Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 26, 2017

D. Kumar
Cisco
M. Wang
Q. Wu
Huawei
R. Rahman
S. Raghavan
Cisco

December 23, 2016

Generic YANG Data Model for Connectionless Operations, Administration, and Maintenance(OAM) protocols
draft-ietf-lime-yang-connectionless-oam-03

Abstract

This document presents a base YANG Data model for connectionless OAM protocols. It provides a technology-independent abstraction of key OAM constructs for connectionless protocols. The Based model presented here can be extended to include technology specific details. This is leading to uniformity between OAM protocols and support nested OAM workflows (i.e., performing OAM functions at different or same levels through a unified interface).

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Operations, Administration, and Maintenance (OAM) are important networking functions that allow operators to:

1. Monitor networks connections (Reachability Verification, Continuity Check).

2. Troubleshoot failures (Fault verification and localization).

3. Monitor Performance

An overview of OAM tools is presented at [RFC7276].

Ping and Traceroute [RFC792], [RFC4443] are well-known fault verification and isolation tools, respectively, for IP networks. Over the years, different technologies have developed similar tools for similar purposes.

In this document, we presents a base YANG Data model for connectionless OAM protocols. The generic YANG model for connectionless OAM only includes configuration data and state data. It can be used in conjunction with data retrieval method model[lime retrieval methods], which focuses on data retrival procedures like RPC. However it also can be used independently of data retrieval method model.

2. Conventions used in this document

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].

The following terms are defined in [RFC6241] and are not redefined here:

o client

o configuration data

o server

o state data

The following terms are defined in [RFC6020] and are not redefined here:

o augment

o data model

o data node

The terminology for describing YANG data models is found in [RFC6020].

2.1. Terminology

TP - Test Point

MAC - Media Access Control

BFD - Bidirectional Forwarding Detection

RPC - A Remote Procedure Call, as used within the NETCONF protocol

2.2. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

Each node is printed as:

```
<status> <flags> <name> <opts> <type>

<status> is one of:
     +  for current
     x  for deprecated
     o  for obsolete


<flags> is one of:

    rw for configuration data
    ro for non-configuration data
    -x for rpcs
    -n for notifications


<name> is the name of the node
```

If the node is augmented into the tree from another module, its name is printed as <prefix>:<name>.

```
<opts> is one of:

     ?  for an optional leaf or choice
     !  for a presence container
     *  for a leaf-list or list
     [<keys>] for a list's keys

<type> is the name of the type for leafs and leaf-lists
```

3. Overview of the Connectionless OAM Model

At the top of the Model, there is an oper container for session statistics. Grouping is also defined for common session statistics and these are applicable for proactive OAM sessions. Multiple test-point-locations keyed using technology specific keys (eg., IPv4 address for IPv4 locations) are possible by augmented network nodes which are defined in [I-D.draft-ietf-i2rs-yang-network-topo] to describe the network hierarchies and the inventory of nodes contained in a network. Each test-point-location is chosen based on location-type which when chosen, leads to a container that includes a list of test-point-locations keyed by technology specific keys. Each test point location includes a test-point-location-info. The test-point-location-info includes tp-technology, tp-tools, and connectionless-oam-layers. The groupings of tp-address and tp-address-vrf are kept out of test-point-location-info to make it addressing agnostic and allow varied composition. Depending upon the choice of the location-type (determined by the tp-address-vrf), the containers differ in its composition of test-point-locations while the test-point-location-info, is a common aspect of every test-point-location. The vrf is used to describe the corresponding network instance. The tp-technology indicate oam technology details. The tp-tools describe the oam tools supported. The connectionless-oam- layers is used to describe the relationship of one test point with other test points. The level in oam-layers indicate whether related oam test point is client layer, server layer or same layer. The Model is augmented to /nd:networks/nd:network/nd:node using Test Point Locations defined below.

3.1. TP Address

In connectionless OAM, the tp address is defined with the following type:

- o MAC address

- o IPv4 or IPv6 address

- o a pair of source, destination addresses, and interface (Useful for BFD)

- o FEC

- o System-id to represent the device or node.

3.2. Tools

In connectionless OAM, the tools attribute is used to describe a toolset for fault detection and isolation, and for performance measurement. And it can serve as a constraint condition when the base model be extended to specific OAM technology. For example, to fulfill the icmp ping configuration, the "../coam:continuity-check" should be set to "true", and then the lime base model should be augmented with icmp ping specific details.

3.3. OAM-layers

As typical networks have a multi-layer architecture, the set of OAM protocols similarly take a multi-layer structure; each layer has its own OAM protocols [RFC7276] and is corresponding to specific network portion or path and has associated test points. OAM-layers is referred to a list of upper layer, lower layer that are related to current test point. This allow users to easily navigate up and down to efficiently troubleshoot a connectivity issue at different layer. In this model, we have kept level default as 0, when all test points are located at the same layer. Level is provided for scenarios where it might be possible to define layering relationship as it can be used to stitching fault at related OAM layers. For example, there is a network in which data traffic between two customer edges is transported over three consecutive network portions, the current test point is located in the second network portion. If there is a defect in the first network portion is located at the upstream of the second network portion, the level of the first network portion is set to "-1". If the third network portion is located at the downstream of the second network portion and the level is set to "1". In another case, if the first network portion and the third network portion is in the same level of thesecond network portion, the level is set to "0". The snippet below depicts an example of OAM layers.

```
                list oam-layers {
                  key "index";
                  leaf index {
                     type uint16 {
                        range "0..65535";
                     }
                  }
                  leaf level {
                      type int32 {
                           range "-1..1";
                      }
                      description
                        "Level";
                  }
                  ordered-by user;
                  description
                     "list of related oam layers.";
                }
```

3.4. Test Point Locations Information

This is a generic grouping for Test Point Locations Information. It Provide details of Test Point Location using Tools, OAM-Layers grouping defined above.

3.5. Test Point Locations

This is a generic grouping for Test Point Locations. Choice statement is used to define locations types, for example ipv4-location-type, ipv6-location-type, etc. Container is defined under each location type containing list keyed to test point address, Test Point Location Information defined in section above, and routing instance vrf name if required.

3.6. Path Discovery Data

This is a generic grouping for path discovery data model that can be retrieved by any data retrieval methods including RPCs. Path discovery data output from methods, includes src-test-point, dst-test-point, sequence-number, hop-cnt, session statistics of various kinds,path verification and path trace related information. Path discovery includes data to be retrieved on a per-hop basis via a list of path-trace-info-list which includes information like timestamps, ingress-interface, egress-interface and app-meta-data. The path discovery data model is made generic enough to allow active, passive and hybrid OAMs to do the retrieval. None of the fields are made mandatory for that reason. Noted that the retrieval methods are defined in [lime retrieval methods].

3.7. Continuity Check Data

This is a generic grouping for continuity check data model that can be retrieved by any data retrieval methods including RPCs. Continuity check data output from methods, includes src-test-point, dst-test-point, sequence-number, hop-cnt and session statistics of various kinds. The continuity check data model is made generic enough to allow active, passive and hybrid OAMs to do the retrieval. None of the fields are made mandatory for that reason. Noted that the retrieval methods are defined in [lime retrieval methods].

