANG model defined in [RFC8349].

1.1. Terminology

This document makes use of the terms as defined in [I-D. draft-ietf-cats-framework].

1.2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Tree Diagrams

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

1.4. Prefixes in Data Node Names

In this document, names of data nodes, actions, and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the corresponding YANG module, as shown in Table 1.

Prefix	YANG module	Reference
if	ietf-interfaces	[RFC8343]
ip	ietf-ip	[RFC8344]
cats	ietf-cats	Section 3
srv6-types	ietf-srv6-types	[I-D.ietf-spring-srv6-yang]
rt-types	ietf-routing-types	[RFC8294]
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]

Table 1: Prefixes and Corresponding YANG Modules

2. Design of the Configuration Data Model

2.1. Scope of Model

The model covers CATS [I-D.ietf-cats-framework].

This model can be used to configure and manage the CATS framework.

The operational state data and statistics can be retrieved by this model. The subscription and push mechanism defined in [RFC8639] and [RFC8641] can be implemented by the user to subscribe to notifications on the data nodes in this model.

The model contains all the basic configuration parameters to operate the protocol. Depending on the implementation choices, some systems may not allow some of the advanced parameters to be configurable.

The occasionally implemented parameters are modeled as optional features in this model. This model can be extended, and it has been structured in a way that such extensions can be conveniently made.

2.2. Specification

This model imports and augments ietf-routing YANG model defined in [RFC8349]. Both configuration branch and state branch of [RFC8349] are augmented. The configuration branch covers node base and policy configuration. The container "cats" is the top level container in this data model.

The YANG data model defined in this document conforms to the Network Management Datastore Architecture (NMDA) [RFC8342]. The operational state data is combined with the associated configuration data in the same hierarchy [RFC8407].

2.3. Overview

As defined in CATS framework, CATS system includes a control center and CATS routers. A control center performs management and control plane including computing information awareness function, service scheduling function, etc., that is, a control center collects computing and network information, and schedules service based on computing and network information. CATS Router performs data planes or routing functions, at least one computing information awareness function, service scheduling function and data forwarding function. CATS routers make forwarding decisions for a given service request that has been received from a client according to the capabilities and status information of both service instances and network.

As shown in Figure 1, the CATS framework structure consists of C-SMA, responsible for maintaining service metrics, C-PCE, responsible for maintaining forwarding table entries, and C-TC, responsible for traffic classification.

CATS-SBI: It could be used to report computing service information from CATS routers to the CATS-control center, and also could be used to send path and service policy information or computing service information to CATS-Forwarders.

C-SMA API: An extended interface between the CATS-Forwarders and the cloud management or between CATS-Forwarders and service-instance, it is used to report computing service metric information to the CATS-control center or to CATS-Forwarders.

