NETMOD
Internet Draft
Intended status: Standards Track

Tissa Senevirathne
Norman Finn
Deepak Kumar
Samer Salam
Cisco

March 29, 2014

Expires: September 2014

YANG Data Model for Operations Administration and Maintenance (OAM)
draft-tissa-netmod-oam-00.txt

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on September 31, 2014.

Copyright Notice

Abstract

This document presents YANG Data model for OAM. It provides protocol and technology independent abstraction of key OAM concepts required for OAM. These abstractions span OAM configuration and operational data; they promote uniformity between OAM technologies and support nested OAM work flows (i.e. drilling down to OAM at different layers) through a unified interface.

Table of Contents

1. Introduction

Operations, Administration and Maintenance (OAM) are important aspect of networking and allow operators to:

1. Configure networks

2. Monitor networks

3. Troubleshoot failures (Fault verification and isolation).

Ping and Traceroute are well known fault verification tools in IP world. Over the years different technologies have developed similar tools for similar purposes.

[8021Q] Connectivity Fault Management is a well-established OAM standard that is widely adopted. ITUT [Y1731], MEF, MPLS-TP [RFC6371], TRILL [TRILLOAM] all define OAM methods based on [8021Q] CFM.

Given the wide spread adoption of the underlying OAM concepts defined in [8021Q] CFM it is a reasonable choice to develop the unified OAM framework based on those concepts. In this document we take the [8021Q] CFM model and extend it to a technology independent framework and build the corresponding YANG model accordingly.

The unification of OAM, according to the proposal of this document, occurs at the management layer. Encapsulations and state machines may differ according to each protocol. A user who wishes to issues a Ping command or a Traceroute or initiate a session monitoring can do so in the same manner regardless of the underlying protocol or technology.

As an example, consider a scenario where an IP ping to device B from Device A failed. Between device A and B there are IEEE 802.1 bridges a,b and c. Let's assume a,b and c are using [8021Q] CFM. A user upon detecting the IP layer ping failure, may desire to drop down to the Ethernet layer and issue the corresponding fault verification (LBM) and fault isolation (LTM) tools, using the same API. This ability to go up and down to different layers for troubleshooting is also known as "nested OAM" and is a very powerful concept that leads to efficient network troubleshooting and maintenance workflows. The OAM YANG model presented in this document facilitates that without needing changes to the underlying protocols.

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 RFC-2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

2.1. Terminology

CCM - Continuity Check Message [8021Q]

ECMP - Equal Cost Multipath

LBM - Loopback Message [8021Q]

MP - Maintenance Point [8021Q]

MEP - Maintenance End Point [TRLOAMFRM] [8021Q] [RFC6371]

MIP - Maintenance Intermediate Point [TRLOAMFRM] [8021Q] [RFC6371]

MA - Maintenance Association [8021Q] [TRLOAMFRM]

MD - Maintenance Domain [8021Q]

MTV - Multi-destination Tree Verification Message

OAM - Operations, Administration, and Maintenance [RFC6291]

TRILL - Transparent Interconnection of Lots of Links [RFC6325]

3. Overview of the OAM Model

In this document we adopt [8021Q] CFM model and structure it such that it can be adapted to different technologies.

At the top of the Model is the Maintenance Domain. Each Maintenance Domain is associated with a Maintenance Name and a Domain Level.

Under each Maintenance Domain there is one or more Maintenance Association (MA). In IP the MA can be per IP Subnet, in NVO3 this can be per VNI and for TRILL this can be per Fine-Grained Label or for VPLS this can be per VPLS instance.

Under each MA, there can be two or more MEP (Maintenance End Points). MEPs are addressed by their respective technology specific addressing identifiers. The YANG model presented here provides flexibility to accommodate different addressing schemes.

In a parallel vertical, presented are the commands. Those, in YANG terms, are the rpc commands. These rpc commands provide uniform APIs for ping, traceroute and their equivalents as well as other OAM commands.