3.8. OAM data hierarchy

The complete data hierarchy related to the OAM YANG model is presented below.

```
module: ietf-connectionless-oam
    +--ro oper {continuity-check}?
       +--ro cc-ipv4-sessions-statistics
       |  +--ro cc-session-statistics
       |     +--ro session-count?              uint32
       |     +--ro session-up-count?           uint32
       |     +--ro session-down-count?         uint32
       |     +--ro session-admin-down-count?   uint32
       +--ro cc-ipv6-sessions-statistics
          +--ro cc-session-statistics
             +--ro session-count?              uint32
             +--ro session-up-count?           uint32
             +--ro session-down-count?         uint32
             +--ro session-admin-down-count?   uint32
  augment /nd:networks/nd:network/nd:node:
    +--rw tp-address-type-value?                        identityref
    +--rw (location-type)?
       +--:(ipv4-location-type)
       |  +--rw test-point-ipv4-location-list
       |     +--rw test-point-locations* [ipv4-location]
       |        +--rw ipv4-location    inet:ipv4-address
       |        +--rw vrf?             routing-instance-ref
       |        +--rw (technology)?
       |        |  +--:(technology-null)
       |        |  |  +--rw tech-null?       empty
       |        |  +--:(technology-string)
       |        |     +--rw ipv4-icmp?       string
       |        +--rw tp-tools
       |        |  +--rw connectivity-verification?   boolean
       |        |  +--rw continuity-check?            boolean
       |        |  +--rw path-discovery?              boolean
       |        +--rw root?
```

```
       |        +--rw oam-layers* [index]
       |           +--rw index                                uint16
       |           +--rw level?                               int32
       |           +--rw (tp-address)?
       |              +--:(mac-address)
       |              |  +--rw mac-address-location?                yang:mac-
address
       |              +--:(ipv4-address)
       |              |  +--rw ipv4-location?                       inet:ipv4-
address
       |              +--:(ipv6-location)
       |              |  +--rw ipv6-address?                        inet:ipv6-
address
       |              +--:(tunnel-location)
       |              |  +--rw tunnel-location?                     uint32
       |              +--:(ip-prefix-location)
       |              |  +--rw ip-prefix-location?                  inet:ip-
prefix
       |              +--:(route-dist-location)
       |              |  +--rw route-dist-location?                 uint32
       |              +--:(group-ip-address-location)
       |              |  +--rw group-ip-address-location?           IP-
Multicast-Group-Address
       |              +--:(as-number-location)
       |              |  +--rw as-number-location?                  inet:as-
number
       |              +--:(lsp-id-location)
       |              |  +--rw lsp-id-location?                     string
       |              +--:(system-id-location)
       |              |  +--rw system-id-location?                  inet:uri
       |              +--:(connection-oriented-location)
       |                 +--rw maintenance-domain?                  -> /
goam:domains/domain/MD-name-string
       |                 +--rw maintenance-association?             -> /
goam:domains/domain[goam:MD-name-string = current()/../maintenance-domain]/MAs/
MA/MA-name-string
       |                 +--rw maintenance-association-end-point?   -> /
goam:domains/domain[goam:MD-name-string = current()/../maintenance-domain]/MAs/
MA[goam:MA-name-string = current()/../maintenance-association]/MEP/mep-name
       +--:(ipv6-location-type)
       |  +--rw test-point-ipv6-location-list
       |     +--rw test-point-locations* [ipv6-location]
       |        +--rw ipv6-location    inet:ipv6-address
       |        +--rw vrf?             routing-instance-ref
       |        +--rw (technology)?
       |        |  +--:(technology-null)
       |        |  |  +--rw tech-null?       empty
       |        |  +--:(technology-string)
```

```
   |        |     +--rw ipv4-icmp?       string
   |        +--rw tp-tools
   |        |  +--rw connectivity-verification?   boolean
   |        |  +--rw continuity-check?            boolean
   |        |  +--rw path-discovery?              boolean
   |        +--rw root?
   |        +--rw oam-layers* [index]
   |           +--rw index                                uint16
   |           +--rw level?                               int32
   |           +--rw (tp-address)?
   |              +--:(mac-address)
```

```
       |              |  +--rw mac-address-location?                yang:mac-
address
       |              +--:(ipv4-address)
       |              |  +--rw ipv4-location?                       inet:ipv4-
address
       |              +--:(ipv6-location)
       |              |  +--rw ipv6-address?                        inet:ipv6-
address
       |              +--:(tunnel-location)
       |              |  +--rw tunnel-location?                     uint32
       |              +--:(ip-prefix-location)
       |              |  +--rw ip-prefix-location?                  inet:ip-
prefix
       |              +--:(route-dist-location)
       |              |  +--rw route-dist-location?                 uint32
       |              +--:(group-ip-address-location)
       |              |  +--rw group-ip-address-location?           IP-
Multicast-Group-Address
       |              +--:(as-number-location)
       |              |  +--rw as-number-location?                  inet:as-
number
       |              +--:(lsp-id-location)
       |              |  +--rw lsp-id-location?                     string
       |              +--:(system-id-location)
       |              |  +--rw system-id-location?                  inet:uri
       |              +--:(connection-oriented-location)
       |                 +--rw maintenance-domain?                  -> /
goam:domains/domain/MD-name-string
       |                 +--rw maintenance-association?             -> /
goam:domains/domain[goam:MD-name-string = current()/../maintenance-domain]/MAs/
MA/MA-name-string
       |                 +--rw maintenance-association-end-point?   -> /
goam:domains/domain[goam:MD-name-string = current()/../maintenance-domain]/MAs/
MA[goam:MA-name-string = current()/../maintenance-association]/MEP/mep-name
       +--:(mac-location-type)
       |  +--rw test-point-mac-address-location-list
       |     +--rw test-point-locations* [mac-address-location]
       |        +--rw mac-address-location    yang:mac-address
       |        +--rw (technology)?
       |        |  +--:(technology-null)
       |        |  |  +--rw tech-null?             empty
       |        |  +--:(technology-string)
       |        |     +--rw ipv4-icmp?             string
       |        +--rw tp-tools
       |        |  +--rw connectivity-verification?   boolean
       |        |  +--rw continuity-check?            boolean
       |        |  +--rw path-discovery?              boolean
       |        +--rw root?
       |        +--rw oam-layers* [index]
       |           +--rw index                                uint16
       |           +--rw level?                               int32
       |           +--rw (tp-address)?
       |              +--:(mac-address)
       |              |  +--rw mac-address-location?                yang:mac-
address
       |              +--:(ipv4-address)
       |              |  +--rw ipv4-location?                       inet:ipv4-
address
       |              +--:(ipv6-location)
       |              |  +--rw ipv6-address?                        inet:ipv6-
address
       |              +--:(tunnel-location)
       |              |  +--rw tunnel-location?                     uint32
       |              +--:(ip-prefix-location)
       |              |  +--rw ip-prefix-location?                  inet:ip-
prefix
       |              +--:(route-dist-location)
       |              |  +--rw route-dist-location?                 uint32
       |              +--:(group-ip-address-location)
       |              |  +--rw group-ip-address-location?           IP-
Multicast-Group-Address
       |              +--:(as-number-location)
       |              |  +--rw as-number-location?                  inet:as-
number
       |              +--:(lsp-id-location)
       |              |  +--rw lsp-id-location?                     string
       |              +--:(system-id-location)
       |              |  +--rw system-id-location?                  inet:uri
       |              +--:(connection-oriented-location)
       |                 +--rw maintenance-domain?                  -> /
goam:domains/domain/MD-name-string
       |                 +--rw maintenance-association?             -> /
goam:domains/domain[goam:MD-name-string = current()/../maintenance-domain]/MAs/
MA/MA-name-string
       |                 +--rw maintenance-association-end-point?   -> /
goam:domains/domain[goam:MD-name-string = current()/../maintenance-domain]/MAs/
MA[goam:MA-name-string = current()/../maintenance-association]/MEP/mep-name
       +--:(tunnel-location-type)
       |  +--rw test-point-tunnel-address-location-list
       |     +--rw test-point-locations* [tunnel-location]
       |        +--rw tunnel-location    uint32
       |        +--rw vrf?               routing-instance-ref
       |        +--rw (technology)?
       |        |  +--:(technology-null)
       |        |  |  +--rw tech-null?         empty
       |        |  +--:(technology-string)
       |        |     +--rw ipv4-icmp?         string
       |        +--rw tp-tools
       |        |  +--rw connectivity-verification?   boolean
       |        |  +--rw continuity-check?            boolean
       |        |  +--rw path-discovery?              boolean
       |        +--rw root?
       |        +--rw oam-layers* [index]
       |           +--rw index                              uint16
       |           +--rw level?                             int32
       |           +--rw (tp-address)?
       |              +--:(mac-address)
       |              |  +--rw mac-address-location?                yang:mac-
address
       |              +--:(ipv4-address)
       |              |  +--rw ipv4-location?                      inet:ipv4-
address
       |              +--:(ipv6-location)
       |              |  +--rw ipv6-address?                       inet:ipv6-
address
       |              +--:(tunnel-location)
       |              |  +--rw tunnel-location?                    uint32
       |              +--:(ip-prefix-location)
       |              |  +--rw ip-prefix-location?                 inet:ip-
prefix
       |              +--:(route-dist-location)
       |              |  +--rw route-dist-location?                uint32
       |              +--:(group-ip-address-location)
       |              |  +--rw group-ip-address-location?           IP-
Multicast-Group-Address
       |              +--:(as-number-location)
       |              |  +--rw as-number-location?                  inet:as-
number
       |              +--:(lsp-id-location)
       |              |  +--rw lsp-id-location?                     string
       |              +--:(system-id-location)
       |              |  +--rw system-id-location?                  inet:uri
       |              +--:(connection-oriented-location)
       |                 +--rw maintenance-domain?                  -> /
goam:domains/domain/MD-name-string
       |                 +--rw maintenance-association?             -> /
goam:domains/domain[goam:MD-name-string = current()/../maintenance-domain]/MAs/
MA/MA-name-string
       |                 +--rw maintenance-association-end-point?   -> /
goam:domains/domain[goam:MD-name-string = current()/../maintenance-domain]/MAs/
MA[goam:MA-name-string = current()/../maintenance-association]/MEP/mep-name
       +--:(ip-prefix-location-type)
       |  +--rw test-point-ip-prefix-location-list
       |     +--rw test-point-locations* [ip-prefix-location]
       |        +--rw ip-prefix-location    inet:ip-prefix
       |        +--rw vrf?                  routing-instance-ref
       |        +--rw (technology)?
       |        |  +--:(technology-null)
       |        |  |  +--rw tech-null?            empty
       |        |  +--:(technology-string)
       |        |     +--rw ipv4-icmp?            string
       |        +--rw tp-tools
       |        |  +--rw connectivity-verification?   boolean
       |        |  +--rw continuity-check?            boolean
       |        |  +--rw path-discovery?              boolean
       |        +--rw root?
       |        +--rw oam-layers* [index]
       |           +--rw index                              uint16
       |           +--rw level?                             int32
       |           +--rw (tp-address)?
       |              +--:(mac-address)
       |              |  +--rw mac-address-location?                yang:mac-
address
       |              +--:(ipv4-address)
       |              |  +--rw ipv4-location?                       inet:ipv4-
address
       |              +--:(ipv6-location)
       |              |  +--rw ipv6-address?                        inet:ipv6-
address
       |              +--:(tunnel-location)
       |              |  +--rw tunnel-location?                     uint32
       |              +--:(ip-prefix-location)
       |              |  +--rw ip-prefix-location?                  inet:ip-
prefix
       |              +--:(route-dist-location)
       |              |  +--rw route-dist-location?                 uint32
       |              +--:(group-ip-address-location)
       |              |  +--rw group-ip-address-location?           IP-
Multicast-Group-Address
       |              +--:(as-number-location)
       |              |  +--rw as-number-location?                  inet:as-
number
       |              +--:(lsp-id-location)
       |              |  +--rw lsp-id-location?                     string
       |              +--:(system-id-location)
       |              |  +--rw system-id-location?                  inet:uri
       |              +--:(connection-oriented-location)
       |                 +--rw maintenance-domain?                  -> /
goam:domains/domain/MD-name-string
       |                 +--rw maintenance-association?             -> /