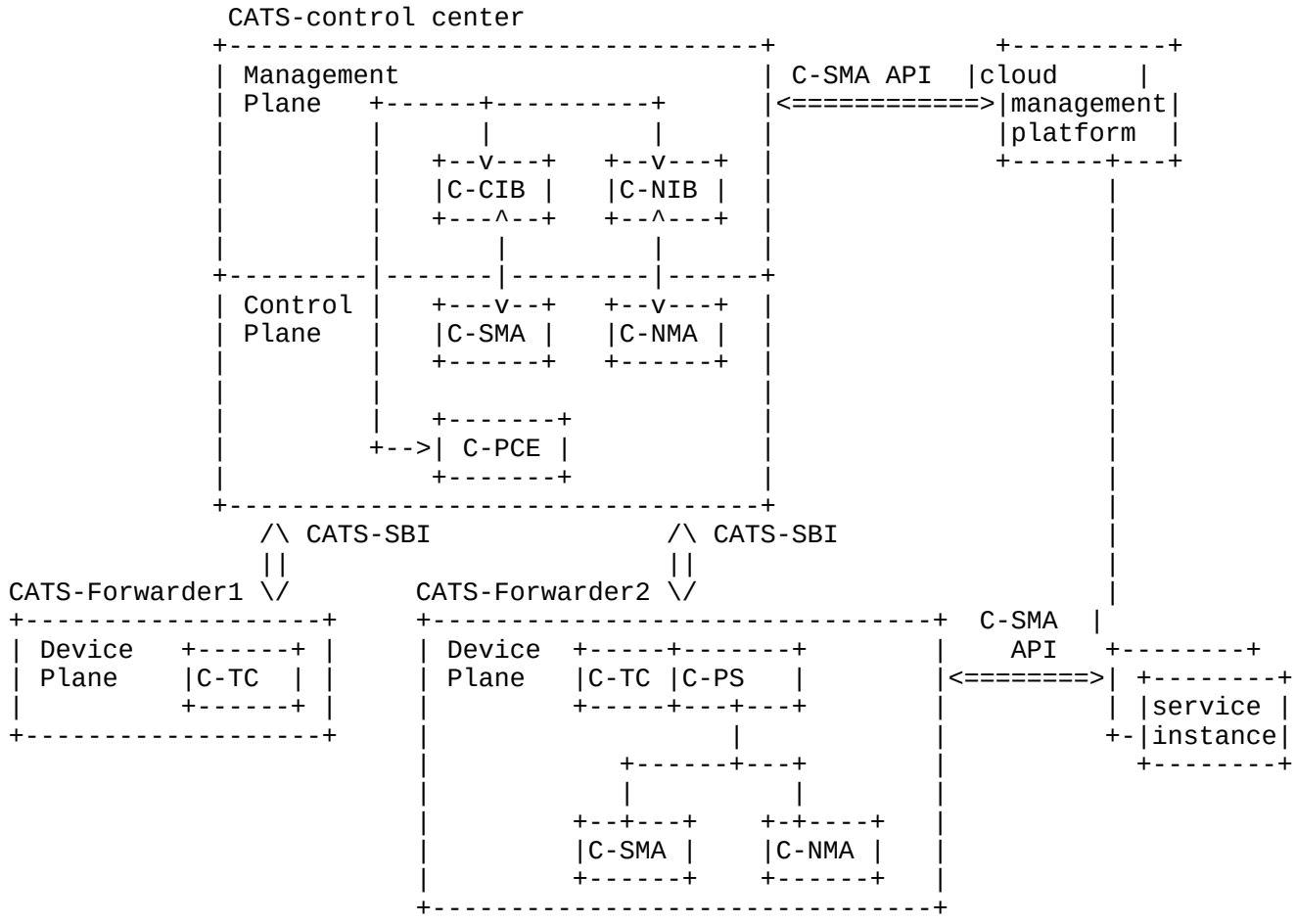

Figure 1: CATS System Architecture

2.4. CATS YANG Structure

This document defines a YANG data model for the configuration and management of CATS corresponding data. The data model is applicable to CATS-SBI interface and C-SMA API interface.

The CATS YANG is divided into two parts: the control-plane YANG data and the forwarding-plane YANG. The control-plane YANG includes basic YANG, traffic-classifier YANG, and service-metric YANG. The forwarding-plane YANG includes forwarding-paths YANG and flow YANG.

- The CATS base table provides interfaces for the functionality of the C-PCE component, which can be used for communication interfaces between the CATS-Control center and the cloud management platform, as well as for interfaces between the CATS-Control center and the CATS-Forwarder.

- The CATS traffic-classifier table provides interfaces for the functionality of the C-TC component, which can be used for interfaces between the CATS-Control center and the CATS-Forwarder.

- The CATS service-metric table provides interfaces for the CATS SMA component, which can be used for interfaces between the CATS-Control center and the CATS-Forwarder, as well as for transmitting service metrics information from the cloud management platform to the CATS-Control center. It is also used for forwarding service metric information from the CATS-Control center to the CATS-Forwarder.

- The CATS notify table is used by the management layer component and can be utilized for the CATS-forwarder to report events to the CATS-control center.

- The CATS Forwarding-Path table and the CATS Flow-Entry table can be used as interfaces for the CATS-Control center to distribute forwarding table entries to the CATS Forwarder, and they can also be utilized for the CATS Forwarder to report traffic statistics information to the CATS-Control center.

- The CATS-Control center can directly distribute specific CATS Flow-Entry to the CATS-Forwarder for guiding forwarding. It can also distribute CATS Forwarding-Path table and CATS traffic-classifier table to the CATS-Forwarder, allowing the CATS-Forwarder to proactively select paths according to forwarding policies and generate the CATS Flow-Entry table.