goam:domains/domain[goam:MD-name-string = current()/../maintenance-domain]/MAs/
MA/MA-name-string
       |                 +--rw maintenance-association-end-point?   -> /
goam:domains/domain[goam:MD-name-string = current()/../maintenance-domain]/MAs/
MA[goam:MA-name-string = current()/../maintenance-association]/MEP/mep-name
       +--:(route-distinguisher-location-type)
       |  +--rw test-point-route-dist-location-list
       |     +--rw test-point-locations* [route-dist-location]
       |        +--rw route-dist-location    uint32
       |        +--rw vrf?                   routing-instance-ref
       |        +--rw (technology)?
       |        |  +--:(technology-null)
       |        |  |  +--rw tech-null?             empty
       |        |  +--:(technology-string)
       |        |     +--rw ipv4-icmp?             string
       |        +--rw tp-tools
       |        |  +--rw connectivity-verification?   boolean
       |        |  +--rw continuity-check?            boolean
       |        |  +--rw path-discovery?              boolean
       |        +--rw root?
       |        +--rw oam-layers* [index]
       |           +--rw index                                 uint16
       |           +--rw level?                                int32
       |           +--rw (tp-address)?
       |              +--:(mac-address)
       |              |  +--rw mac-address-location?                yang:mac-
address
       |              +--:(ipv4-address)
       |              |  +--rw ipv4-location?                       inet:ipv4-
address
       |              +--:(ipv6-location)
       |              |  +--rw ipv6-address?                        inet:ipv6-
address
       |              +--:(tunnel-location)
       |              |  +--rw tunnel-location?                     uint32
       |              +--:(ip-prefix-location)
       |              |  +--rw ip-prefix-location?                  inet:ip-
prefix
       |              +--:(route-dist-location)
       |              |  +--rw route-dist-location?                 uint32
       |              +--:(group-ip-address-location)
       |              |  +--rw group-ip-address-location?           IP-
Multicast-Group-Address
       |              +--:(as-number-location)
       |              |  +--rw as-number-location?                  inet:as-
number
       |              +--:(lsp-id-location)
       |              |  +--rw lsp-id-location?                     string
       |              +--:(system-id-location)
       |              |  +--rw system-id-location?                  inet:uri
       |              +--:(connection-oriented-location)
       |                 +--rw maintenance-domain?                  -> /
goam:domains/domain/MD-name-string
```

```
       |                 +--rw maintenance-association?             -> /
goam:domains/domain[goam:MD-name-string = current()/../maintenance-domain]/MAs/
MA/MA-name-string
       |                 +--rw maintenance-association-end-point?   -> /
goam:domains/domain[goam:MD-name-string = current()/../maintenance-domain]/MAs/
MA[goam:MA-name-string = current()/../maintenance-association]/MEP/mep-name
       +--:(group-ip-address-location-type)
       |  +--rw test-point-group-ip-address-location-list
       |     +--rw test-point-locations* [group-ip-address-location]
       |        +--rw group-ip-address-location    IP-Multicast-Group-Address
       |        +--rw vrf?                         routing-instance-ref
       |        +--rw (technology)?
       |        |  +--:(technology-null)
       |        |  |  +--rw tech-null?                   empty
       |        |  +--:(technology-string)
       |        |     +--rw ipv4-icmp?                   string
       |        +--rw tp-tools
       |        |  +--rw connectivity-verification?   boolean
       |        |  +--rw continuity-check?            boolean
       |        |  +--rw path-discovery?              boolean
       |        +--rw root?
       |        +--rw oam-layers* [index]
       |           +--rw index                                 uint16
       |           +--rw level?                                int32
       |           +--rw (tp-address)?
       |              +--:(mac-address)
       |              |  +--rw mac-address-location?                yang:mac-
address
       |              +--:(ipv4-address)
       |              |  +--rw ipv4-location?                       inet:ipv4-
address
       |              +--:(ipv6-location)
       |              |  +--rw ipv6-address?                        inet:ipv6-
address
       |              +--:(tunnel-location)
       |              |  +--rw tunnel-location?                     uint32
       |              +--:(ip-prefix-location)
       |              |  +--rw ip-prefix-location?                  inet:ip-
prefix
       |              +--:(route-dist-location)
       |              |  +--rw route-dist-location?                 uint32
       |              +--:(group-ip-address-location)
       |              |  +--rw group-ip-address-location?           IP-
Multicast-Group-Address
       |              +--:(as-number-location)
       |              |  +--rw as-number-location?                  inet:as-
number
       |              +--:(lsp-id-location)
       |              |  +--rw lsp-id-location?                    string
       |              +--:(system-id-location)
       |              |  +--rw system-id-location?                 inet:uri
       |              +--:(connection-oriented-location)
       |                 +--rw maintenance-domain?                 -> /
goam:domains/domain/MD-name-string
       |                 +--rw maintenance-association?            -> /
goam:domains/domain[goam:MD-name-string = current()/../maintenance-domain]/MAs/
MA/MA-name-string
       |                 +--rw maintenance-association-end-point?   -> /
goam:domains/domain[goam:MD-name-string = current()/../maintenance-domain]/MAs/
MA[goam:MA-name-string = current()/../maintenance-association]/MEP/mep-name
       +--:(group-as-number-location-type)
       |  +--rw test-point-as-number-location-list
       |     +--rw test-point-locations* [as-number-location]
       |        +--rw as-number-location    inet:as-number
       |        +--rw vrf?                  routing-instance-ref
       |        +--rw (technology)?
       |        |  +--:(technology-null)
       |        |  |  +--rw tech-null?            empty
       |        |  +--:(technology-string)
       |        |     +--rw ipv4-icmp?            string
       |        +--rw tp-tools
       |        |  +--rw connectivity-verification?   boolean
       |        |  +--rw continuity-check?            boolean
       |        |  +--rw path-discovery?              boolean
       |        +--rw root?
       |        +--rw oam-layers* [index]
       |           +--rw index                                  uint16
       |           +--rw level?                                 int32
       |           +--rw (tp-address)?
       |              +--:(mac-address)
       |              |  +--rw mac-address-location?                  yang:mac-
address
       |              +--:(ipv4-address)
       |              |  +--rw ipv4-location?                         inet:ipv4-
address
       |              +--:(ipv6-location)
       |              |  +--rw ipv6-address?                          inet:ipv6-
address
       |              +--:(tunnel-location)
       |              |  +--rw tunnel-location?                       uint32
       |              +--:(ip-prefix-location)
       |              |  +--rw ip-prefix-location?                    inet:ip-
prefix
       |              +--:(route-dist-location)
       |              |  +--rw route-dist-location?                   uint32
       |              +--:(group-ip-address-location)
       |              |  +--rw group-ip-address-location?             IP-
Multicast-Group-Address
       |              +--:(as-number-location)
       |              |  +--rw as-number-location?                    inet:as-
number
       |              +--:(lsp-id-location)
       |              |  +--rw lsp-id-location?                       string
       |              +--:(system-id-location)
       |              |  +--rw system-id-location?                    inet:uri
       |              +--:(connection-oriented-location)
       |                 +--rw maintenance-domain?                    -> /
goam:domains/domain/MD-name-string
       |                 +--rw maintenance-association?               -> /
goam:domains/domain[goam:MD-name-string = current()/../maintenance-domain]/MAs/
MA/MA-name-string
```

```
       |                 +--rw maintenance-association-end-point?   -> /
goam:domains/domain[goam:MD-name-string = current()/../maintenance-domain]/MAs/
MA[goam:MA-name-string = current()/../maintenance-association]/MEP/mep-name
       +--:(group-lsp-id-location-type)
       |  +--rw test-point-lsp-id-location-list
       |     +--rw test-point-locations* [lsp-id-location]
       |        +--rw lsp-id-location    string
       |        +--rw vrf?               routing-instance-ref
       |        +--rw (technology)?
       |        |  +--:(technology-null)
       |        |  |  +--rw tech-null?         empty
       |        |  +--:(technology-string)
       |        |     +--rw ipv4-icmp?         string
       |        +--rw tp-tools
       |        |  +--rw connectivity-verification?   boolean
       |        |  +--rw continuity-check?            boolean
       |        |  +--rw path-discovery?              boolean
       |        +--rw root?
       |        +--rw oam-layers* [index]
       |           +--rw index                                uint16
       |           +--rw level?                               int32
       |           +--rw (tp-address)?
       |              +--:(mac-address)
       |              |  +--rw mac-address-location?                yang:mac-
address
       |              +--:(ipv4-address)
       |              |  +--rw ipv4-location?                       inet:ipv4-
address
       |              +--:(ipv6-location)
       |              |  +--rw ipv6-address?                        inet:ipv6-
address
       |              +--:(tunnel-location)
       |              |  +--rw tunnel-location?                     uint32
       |              +--:(ip-prefix-location)
       |              |  +--rw ip-prefix-location?                  inet:ip-
prefix
       |              +--:(route-dist-location)
       |              |  +--rw route-dist-location?                 uint32
       |              +--:(group-ip-address-location)
       |              |  +--rw group-ip-address-location?           IP-
Multicast-Group-Address
       |              +--:(as-number-location)
       |              |  +--rw as-number-location?                  inet:as-
number
       |              +--:(lsp-id-location)
       |              |  +--rw lsp-id-location?                     string
       |              +--:(system-id-location)
       |              |  +--rw system-id-location?                  inet:uri
       |              +--:(connection-oriented-location)
       |                 +--rw maintenance-domain?                  -> /
goam:domains/domain/MD-name-string
       |                 +--rw maintenance-association?             -> /
goam:domains/domain[goam:MD-name-string = current()/../maintenance-domain]/MAs/
MA/MA-name-string
       |                 +--rw maintenance-association-end-point?   -> /
goam:domains/domain[goam:MD-name-string = current()/../maintenance-domain]/MAs/
MA[goam:MA-name-string = current()/../maintenance-association]/MEP/mep-name
       +--:(group-system-id-location-type)
          +--rw test-point-system-info-location-list
```

```
             +--rw test-point-locations* [system-id-location]
                +--rw system-id-location    inet:uri
                +--rw vrf?                  routing-instance-ref
                +--rw (technology)?
                |  +--:(technology-null)
                |  |  +--rw tech-null?            empty
                |  +--:(technology-string)
                |     +--rw ipv4-icmp?            string
                +--rw tp-tools
                |  +--rw connectivity-verification?   boolean
                |  +--rw continuity-check?            boolean
                |  +--rw path-discovery?              boolean
                +--rw root?
                +--rw oam-layers* [index]
                   +--rw index                                uint16
                   +--rw level?                               int32
                   +--rw (tp-address)?
                      +--:(mac-address)
                      |  +--rw mac-address-location?                yang:mac-
address
                      +--:(ipv4-address)
                      |  +--rw ipv4-location?                       inet:ipv4-
address
                      +--:(ipv6-location)
                      |  +--rw ipv6-address?                        inet:ipv6-
address
                      +--:(tunnel-location)
                      |  +--rw tunnel-location?                     uint32
                      +--:(ip-prefix-location)
                      |  +--rw ip-prefix-location?                  inet:ip-
prefix
                      +--:(route-dist-location)
                      |  +--rw route-dist-location?                 uint32
                      +--:(group-ip-address-location)
                      |  +--rw group-ip-address-location?           IP-
Multicast-Group-Address
                      +--:(as-number-location)
                      |  +--rw as-number-location?                  inet:as-
number
                      +--:(lsp-id-location)
                      |  +--rw lsp-id-location?                     string
                      +--:(system-id-location)
                      |  +--rw system-id-location?                  inet:uri
                      +--:(connection-oriented-location)
                         +--rw maintenance-domain?                  -> /
goam:domains/domain/MD-name-string
                         +--rw maintenance-association?             -> /
goam:domains/domain[goam:MD-name-string = current()/../maintenance-domain]/MAs/
MA/MA-name-string
                         +--rw maintenance-association-end-point?   -> /
goam:domains/domain[goam:MD-name-string = current()/../maintenance-domain]/MAs/
MA[goam:MA-name-string = current()/../maintenance-association]/MEP/mep-name
```

data hierarchy of OAM

4. OAM YANG Module

```
   <CODE BEGINS> file "ietf-connectionless-oam.yang"
```

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

    prefix coam;

    import ietf-yang-schema-mount {
     prefix yangmnt;
    }

    import ietf-network{
      prefix nd;
    }
    import ietf-yang-types {
      prefix yang;
    }
    import ietf-interfaces {
      prefix if;
    }
    import ietf-inet-types {
      prefix inet;
    }
    import ietf-network-instance {
      prefix "ni";
    }
        import ietf-conn-oam{
          prefix "goam";
        }
    organization "IETF LIME Working Group";
    contact

        "Deepak Kumar dekumar@cisco.com
         Qin Wu       bill.wu@huawei.com
         S Raghavan   srihari@cisco.com
         Zitao Wang   wangzitao@huawei.com
         R Rahman     rrahman@cisco.com";

    description

      "This YANG module defines the generic configuration,
       data model, statistics for connectionless OAM to be
       used within IETF in a protocol indpendent manner.
       Functional level abstraction is indendent with
       YANG modeling. It is assumed that each protocol maps
       corresponding abstracts to its native format.
       Each protocol may extend the YANG model defined
       here to include protocol specific extensions";
    revision 2016-12-16 {
      description
        "Initial revision. - 08 version";
      reference "";
    }
    /* features */
    feature connection-less {
      description
        "this feature indicates that OAM solution is connection less.";
    }
    feature continuity-check {
      description
```

```
        "This feature indicates that the server supports
         executing continuity check OAM command and
         returning a response. Servers that do not advertise
         this feature will not support executing
         continuity check command or rpc model for
         continuity check command.";
    }
    feature path-discovery {
        description
        "This feature indicates that the server supports
         executing path discovery OAM command and
         returning a response. Servers that do not advertise
         this feature will not support executing
         path discovery command or rpc model for
         path discovery command.";
    }

    /* Identities */
    /* typedefs */
      typedef routing-instance-ref {

        type leafref {
          path "/ni:network-instances/ni:network-instance/ni:name";
        }
        description
        "This type is used for leafs that reference a routing instance
         configuration.";
      }

       typedef IPv4-Multicast-Group-Address {
            type string {
             pattern '(2((2[4-9])|(3[0-9]))\.)'
                         +'(([0-9]|[1-9][0-9]|1[0-9][0-9]|'
                         +'2[0-4][0-9]|25[0-5])\.){2}'
                         +'([0-9]|[1-9][0-9]|1[0-9][0-9]'
                         +'|2[0-4][0-9]|25[0-5])';
            }
            description
              "The IPv4-Multicast-Group-Address type
               represents an IPv4 multicast address
               in dotted-quad notation.";
            reference "RFC4607";
      } // typedef IPv4-Multicast-Group-Address
      typedef IPv6-Multicast-Group-Address {
            type string {
            pattern
                        '(((FF|ff)[0-9a-fA-F]{2}):)([0-9a-fA-F]'
                        +'{0,4}:){0,5}((([0-9a-fA-F]{0,4}:)?'
```

```
                        +'(:|[0-9a-fA-F]{0,4}))|(((25[0-5]|2[0-4]'
                        +'[0-9]|[01]?[0-9]?[0-9])\.){3}(25[0-5]|'
                        +'2[0-4][0-9]|[01]?[0-9]?[0-9])))';
            pattern
            '(([^:]+:){6}(([^:]+:[^:]+)|'
                        +'(.*\..*)))|((([^:]+:)*[^:]+)'
                        +'?::(([^:]+:)*[^:]+)?)';
             }
            description
                        "The IPv6-Multicast-Group-Address
                        type represents an IPv6 address in full,
                        mixed, shortened, and shortened-mixed
                        notation.";
            reference "RFC4291 2.7.
            ietf-inet-types:ipv6-address";
      }
      typedef IP-Multicast-Group-Address {
            type union {
              type IPv4-Multicast-Group-Address;
              type IPv6-Multicast-Group-Address;
            }

            description
             "The IP-Multicast-Group-Address type
              represents an IP multicast address and
              is IP version neutral. The format of the
              textual representations implies the IP version.";
      } // typedef IP-Multicast-Group-Address

    identity fec-types {

      description
      "This is base identity of fec types which are ip-prefix,
       bgp, tunnel, pwe3, vpls, etc.";
    }

    typedef fec-type {
      type identityref {
        base fec-types;
      }
      description "Target FEC type.";
    }

    typedef oam-counter32 {
      type yang:zero-based-counter32;
      description
        "defines 32 bit counter for OAM";
    }
```

```
    identity time-resolution{
      description
        "Time interval resolution";
    } //base identity

    identity hours {
      base time-resolution;
      description
        "Hours";
    }

    identity minutes {
      base time-resolution;
      description
        "Minutes";
    }

    identity seconds {
      base time-resolution;
      description
        "Seconds";

    }

    identity milliseconds {
      base time-resolution;
      description
        "Milliseconds";
    }

    identity microseconds {

      base time-resolution;
      description
        "Microseconds";
    }

    identity nanoseconds {
      base time-resolution;
      description
        "Nanoseconds";
    }