The following is a tree representation of the CATS YANG:

```
    module: ietf-cats
        +--rw cats
        |  +--rw base
        |  |  +--rw enable bool
        |  |  +--rw update-interval uint
        |  |  +--rw metric-limits   uint64
        |  |  +--rw flow-limits     uint64
        |  |  +--rw flow-timeout    uint
        |  |  +--rw service-policy
        |  |     +--rw cs-id        index, type cs-id
        |  |     +--rw policy-type  enumeration
        |  |
        |  +--rw traffic-classifiers
        |  |   +--rw traffic-classifier
        |  |      +--rw cs-id               index, type cs-id
        |  |      +--rw description         string
        |  |      +--rw server-port         ushort
        |  |      +--rw protocol            ushort
        |  |
        |  +--rw service-metrics
        |  |   +--rw service-metric
        |  |      +--rw cs-id               index, cs-id
        |  |      +--rw cis-id              index, cis-id
        |  |      +--rw source-type         index, uint
        |  |      +--rw priority            uint
        |  |      +--rw affinity            uint
        |  |      +--rw location            inet:ip-address
        |  |      +--rw metric
        |  |         +--metric-type          index, type uint
        |  |         +--metric               uint
        |  |
        |  +--rw forwarding-paths
        |  |   +--rw forwarding-path
        |  |      +--rw cs-id               index, type cs-id
        |  |      +--rw cis-id              index, type cis-id
        |  |      +--rw policy-type         enumeration
        |  |      +--rw weight              uint32
        |  |      +--rw forwarding-path-state
        |  |         +--rw path-id          index, type uint8
        |  |         +--rw next-hop-address inet:ip-address
        |  |         +--rw interface        if:interface
        |  |         +--rw dataplanetype
        |  |
        |  +--rw flow-entrys
        |  |     +--rw flow-entry
        |  |        +--rw source-address     index, inet:ip-address
        |  |        +--rw dest-address       index, inet:ip-address
        |  |        +--rw source-port        index, inet:ip-address
```

```
|  |        +--rw dest-port          index, inet:ip-address
|  |        +--rw protocol           index, ushort
|  |        +--rw forwarding-path
|  |           +--rw cs-id           type cs-id
|  |           +--rw cis-id          type cis-id
|  |           +--rw affinity        type uint
|  |           +--rw forwarding-path-state
|  |              +--rw path-id          index, type uint8
|  |              +--rw next-hop-address inet:ip-address
|  |              +--rw interface        if:interface
|  |              +--rw dataplanetype
|  |
|  |        +--ro flow-statistics
|  |           +--ro pkts      uint64
|  |           +--ro octets    uint64
|  |
|  +--n notify
|  |     +--ro metric-limit-reached  boolean
|  |     +--ro flow-limit-reached   boolean
```

Figure 2: Yang Organization and Hierarchy

2.5. CATS Control plane YANG Attributes

2.5.1. CATS base

```
|  +--rw base
|  |  +--rw enable          bool
|  |  +--rw update-interval uint
|  |  +--rw metric-limits   uint64
|  |  +--rw flow-limits     uint64
|  |  +--rw flow-timeout    uint
|  |  +--rw service-policy
|  |     +--rw cs-id        index, type cs-id
|  |     +--rw policy-type  enumeration
```

Figure 3: Base configuration tree view

In the base, you can set the "enable" attribute to enable/disable CATS function.

you can set the "cats-update-interval" parameter to determine the interval at which C-SMA notifies C-PS of metric changes. The default value for this interval is 30 seconds.

You can set the maximum number of table entries by configuring "entry-limits".

You can set the maximum number of flow tables by configuring "flow-limits".

When flow tables have been inactive for a long period of time, it is necessary to age out the stale state entries. This can be achieved by configuring "flow-timeout" to control the aging time of flow tables.

You can set the service-policy table for traffic routing, which includes affinity-based, service-metric-based, network-metric-based, and combined service-metric and network-metric-based routing.

2.5.2. CATS traffic-classifier

```
|  +--rw traffic-classifiers
|  |   +--rw traffic-classifier
|  |      +--rw cs-id               index,type cs-id
|  |      +--rw description         string
|  |      +--rw server-port         ushort
|  |      +--rw protocol            ushort
```

Figure 4: traffic-classifier tree view

The cats traffic-classifier table is used to define the features of the service and to classify the traffic.