    /* groupings */
    grouping cc-session-statsitics {
      description "Grouping for session statistics.";
      container cc-session-statistics {
        description "cc session counters";
```

```
        leaf session-count {
           type uint32;
           description "Number of cc sessions.";
        }
        leaf session-up-count {
           type uint32;
           description "Number of sessions which are up.";
        }
        leaf session-down-count {
           type uint32;
           description "Number of sessions which are down.";
        }
        leaf session-admin-down-count {
           type uint32;
           description "Number of sessions which are admin-down.";
        }
      }
    }

    grouping session-packet-statistics {
      description "Grouping for per session packet statistics";
      container session-packet-statistics {

        description "Per session packet statistics.";
        leaf rx-packet-count {
          type uint32;
          description "Total received packet count.";
        }
        leaf tx-packet-count {
          type uint32;
          description "Total transmitted packet count.";
        }
        leaf rx-bad-packet {

          type uint32;
          description "Total received bad packet.";
        }
        leaf tx-packet-failed {
          type uint32;
          description "Total send packet failed.";
        }
      }
    }

    grouping cc-per-session-statistics {
      description "Grouping for per session statistics";
      container cc-per-session-statistics {
        description "per session statistics.";
```

```
        leaf create-time {
          type yang:date-and-time;
          description "Time and date when session is created.";
        }
        leaf last-down-time {
          type yang:date-and-time;
          description "Time and date last time session is down.";
        }
        leaf last-up-time {
          type yang:date-and-time;
          description "Time and date last time session is up.";
        }
        leaf down-count {
          type uint32;
          description "Total down count.";
        }
        leaf admin-down-count {
          type uint32;
          description "Total down count.";
        }

        uses session-packet-statistics;
      }

    }

    grouping session-error-statistics {
      description "Grouping for per session error statistics";
      container session-error-statistics {
        description "Per session error statistics.";
        leaf packet-drops-count {
          type uint32;
          description "Total received packet drops count.";
        }
        leaf packet-reorder-count {

          type uint32;
          description "Total received packet reordered count.";
        }
        leaf packets-out-of-seq-count {
          type uint32;
          description "Total received out of sequence count.";
        }
        leaf packets-dup-count {
          type uint32;
          description "Total received packet duplicates count.";
        }
      }
```

```
    }

    grouping session-delay-statistics {
      description "Grouping for per session delay statistics";
      container session-delay-statistics {
        description "Session delay summarised information.";
        leaf time-resolution-value {
          type identityref {
            base time-resolution;
          }
          description "Time units among choice of s,ms,ns etc.";
        }
        leaf min-delay-value {
          type uint32;
          description "Minimum delay value observed.";
        }
        leaf max-delay-value {
          type uint32;
          description "Maximum delay value observed.";
        }
        leaf average-delay-value {
          type uint32;
          description "Average delay value observed.";
        }

      }
    }

    grouping session-jitter-statistics {
      description "Grouping for per session jitter statistics";
      container session-jitter-statistics {
        description "Session jitter summarised information.";
        leaf time-resolution-value {
          type identityref {
            base time-resolution;
          }
          description "Time units among choice of s,ms,ns etc.";

        }
        leaf min-jitter-value {
          type uint32;
          description "Minimum jitter value observed.";
        }
        leaf max-jitter-value {
          type uint32;
          description "Maximum jitter value observed.";
        }
        leaf average-jitter-value {
```

```
          type uint32;
          description "Average jitter value observed.";
        }
      }
    }

    grouping session-path-verification-statistics {
      description "Grouping for per session path verification statistics";
      container session-path-verification-statistics{
        description "OAM per session path verification statistics.";
        leaf verified-count {
          type uint32;
          description "Total number of packets that went through a path as
intended.";
        }
        leaf failed-count {
          type uint32;
          description "Total number of packets that went through an unintended
path.";
        }
      }
    }

    grouping session-type {
      description
        "This object indicates the current session
         definition.";

      leaf session-type-enum {
        type enumeration {
          enum proactive {
            description
              "The current session is proactive";
          }
          enum on-demand {
            description
              "The current session is on-demand.";
          }
        }
        default "on-demand";
        description

          "session type enum";
      }
    }

    identity tp-address-type {
      description
```

```
          "Test point address type";
      } //base identity
```

```
    identity mac-address-type {
      base tp-address-type;
      description
        "MAC address type";
    }

    identity ipv4-address-type {
      base tp-address-type;
      description
        "IPv4 address type";
    }

    identity ipv6-address-type {
      base tp-address-type;
      description
        "IPv6 address type";
    }

    identity src-dst-address-type {
      base tp-address-type;
      description
        "Source/Dest address type";
    }

    identity fec-address-type {

      base tp-address-type;
      description
        "FEC address type";
    }

    identity tlv-address-type {
      base tp-address-type;
      description
        "TLV address type";
    }

    identity system-id-address-type {
      base tp-address-type;
      description

        "System id address type";
    }

    identity lsp-id-address-type {
      base tp-address-type;
      description
        "LSP ID address type";
```

```
    }

    identity as-number-address-type {
      base tp-address-type;
      description
        "AS number address type";
    }

    identity group-ip-address-type {
      base tp-address-type;
      description
        "Group IP address type";
    }

    identity route-distinguisher-address-type {
      base tp-address-type;
      description
        "Route Distinguisher address type";
    }

    identity ip-prefix-address-type {
      base tp-address-type;
      description
        "IP prefix address type";
    }

    identity tunnel-address-type {
      base tp-address-type;
      description
        "Tunnel address type";
    }

    grouping tp-address {
      leaf tp-address-type-value {
        type identityref {
          base tp-address-type;
        }
        description "Test point address type.";
      }

      choice tp-address {

        case mac-address {
          when "'tp-address-type-value' = 'mac-address-type'" {
            description "MAC address type";
          }
          leaf mac-address {
            type yang:mac-address;
```

```
            description
            "MAC Address";
          }
          description
          "MAC Address based MP Addressing.";
        }
        case ipv4-address {
          when "'tp-address-type-value' = 'ipv4-address-type'" {
            description "IPv4 address type";
          }
          leaf ipv4-address {
            type inet:ipv4-address;
            description
            "Ipv4 Address";
          }
          description
          "Ip Address based MP Addressing.";
        }
        case ipv6-address {
          when "'tp-address-type-value' = 'ipv6-address-type'" {
            description "IPv6 address type";
          }
          leaf ipv6-address {
            type inet:ipv6-address;
            description
            "Ipv6 Address";

          }
          description
          "ipv6 Address based MP Addressing.";
        }
        case src-dst-address {
          when "'tp-address-type-value' = 'src-dst-address-type'" {
            description "Src dest address type for BFD";
          }
          leaf src-ip-address {
            type inet:ip-address;
                    description
                    "source ip address.";
          }
          leaf dst-ip-address {
            type inet:ip-address;
                    description

                    "destination ip address.";
          }
          leaf Interface {
            type if:interface-ref;
```

```
                    description
                    "interface.";
          }
        }
        case fec {
          when "'tp-address-type-value' = 'fec-address-type'" {
            description "FEC address type";
          }
          leaf fec-type {
            type fec-type;
                    description
                    "fec type.";
          }
          choice fec-value {
                   description
                   "fec value.";
            case ip-prefix {
              leaf ip-prefix {
                type inet:ip-prefix;
                            description
                            "ip prefix.";
              }
            }
            case bgp {
              leaf bgp {
                type inet:ip-prefix;
                            description

                            "BGP Labeled Prefix ";
              }
            }
            case tunnel {
              leaf tunnel-interface {
                type uint32;
                            description
                            "VPN Prefix ";
              }
            }

            case pw {
              leaf remote-pe-address{
               type inet:ip-address;
                           description
                           "remote pe address.";
              }
              leaf pw-id {
                type uint32;
                            description
```

```
                              "Pseudowire id.";
                  }
                }
                case vpls {
                  leaf route-distinguisher {
                    type uint32;
                              description
                              "Route Distinguisher(8 octets).";
                  }
                  leaf sender-ve-id{
                   type uint32;
                             description
                             "Sender's VE ID.";
                  }
                  leaf receiver-ve-id{
                   type uint32;
                             description
                             "Receiver's VE ID.";
                  }
                }
                case mpls-mldp{
                 choice root-address{
                          description
                              "root address choice.";
                  case ip-address{
                   leaf source-address{
                    type inet:ip-address;
                              description

                              "ip address.";
                   }
                   leaf group-ip-address{
                    type IP-Multicast-Group-Address;
                              description
                              "group ip address.";
                   }
                  }
                  case vpn{
                   leaf as-number{
                    type inet:as-number;

                              description
                              "AS number.";
                   }
                  }
                  case global-id{
                   leaf lsp-id{
                    type string;
```

```
                            description
                            "lsp id.";
                 }
                }
               }
              }
             }
            }
            case tlv-address {
              when "'tp-address-type-value' = 'tlv-address-type'" {
                description "TLV address type";
              }
               leaf tlv-type {
                 type int16;
                 description
                   "Type of MEP-ID";
               }
               leaf tlv-len {
                 type int16;
                 description
                   "Length of MEP-ID value";
               }
               leaf tlv-value {
                 type binary {
                   length "12..255";
               }
               description
                 "Value please refer RFC6428 (Figure 4,5,6).";
               }

            description
              "MEP-ID";
            }
            case system-info {
              when "'tp-address-type-value' = 'system-id-address-type'" {
                description "System id address type";
              }
              leaf system-id {
                type inet:uri;
                description
                  "System ID assigned to this node.";
              }

            }
            description
              "TP Addressing.";
          }
          description
```

```
        "TP Address";
    }

    grouping tp-address-vrf {
      description
        "Test point address with VRF.";
      leaf vrf {
        type routing-instance-ref;
            description
            "The vrf is used to describe the
             corresponding network instance";
      }

      uses tp-address;
    }

    grouping connectionless-oam-layers {
      list oam-layers {
         key "index";
         leaf index {
             type uint16 {
                  range "0..65535";
             }
             description
               "Index";
         }
        leaf level {
            type int32 {
                 range "-1..1";
            }

            default 0;
            description
              "Level 0 indicates default level, -1 means server
               and +1 means client layer.
               In relationship 0 means same layer.";
        }
                choice tp-address {

        case mac-address {
          leaf mac-address-location {
            type yang:mac-address;
            description
            "MAC Address";
          }
          description
          "MAC Address based MP Addressing.";
        }
```

```
        case ipv4-address {
          leaf ipv4-location {
            type inet:ipv4-address;
            description
            "Ipv4 Address";
          }
          description
          "Ip Address based MP Addressing.";
        }
        case ipv6-location {
          leaf ipv6-address {
            type inet:ipv6-address;
            description
            "Ipv6 Address";

          }
          description
          "ipv6 Address based MP Addressing.";
        }

        case tunnel-location{
         leaf tunnel-location{
                  type uint32;
                  description
                  "VPN Prefix";
                 }
                 description
                 "tunnel location";
                 }

                 case ip-prefix-location{
                  leaf ip-prefix-location{
                   type inet:ip-prefix;
                   description
                   "Ip prefix location";
                  }
                  description
                  "IP prefix location";
                 }

                 case route-dist-location{
                  leaf route-dist-location{
                   type uint32;
                   description
                   "Route Distinguisher (8 octets)";
                  }
                  description
                  "Route distinguisher location";
```

```
                 }

                 case group-ip-address-location{
                  leaf group-ip-address-location{
                   type IP-Multicast-Group-Address;
                   description
                   "Group IP address location";
                  }
                  description
                  "Group IP address";
                 }

                 case as-number-location{
                  leaf as-number-location{
                   type inet:as-number;
                   description
                   "AS number location";
                  }
                  description
                  "AS number";
                 }

                 case lsp-id-location{
                  leaf lsp-id-location{
                   type string;
                   description
                   "LSP id";
                  }
                  description
                  "LSP ID";
                 }

                 case system-id-location{
                  leaf system-id-location{
                   type inet:uri;
                   description
                   "system id location";
                  }
                  description
                  "System ID";
                 }

                 case connection-oriented-location{
                  uses goam:maintenance-association-end-point-reference;
                  description
                  "connection oriented location";
                 }
        description
```

```
          "TP location.";

          }
        ordered-by user;
         description
            "list of related oam layers.
             0 means they are in same level, especially
             interworking scenarios of stiching multiple
             technology at same layer.
             -1 means server layer, for eg:- in case of

             Overlay and Underlay, Underlay is server layer for
             Overlay Test Point.
             +1 means client layer, for eg:- in case of
             Service OAM and Transport OAM, Service OAM is client
             layer to Transport OAM.";
      }
      description
        "connectionless related OAM layer";
    }

    grouping tp-technology {
       choice technology {
          default technology-null;
          case technology-null {
            description
              "this is a placeholder when no technology is needed.";
            leaf tech-null {
               type empty;
               description
                 "there is no technology define";
            }
          }
          description
            "technology choice null";
          case technology-string {
            description
            "oam technology string";
            leaf ipv4-icmp {
              type string;
              description
                "name to identify oam technology";
            }
         }
       }

       description
        "OAM Technology";
```

```
    }

    grouping tp-tools {
      description
        "Test Point OAM Toolset.";
      container tp-tools{
           leaf connectivity-verification{
            type boolean;
                description
                "A flag indicating whether or not the
                connectivity-verification function is supported.";
                reference
         "RFC 792: INTERNET CONTROL MESSAGE PROTOCOL.
                  RFC 4443: Internet Control Message Protocol (ICMPv6)
          for the Internet Protocol Version 6 (IPv6) Specification.
                  RFC 5880: Bidirectional Forwarding Detection.
                  RFC 5881: BFD for IPv4 and IPv6.
                  RFC 5883: BFD for Multihop Paths.
                  RFC 5884: BFD for MPLS Label Switched Paths.
                  RFC 5885: BFD for PW VCCV.
                  RFC 4379: LSP-PING.";
           }
           leaf continuity-check{
            type boolean;
                description
                "A flag indicating whether or not the
                continuity check function is supported.";
                reference
         "RFC 792: INTERNET CONTROL MESSAGE PROTOCOL.
                  RFC 4443: Internet Control Message Protocol (ICMPv6)
          for the Internet Protocol Version 6 (IPv6) Specification.
                  RFC 5880: Bidirectional Forwarding Detection.
                  RFC 5881: BFD for IPv4 and IPv6.
                  RFC 5883: BFD for Multihop Paths.
                  RFC 5884: BFD for MPLS Label Switched Paths.
                  RFC 5885: BFD for PW VCCV.";
           }
           leaf path-discovery{
            type boolean;
                description
                "A flag indicating whether or not the
                path discovery function is supported.";
            reference
         "RFC 792: INTERNET CONTROL MESSAGE PROTOCOL.
                  RFC 4443: Internet Control Message Protocol (ICMPv6)
          for the Internet Protocol Version 6 (IPv6) Specification.
                  RFC 4884: Extended ICMP to Support Multi-part Message.
                  RFC 5837:Extending ICMP for Interface and Next-Hop
```

Identification.

```
                 RFC 4379: LSP-PING.";
          }
          description
          "Container for test point OAM tools set.";
         }

    }

    grouping test-point-location-info {
      uses tp-technology;
      uses tp-tools;
      anydata root {
       yangmnt:mount-point root;
       description
           "Root for models supported per
        test point";
      }
      uses connectionless-oam-layers;
      description
        "Test point Location";
    }

    grouping test-point-locations {
      description "Group of test point locations.";
      leaf tp-address-type-value {

        type identityref {
          base tp-address-type;
        }
        description "Test point address type.";

      }
      choice location-type {
        case ipv4-location-type {
          when "'tp-address-type-value' = 'ipv4-address-type'" {
          description
            "when test point address is equal to ipv4 address.";
          }
          container test-point-ipv4-location-list {
            list test-point-locations {
              key "ipv4-location";
              leaf ipv4-location {
                type inet:ipv4-address;
                description
                  "Ipv4 Address.";
              }
              leaf vrf {
                type routing-instance-ref;
```

```
                description
                "The vrf is used to describe the
                 corresponding network instance";
          }

          uses test-point-location-info;

          ordered-by user;
          description
            "list of test point locations.";
        }
        description
          "Serves as top-level container for test point location list.";
      }
    }
    case ipv6-location-type {
      when "'tp-address-type-value' = 'ipv6-address-type'" {
      description
        "when test point address is equal to ipv6 address";
      }
      container test-point-ipv6-location-list {
        list test-point-locations {
          key "ipv6-location";
          leaf ipv6-location {
            type inet:ipv6-address;
            description
              "Ipv6 Address.";
          }
          leaf vrf {

            type routing-instance-ref;
                description
                "The vrf is used to describe the

                 corresponding network instance";
          }
          uses test-point-location-info;
          ordered-by user;
          description
            "list of test point locations.";
        }
        description
          "Serves as top-level container for test point location list.";
      }
    }
    case mac-location-type {
      when "'tp-address-type-value' = 'mac-address-type'" {
        description
```

```
              "when test point address is equal to mac address.";
          }
          container test-point-mac-address-location-list {
            list test-point-locations {
            key "mac-address-location";
            leaf mac-address-location {
              type yang:mac-address;
              description
                "MAC Address";
            }
            uses test-point-location-info;
            ordered-by user;
            description
             "list of test point locations.";
            }
            description
              "Serves as top-level container for test point location list.";
          }
        }
        case tunnel-location-type {
          when "'tp-address-type-value' = 'tunnel-address-type'" {
            description
              "when test point address is equal to tunnel type.";
          }
          container test-point-tunnel-address-location-list {
            list test-point-locations {
            key "tunnel-location";
            leaf tunnel-location {
              type uint32;
              description
                "VPN Prefix";

            }
            leaf vrf {

              type routing-instance-ref;
                  description
                  "The vrf is used to describe the
                   corresponding network instance";
            }
            uses test-point-location-info;
            ordered-by user;
            description
             "list of test point locations.";
            }
            description
              "Serves as top-level container for test point location list.";
          }
```