2.5.3. CATS service-metric

```
|  +--rw service-metrics
|  |   +--rw cats-service-metric
|  |      +--rw cs-id               index, type cs-id
|  |      +--rw cis-id              index, type cis-id
|  |      +--rw source-type         index, source-type
|  |      +--rw priority            uint
|  |      +--rw affinity            uint
|  |      +--rw location            inet:ip-address
|  |      +--rw metric
|  |         +--metric-type         index, type uint
|  |         +--metric              uint
```

Figure 5: Service-metric tree view

The cats service-metric table is used to control the delivery service metrics on the control plane, thereby generating the forwarding table on the forwarding plane in conjunction with network metrics.

2.5.4. CATS notify

```
|  +--n notify
|  |     +--ro entry-limit-reached  boolean
|  |     +--ro flow-limit-reached   boolean
```

Figure 6: Notify tree view

When the number of cats table entries reaches the maximum and when the number of entries goes from maximum to not being maximum, a event notification will be sent indicating the change in the number of cats table entries reaching the maximum.

2.6. CATS Forwarding plane YANG Attributes

2.6.1. CATS forwarding-paths

```
|  +--rw forwarding-paths
|  |   +--rw forwarding-path
|  |      +--rw cs-id               index, type cs-id
|  |      +--rw cis-id              index, type cis-id
|  |      +--rw policy-type         enumeration
|  |      +--rw weight              uint32
|  |      +--rw forwarding-path-state
|  |         +--rw path-id          index, type uint8
|  |         +--rw forwarding-path-info
|  |            +--rw next-hop-address inet:ip-address
|  |            +--rw interface        if:interface
|  |            +--rw dataplanetype
```

Figure 7: Forwarding-path tree view

The cats forwarding-paths table is used for forwarding service traffic on the data plane. In scenarios with multiple paths, load balancing can be achieved based on the assigned weights.

2.6.2. CATS Flow Entry

```
|  +--rw flow-entrys
|  |     +--rw flow-entry
|  |        +--rw source-address     index, inet:ip-address
|  |        +--rw dest-address       index, inet:ip-address
|  |        +--rw source-port        index, inet:ip-address
|  |        +--rw dest-port          index, inet:ip-address
|  |        +--rw protocol           index, ushort
|  |        +--rw forwarding-path
|  |           +--rw cs-id           type cs-id
|  |           +--rw cis-id          type cis-id
|  |           +--rw affinity        type uint
|  |           +--rw forwarding-path-info
|  |              +--rw next-hop-address inet:ip-address
|  |              +--rw interface        if:interface
|  |              +--rw dataplanetype
|  |
|  |        +--ro flow-statistics
|  |           +--ro pkts      uint64
|  |           +--ro octets    uint64
```

Figure 8: Flow-entry tree view

When there is service traffic, Ingress CATS-Forwarder maintains a flow table to guide the forwarding of this flow. If the flow table remains in a stale state for more than flow-timeout, it will be deleted. The flow table also includes statistical information related to this flow.

3. CATS YANG Data model

<CODE BEGINS> file ietf-cats@2024-06-20.yang

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

   import ietf-inet-types {
      prefix inet;
      reference
         "RFC 6991 Common YANG Data Types";
   }

   import ietf-routing-types {
      prefix "rt-types";
   }

   import ietf-srv6-types {
      prefix "srv6-types";
   }

   import ietf-interfaces {
      prefix if;
   }

   organization "IETF CATS";
   contact
      "WG Web:
      WG List:
      ";

   description
        "This module describes a YANG model for CATS.
        This YANG model conforms to the Network Management
        Datastore Architecture (NMDA) as described in RFC 8342.