```
        }
        case ip-prefix-location-type {
          when "'tp-address-type-value' = 'ip-prefix-address-type'" {
            description
              "when test point address is equal to ip prefix.";
          }
          container test-point-ip-prefix-location-list {
            list test-point-locations {
            key "ip-prefix-location";
            leaf ip-prefix-location {
              type inet:ip-prefix;
              description
                "IP Prefix";
            }
            leaf vrf {
              type routing-instance-ref;
                  description
                  "The vrf is used to describe the
                   corresponding network instance";
            }
            uses test-point-location-info;
            ordered-by user;
            description
             "list of test point locations.";
            }
            description
              "Serves as top-level container for test point location list.";
          }
        }
        case route-distinguisher-location-type {
          when "'tp-address-type-value' = 'route-distinguisher-address-type'" {
            description "when test point address is equal to
            route distinguiher.";

          }

          container test-point-route-dist-location-list {
            list test-point-locations {
            key "route-dist-location";
            leaf route-dist-location {
              type uint32;
              description
                "Route Distinguisher(8 octets).";
            }
            leaf vrf {
              type routing-instance-ref;
                  description
                  "The vrf is used to describe the
```

```
                   corresponding network instance";
            }
            uses test-point-location-info;
            ordered-by user;
            description
             "list of test point locations.";
            }
            description
              "Serves as top-level container for test point location list.";
          }
        }
        case group-ip-address-location-type {
          when "'tp-address-type-value' = 'group-ip-address-type'" {
            description "when test point address is equal to
                         group ip address.";
          }
          container test-point-group-ip-address-location-list {
            list test-point-locations {
            key "group-ip-address-location";
            leaf group-ip-address-location {
              type IP-Multicast-Group-Address;
              description
                "Group IP address.";
            }
            leaf vrf {
              type routing-instance-ref;
                  description
                  "The vrf is used to describe the
                   corresponding network instance";
            }
            uses test-point-location-info;
            ordered-by user;
            description
             "list of test point locations.";
            }

            description
              "Serves as top-level container for test point location list.";
          }
        }
        case group-as-number-location-type {
          when "'tp-address-type-value' = 'as-number-address-type'" {
            description "when test point address is equal to
                         as-number.";
          }
          container test-point-as-number-location-list {
            list test-point-locations {
            key "as-number-location";
```

```
            leaf as-number-location {
              type inet:as-number;
              description
                "AS number.";
            }
            leaf vrf {
              type routing-instance-ref;
                  description
                  "The vrf is used to describe the
                   corresponding network instance";
            }
            uses test-point-location-info;
            ordered-by user;
            description
             "list of test point locations.";
            }
            description
              "Serves as top-level container for test point location list.";
          }
        }
        case group-lsp-id-location-type {
          when "'tp-address-type-value' = 'lsp-id-address-type'" {
            description "when test point address is equal to lspid.";
          }
          container test-point-lsp-id-location-list {
            list test-point-locations {
            key "lsp-id-location";
            leaf lsp-id-location {
              type string;
              description
                "LSP Id.";
            }
            leaf vrf {
              type routing-instance-ref;
                  description
                  "The vrf is used to describe the

                   corresponding network instance";
            }
            uses test-point-location-info;
            ordered-by user;
            description
             "list of test point locations.";
            }
            description
              "Serves as top-level container for test point location list.";
          }
        }
```

```
        case group-system-id-location-type {
          when "'tp-address-type-value' = 'system-id-address-type'" {
            description "when test point address is equal to
                         system info.";
          }
          container test-point-system-info-location-list {
            list test-point-locations {
            key "system-id-location";
            leaf system-id-location {
              type inet:uri;
              description
                "System Id.";
            }
            leaf vrf {
              type routing-instance-ref;
                  description
                  "The vrf is used to describe the
                   corresponding network instance";
            }
            uses test-point-location-info;
            ordered-by user;
            description
              "list of test point locations.";
            }
            description
              "Serves as top-level container for test point location list.";
          }
        }
        description
          "Choice of address types.";
      }
    }

    augment "/nd:networks/nd:network/nd:node"{
      description
        "Augment test points of connectionless oam.";
      uses test-point-locations;

    }

    grouping path-discovery-data {
      description "Path discovery related data output from nodes.";
      container src-test-point {
        description "Source test point.";
        uses tp-address-vrf;
      }
      container dest-test-point {
        description "Destination test point.";
```

```
          uses tp-address-vrf;
        }
        leaf sequence-number {
          type uint64;
          description "Sequence number in data packets.";
        }
        leaf hop-cnt {
          type uint8;
          description "hop count.";
        }

        uses session-packet-statistics;
        uses session-error-statistics;
        uses session-delay-statistics;
        uses session-jitter-statistics;

        container path-verification {
          description "Optional path verification related information.";
          leaf flow-info {
            type string;
            description
              "ACL name that refers to the flow, if any.";
          }
          uses session-path-verification-statistics;
        }

        container path-trace-info {
          description "Optional path trace per-hop test point information.
                       The list has typically a single element for per-hop
                       cases like path-discovery RPC but allows a list of
                       hop related information for other types of
                       data retrieval methods.";
          list path-trace-info-list {
            key "index";
            description
              "Path trace information list.";
            leaf index {
              type uint32;

              description "Trace information index.";
            }

            uses tp-address-vrf;

            leaf timestamp-val {
              type yang:date-and-time;
              description "Timestamp value";
            }
```

```
          leaf ingress-intf-name {
            type if:interface-ref;
            description
              "Ingress interface name";
          }
          leaf egress-intf-name {
            type if:interface-ref;
            description
              "Egress interface name";
          }
          leaf app-meta-data {
            type uint32;
            description
              "Application specific data added by node.";
          }
        }
      }
    }

    grouping continuity-check-data {
      description "Continuity check data output from nodes.";
      container src-test-point {
        description "Source test point.";
        uses tp-address-vrf;

        leaf egress-intf-name {
          type if:interface-ref;
          description
            "Egress interface name";
        }
      }
      container dest-test-point {
        description "Destination test point.";
        uses tp-address-vrf;

        leaf ingress-intf-name {
          type if:interface-ref;
          description
            "Ingress interface name";

        }
      }
      leaf sequence-number {
        type uint64;
        description "Sequence number.";
      }
      leaf hop-cnt {
        type uint8;
```

```
        description "hop count.";
      }

      uses session-packet-statistics;
      uses session-error-statistics;
      uses session-delay-statistics;
      uses session-jitter-statistics;
    }

    container oper {
      if-feature continuity-check;
      config "false";
      description "cc operational information.";
      container cc-ipv4-sessions-statistics {
        description "cc ipv4 sessions";
        uses cc-session-statsitics;
      }
      container cc-ipv6-sessions-statistics {
        description "cc ipv6 sessions";
        uses cc-session-statsitics;
      }
    }
}
```

YANG module of OAM

<CODE ENDS>

5. CL model applicability

ietf-connectionless-oam model defined in this document provides technology-independent abstraction of key OAM constructs for connectionless protocols. This model can be further extended to include technology specific details, e.g., adding new data nodes with technology specific functions and parameters into proper anchor points of the base model, so as to develop a technology-specific connectionless OAM model.

This section demonstrates the usability of the connectionless YANG OAM data model to various connectionless OAM technologies, e.g., BFD, LSP ping. Note that, in this section, we only present several snippets of technology-specific model extensions for illustrative purposes. The complete model extensions should be worked on in respective protocol working groups.

5.1. BFD Extension

5.1.1. Augment Method

The following sections shows how the "ietf-connectionless-oam" model can be extended to cover BFD technology. For this purpose, a set of extension are introduced such as technology-type extension and test-point attributes extension.

Note that in BFD WG, there is a BFD yang data model [I-D.ietf-bfd-yang] to be produced. Users can choose to use "ietf-connectioless-oam" as basis and augment the "ietf-connectionless-oam" model with bfd specific details. The bfd specific details can be the grouping defined in the BFD model.

5.1.1.1. technology type extension

No BFD technology type has been defined in the "ietf-connectionless-oam" model. Therefore a technology type extension is required in the model Extension.

The snippet below depicts an example of augmenting "bfd" type into the ietf-connectionless-oam":

```
augment "/nd:networks/nd:network/nd:node/"
+"coam:location-type/coam:ipv4-location-type"
+"/coam:test-point-ipv4-location-list/"
       +"coam:test-point-locations/coam:technology"
+"/coam:technology-string"
{
    leaf bfd{
   type string;
  }
}
```

5.1.1.2. test point attributes extension

To support bfd technology, the "ietf-connectionless-oam" model can be extended and add bfd specific parameters under "test-point-location" list and/or add new location type such as "bfd over MPLS-TE" under "location-type".

5.1.1.2.1. Define and insert new nodes into corresponding test-point-location

In the "ietf-connectionless-oam" model, multiple "test-point-location" lists are defined under the "location-type" choice node. Therefore, to derive a model for some bfd technologies (such as ip

single-hop, ip multi-hops, etc), data nodes for bfd specific details need to be added into corresponding "test-point-locations" list. In this section, we reuse some groupings which are defined in [I-D.ietf-bfd-yang] as following:

The snippet below shows how the "ietf-connectionless-oam" model can be extended to support "BFD IP single-hop":

```
augment "/nd:networks/nd:network/nd:node/"
+"coam:location-type/coam:ipv4-location-type"
+"/coam:test-point-ipv4-location-list/"
        +"coam:test-point-locations"
{
        container session-cfg {
          description "BFD IP single-hop session configuration";
          list sessions {
            key "interface dest-addr";
            description "List of IP single-hop sessions";
            leaf interface {
              type if:interface-ref;
              description
                "Interface on which the BFD session is running.";
            }
            leaf dest-addr {
              type inet:ip-address;
              description "IP address of the peer";
            }
            uses bfd:bfd-grouping-common-cfg-parms;
            uses bfd:bfd-grouping-echo-cfg-parms;
          }
        }
}
```

Similar augmentations can be defined to support other BFD technologies such as BFD IP multi-hop, BFD over MPLS, etc.

5.1.1.2.2. Add new location-type cases

In the "ietf-connectionless-oam" model, If there is no appropriate "location type" case that can be extended, a new "location-type" case can be defined and inserted into the "location-type" choice node.

Therefore, the model user can flexibly add "location-type" to support other type of test point which are not defined in the "ietf-connectionless-oam" model. In this section, we add a new "location-type" case and reuse some groupings which are defined in [I-D.ietf-bfd-yang] as follows:

The snippet below shows how the "ietf-connectionless-oam" model can be extended to support "BFD over MPLS-TE":

```
augment "/nd:networks/nd:network/nd:node/coam:location-type"{
 case te-location{
  list test-point-location-list{
   key "tunnel-name";
   leaf tunnel-name{
    type leafref{
 path "/te:te/te:tunnels/te:tunnel/te:name";
}
description
"point to a te instance.";
   }
    uses bfd:bfd-grouping-common-cfg-parms;
        uses bfd-mpls:bfd-encap-cfg;
  }
 }
}
```

Similar augmentations can be defined to support other BFD technologies such as BFD over LAG, etc.

5.1.2. Schema Mount

And anohter alternative method is using schema mount mechanism [draft-ietf-netmod-schema-mount] in the "ietf-connectionless-oam". Within the "test-point-location" list, a "root" attribute is defined to provide a mounted point for models mounted per "test-point-location". Therefore, the "ietf-connectionless-oam" model can provide a place in the node hierarchy where other OAM YANG data models can be attached, without any special extension in the "ietf-connectionless-oam" YANG data models [draft-ietf-netmod-schema-mount]. Note that the limitation of the Schema Mount method is it is not allowed to specify certain modules that are required to be mounted under a mount point.

The snippet below depicts the definition of "root" attribute.

```
      anydata root {
       yangmnt:mount-point root;
       description
      "Root for models supported per
        test point";
      }
```

The following section shows how the "ietf-connectionless-oam" model can use schema mount to support BFD technology.

5.1.2.1. BFD Modules be populated in schema-mount

To support BFD technology, "ietf-bfd-ip-sh" and "ietf-bfd-ip-mh" YANG modules might be populated in the "schema-mounts" container:

```
<schema-mounts
    xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-schema-mount">
  <mount-point>
    <module> ietf-connectionless-oam </module>
    <name>root</name>
    <use-schema>
      <name>root</name>
    </use-schema>
  </mount-point>
  <schema>
    <name>root</name>
    <module>
      <name>ietf-bfd-ip-sh </name>
      <revision>2016-07-04</revision>
      <namespace>
        urn:ietf:params:xml:ns:yang: ietf-bfd-ip-sh
      </namespace>
      <conformance-type>implement</conformance-type>
    </module>
    <module>
      <name>ietf-bfd-ip-mh </name>
      <revision> 2016-07-04</revision>
      <namespace>
        urn:ietf:params:xml:ns:yang: ietf-bfd-ip-mh
      </namespace>
      <conformance-type>implement</conformance-type>
    </module>
  </schema>
</schema-mounts>
```

and the " ietf-connectionless-oam " module might have:

```
<ietf-connectionless-oam uri="urn:ietf:params:xml:ns:yang:ietf-connectionless-
oam">
   ......
 <test-point-locations>
  <ipv4-location> 1.1.1.1</ipv4-location>
   ......
  <root>
   <ietf-bfd-ip-sh uri="urn:ietf:params:xml:ns:yang:ietf-bfd-ip-sh">
    <ip-sh>
     foo
     ......
    </ip-sh>
   </ietf-bfd-ip-sh>
   <ietf-bfd-ip-mh uri="urn:ietf:params:xml:ns:yang:ietf-bfd-ip-mh">
    <ip-mh>
     foo
     ......
    </ip-mh>
   </ietf-bfd-ip-mh>
  </root>
 </test-point-locations>
</ietf-connectionless-oam>
```

5.2. LSP ping extension

The following sections shows how the "ietf-connectionless-oam" model can be extended to support LSP ping technology. For this purpose, a set of extension are introduced such as technology-type extension and test-point attributes extension.

Note that in MPLS WG, there is a LSP Ping yang data model [I-D.draft-zheng-mpls-lsp-ping-yang-cfg] to be produced. Users can choose to use "ietf-connectioless-oam" as basis and augment the "ietf-connectionless-oam" model with LSP Ping specific details in the model extension. The LSP Ping specific details can be the grouping defined in the LSP ping model.

5.2.1. technology type extension

No lsp-ping technology type has been defined in the "ietf-connectionless-oam" model. Therefore a technology type extension is required in the model extension.

The snippet below depicts an example of augmenting the "ietf-connectionless-oam" with "lsp-ping" type:

```
   augment "/nd:networks/nd:network/nd:node/"
   +"coam:location-type/coam:ipv4-location-type"
   +"/coam:test-point-ipv4-location-list/"
           +"coam:test-point-locations/coam:technology"
   +"/coam:technology-string"
   {
      leaf lsp-ping{
      type string;
     }
   }
```

5.2.2. test point attributes extension

To support lsp-ping, the "ietf-connectionless-oam" model can be extended and add lsp-ping specific parameters can be defined and under "test-point-location" list.

User can reuse the attributes or groupings which are defined in [I-D.draft-zheng-mpls-lsp-ping-yang-cfg] as follows:

The snippet below depicts an example of augmenting the "test-point-locations" list with lsp ping attributes:

```
   augment "/nd:networks/nd:network/nd:node/"
   +"coam:location-type/coam:ipv4-location-type"
   +"/coam:test-point-ipv4-location-list/"
           +"coam:test-point-locations"
   {
   list lsp-ping {
            key "lsp-ping-name";
            leaf lsp-ping-name {
             type string {
               length "1..31";
            }
           mandatory "true";
           description "LSP Ping test name.";
           ......
         }
```

6. Security Considerations

TBD.

7. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688] [RFC3688]. Following the format in RFC 3688, the following registration is requested to be made:

URI: urn:ietf:params:xml:ns:yang:ietf-connectionless-oam

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

This document registers a YANG module in the YANG Module Names registry [RFC6020].

name: ietf-connectionless-oam namespace: urn:ietf:params:xml:ns:yang:ietf-connectionless-oam
prefix: goam reference: RFC XXXX

8. Acknowlegements

The authors of this document would like to thank Greg Mirskey and others for their sustainable review and comments, proposals to improve and stablize document.

9. Normative References

[I-D.draft-ietf-i2rs-yang-network-topo]
Clemm, A., Medved, J., Tkacik, T., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", I-D draft-ietf-i2rs-yang-network-topo-05, July 2016.

[I-D.draft-zheng-mpls-lsp-ping-yang-cfg]
Zheng, L., Aldrin, S., Zheng, G., Mirsky, G., and R. Rahman, "Yang Data Model for LSP-PING", I-D draft-zheng-mpls-lsp-ping-yang-cfg-03, March 2016.

[I-D.ietf-bfd-yang]
Zheng, L., Rahman, R., Networks, J., Jethanandani, M., and G. Mirsky, "Yang Data Model for Bidirectional Forwarding Detection (BFD)", draft-ietf-bfd-yang-03 (work in progress), July 2016.

[I-D.ietf-netmod-schema-mount]
Bjorklund, M. and L. Lhotka, "YANG Schema Mount", draft-ietf-netmod-schema-mount-03 (work in progress), October 2016.

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

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

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

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

[RFC792] Postel, J., "Internet Control Message Protocol", RFC 792, September 1981.

Authors' Addresses

Deepak Kumar
CISCO Systems
510 McCarthy Blvd
Milpitas, CA 95035
USA

Email: dekumar@cisco.com

Michael Wang
Huawei Technologies,Co.,Ltd
101 Software Avenue, Yuhua District
Nanjing 210012
China

Email: wangzitao@huawei.com

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing, Jiangsu 210012
China

Email: bill.wu@huawei.com

Reshad Rahman
CISCO Systems
2000 Innovation Drive
KANATA, ONTARIO K2K 3E8
CANADA

Email: rrahman@cisco.com

Srihari Raghavan
CISCO Systems
TRIL INFOPARK SEZ, Ramanujan IT City
NEVILLE BLOCK, 2nd floor, Old Mahabalipuram Road
CHENNAI, TAMIL NADU 600113
INDIA

Email: srihari@cisco.com