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

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject
        to the license terms contained in, the Revised BSD License
        set forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (httpstrustee.ietf.orglicense-info).
```

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

          The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL','SHALL
          NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT
          RECOMMENDED', 'MAY', and 'OPTIONAL' in this document are to
          be interpreted as described in BCP 14 (RFC 2119) (RFC 8174)
          when, and only when, they appear in all capitals, as shown
          here.";

      revision 2024-06-20 {
         description
            "Initial Version";
         reference
            "RFC XXXX: YANG Data Model for CATS";
     }

      typedef cs-id {
         type union {
            type uint32;
            type inet:ipv4-address;
            type inet:ipv6-address;
         }
         description
           "This type is for CATS CS-ID.";
      }

      typedef cis-id {
         type union {
            type uint32;
            type inet:ipv4-address;
            type inet:ipv6-address;
         }
         description
           "This type is for CATS CIS-ID.";
      }

      grouping service-policy-type {
         description
           "service policy type";
         leaf policy-type {
             description "service policy type";
             type enumeration {
                enum base-on-affinity {
                   value 0;
                   description "base on affinity";
                }
```

```
                enum base-on-sm-only {
                   value 1;
                   description "base on service metric only";
                }
                enum base-on-sm-nm {
                   value 2;
                   description "base on service metric and network
                   metric";
                }
             }
          }
       }
       container base {
          description
            "CATS base configuration.";
          leaf enable {
              type boolean;
              description
                "enable CATS"
          }
          leaf update-interval {
             type uint32;
             description
               "update-interval of CATS metric";
          }
          leaf entry-limits {
             type uint64;
             description
               "CATS metric entry limit";
          }
          leaf flow-limits {
             type uint64;
             description
               "CATS flow entry limit";
          }
          leaf flow-timeout {
             type uint32;
             description
               "CATS flow timeout when no flow";
          }
          container service-policy {
             description
               "CATS sercice policy";
             leaf cs-id {
                type cs-id;
                description
                  "cs-id";
             }
```

```
            uses service-policy-type;
          }
        }

        container traffic-classifiers {
          description
            "CATS traffic-classifier feature";
          list traffic-classifier {
            description
              "CATS traffic-classifier feature";
            key "cs-id";
            leaf cs-id {
              type cs-id;
              description
                "CATS CIS-ID";
            }
            leaf description {
              type string;
              description
                "description of this service, example: http";
            }
            leaf server-port {
              type uint16;
              description
                "server-port of the service.";
            }
            leaf protocol {
              type uint16;
              description
                "protocol of the service.";
            }
          }
        }

        container service-metrics {
          description
            "CATS service metric entry";
          list service-metric {
            description
              "CATS service metric entry";
            key "cs-id cis-id source-type";
            leaf cs-id {
              type cs-id;
              description
                "CATS cs-id";
            }
            leaf cis-id {
              type cis-id;
```

```
               description
                 "CATS cis-id";
            }
            leaf source-type {
               description
                 "source-type of the service-metric";
               type enumeration {
                  enum static {
                     value 0;
                     description "static configuration";
                  }
                  enum bgp {
                     value 1;
                     description "bgp protocol";
                  }
               }
            }
            leaf priority {
               type uint32;
               description
                 "server priority";
            }
            leaf affinity {
               type uint32;
               description
                 "server affinity";
            }
            leaf location {
               type inet:ip-address;
               description
                 "server location";
            }
            container service-metric {
               description
                 "service metric";
               list metric {
                  description "Different types of service     ";
                  key "metric-type";
                  leaf metric-type {
                     description "metric type";
                     type enumeration {
                        enum delay {
                           value 0;
                           description " Calculate the metric based on
                             transmission delay, where the metric value
                             is the delay time in milliseconds.";
                        }
                        enum service-capacity {
```

```
                           value 1;
                           description "Calculate the metric based on
                           service capacity, where the metric value is
                           the service capacity.";
                        }
                        enum service-availability-ratio {
                           value 2;
                           description "Calculate the metric based on
                           service capacity, where the metric value is
                           the current available capacity percentage.";
                        }
                        enum memory-capacity {
                           value 3;
                           description "Calculate the metric based on
                             memory capacity, where the metric value
                             is the memory capacity";
                        }
                        enum memory-ratio {
                           value 4;
                           description "Calculate the metric based on
                             memory utilization percentage, where the
                             metric value is the current memory
                             utilization percentage";
                        }
                     }
                  }
                  leaf metric {
                     type uint32;
                     description "metric value";
                  }
               }
            }
         }
      }

      grouping mpls-label-stack {
         description
            "Grouping for MPLS label stack";

         list labels {
            key "index";
            description
               "Stack containing MPLS labels";

            leaf index {
               type uint32;
               description "A unique ID of an MPLS label in labels
   list";
```

```
            }
            leaf label {
               type rt-types:mpls-label;
               description
                 "MPLS label value";
            }
         }
      }

      grouping srv6-sid-stack {
         description
          "Grouping for SRv6 label stack";

         list sids {
           key "index";
           description
             "Stack containing SRv6 SIDs";

           leaf index {
              type uint32;
              description "A unique ID of an SRv6 sid in sid list";
           }
           leaf sid {
              type srv6-types:srv6-sid;
              description
                "SRv6 sid value";
           }
         }
      }
      grouping path-forwarding-info {
         leaf next-hop-address {
            type inet:ip-address;
            description "Nexthop address";
         }
         leaf interface {
            type if:interface-ref;
            description "Outgoing interface handle";
         }
         container sid-list {
            description
              "Outgoing sid stack";
            choice dataplanetype {
               description
                 "Outgoing sids dataplane choice";
               case mpls {
                 uses mpls-label-stack;
               }
               case srv6 {
```

```
                uses srv6-sid-stack;
              }
            }
      }

      grouping path-forwarding-state {
         description "cats Forwarding path information";
         leaf path-id {
            type uint8;
            description "Primary path id";
         }
         uses path-forwarding-info;
        }
      }

      container forwarding-paths {
         description
           "Forwarding state of paths";
         list forwarding-path {
            description "Forwarding state of paths";
            key "cs-id cis-id";
            leaf cs-id {
               type cs-id;
               description "CATS cs-id";
            }
            leaf cis-id {
               type cis-id;
               description "CATS cis-id";
            }
            uses service-policy-type;
            leaf weight {
               type uint32;
               description "Path's weight for W-ECMP balancing";
            }
            list forwarding-path-state {
              key "path-id";
              description "Forwarding path state";
              uses path-forwarding-state;
            }
         }
     }

     container flow-entrys {
        description "flow entry";
        list flow-entry {
           description "flow entry";
           key "source-address dest-address source-port dest-port
   protocol";
```

```
            leaf source-address {
               type inet:ip-address;
               description "source address of flow";
            }
            leaf dest-address {
               type inet:ip-address;
               description "destination address of flow";
            }
            leaf source-port {
               type uint16;
               description "source port of flow";
            }
            leaf dest-port {
               type uint16;
               description "destination port of flow";
            }
            leaf protocol {
               type uint16;
               description "protocol of flow";
            }
            leaf cs-id {
               type cs-id;
               description "CATS cs-id";
            }
            leaf cis-id {
               type cis-id;
               description "CATS cis-id";
            }
            leaf affinity {
               type uint32;
               description "affinity";
            }
            uses path-forwarding-info;
            container flow-statistics {
               description "flow statistics";
               leaf pkts {
                  type uint64;
                  description "pkts";
               }
               leaf octets {
                  type uint64;
                  description "octets";
               }
            }
         }
      }

      container notify {
```

```
         description "event notify";
         leaf entry-limit-reached {
            type boolean;
            description "entry limit reached";
         }
         leaf flow-limit-reached {
            type boolean;
            description "flow entry limit reached";
         }
      }
   }
   <CODE ENDS>
```

4. Security Considerations

TBD

5. IANA Considerations

TBD

6. References

6.1. 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>.

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

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

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

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

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

[RFC8344] Bjorklund, M., "A YANG Data Model for IP Management", RFC 8344, DOI 10.17487/RFC8344, March 2018, <https://www.rfc-editor.org/info/rfc8344>.

6.2. Informative References

[I-D. draft-ietf-cats-framework] C. Li.,Z. Du.,M. Boucadair.,L. M. Contreras., J. Drake., " A Framework for Computing-Aware Traffic Steering (CATS)", draft-ietf-cats-framework-02(work in progress), April 2024.

[RFC7895] Bierman, A., Bjorklund, M., and K. Watsen, "YANG Module Library", RFC 7895, DOI 10.17487/RFC7895, June 2016, <https://www.rfc-editor.org/info/rfc7895>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

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

Contributors

TBD

Authors' Addresses

Huijuan Yao
China Mobile
No.32 XuanWuMen West Street
Beijing
100053
China
Email: yaohuijuan@chinamobile.com

Changwang Lin
New H3C Technologies
Beijing
China

Email: linchangwang.04414@h3c.com

Zhenqiang Li
China Mobile
China
Email: lizhenqiang@chinamobile.com

Quan Xiong
ZTE Corporation
Email: xiong.quan@zte.com.cn