3.1. Maintenance Domain (MD) configuration

The container "domains" is the top level container within the ietf-oam module. Within the container "domains", separate list is maintained per MD. The MD list uses the key MD-name for indexing.

```
module: ietf-oam
   +--rw domains
   |  +--rw domain* [md-name]
   |     +--rw technology        identityref
   |     +--rw md-name-format    MD-name-format
   |     +--rw md-name           binary
   |     +--rw md-level          int32

   .
   .
```

Figure 1 Snippet of data hierarchy related to OAM domains

3.2. Maintenance Association (MA) configuration

Within a given Maintenance Domain there can be one or more Maintenance Associations (MA). MAs are represented as a list and indexed by the MA-name.

```
module: ietf-oam
   +--rw domains
   |  +--rw domain* [md-name]
   |     +--rw technology        identityref
   |     +--rw md-name-format    MD-name-format
   |     +--rw md-name           binary
   |     +--rw md-level          int32
   |     +--rw MAs!
   |        +--rw MA* [ma-name]
   |           +--rw ma-name-format        MA-name-format
   |           +--rw ma-name               binary
   .
   .
```

Figure 2 Snippet of data hierarchy related to Maintenance Associations (MA).

3.3. Maintenance Endpoint (MEP) configuration

Within a given Maintenance Association (MA), there can be one or more Maintenance End Points (MEP). MEPs are represented as a list within the data hierarchy and indexed by the key MEP-id.

```
module: ietf-oam
   +--rw domains
   |  +--rw domain* [md-name]
   |     +--rw technology        identityref
   |     +--rw md-name-format    MD-name-format
   |     +--rw md-name           binary
   |     +--rw md-level          int32
   |     +--rw MAs!
   |        +--rw MA* [ma-name]
   |           +--rw ma-name-format        MA-name-format
   |           +--rw ma-name               binary

   .
   .
   |           +--rw MEP* [mep-id]
   |           |  +--rw mep-id           MEP-id
   |           |  +--rw mep-name?        string
   |           |  +--rw mep-direction    MEP-direction
   |           |  +--rw context-id?      uint32
   .
   .
```

Figure 3 Snippet of data hierarchy related to Maintenance Endpoint (MEP).

3.4. rpc definitions

The rpc model facilitates issuing commands to a NETCONF server (in this case to the device that need to execute the OAM command) and obtaining a response. rpc model defined here abstracts OAM specific commands in a technology independent manner.

There are several rpc commands defined for the purpose of OAM. In this section we present a snippet of the ping command for illustration purposes. Please refer to Section 4 for the complete data hierarchy and Section 5 for the YANG model.

```
module: ietf-oam
   +--rw domains
   |  +--rw Domain* [MA-domain-name]
   |     +--rw technology               technology
   |     +--rw MA-domain-name-format    int32
   |     +--rw MA-domain-name           binary
   |     +--rw MD-level                 int32

   .
   .

   rpcs:
   +---x ping
   |  +--ro input
   |  |  +--ro technology             identityref
   |  |  +--ro md-name-format         MD-name-format
   |  |  +--ro md-name?               binary
   |  |  +--ro md-level               int32
   |  |  +--ro ma-name-format         MA-name-format
   |  |  +--ro ma-name                binary
   |  |  +--ro source-mep-id?         MEP-id
   |  |  +--ro destination-mepid?     MEP-id
   |  |  +--ro ttl?                   uint8
   |  |  +--ro flow-entropy?          binary
   |  |  +--ro ecmp-choice?           ecmp-choices
   |  |  +--ro outgoing-interfaces*   [interface]
   |  |     +--ro interface           if:interface-ref
   |  +--ro output
   |     +--ro tx-packt-count?    yang:zero-based-counter32
   |     +--ro rx-packet-count?   yang:zero-based-counter32
   |     +--ro min-delay?         yang:zero-based-counter32
   |     +--ro average-delay?     yang:zero-based-counter32
   |     +--ro max-delay?         yang:zero-based-counter32
```

Figure 4 Snippet of data hierarchy related to rpc call Ping

4. OAM data hierarchy

The complete data hierarchy related to the OAM YANG model is presented below. The following notations are used within the data tree and carry the meaning as below.

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
```

```
module: ietf-oam
   +--rw domains
   |  +--rw domain* [md-name]
   |     +--rw technology        identityref
   |     +--rw md-name-format    MD-name-format
   |     +--rw md-name           binary
   |     +--rw md-level          int32
   |     +--rw MAs!
   |        +--rw MA* [ma-name]
   |           +--rw ma-name-format        MA-name-format
   |           +--rw ma-name               binary
   |           +--rw ccm-Interval?         CCM-Interval
   |           +--rw ccm-loss-threshold?   uint32
   |           +--rw ccm-ttl?              uint8
   |           +--rw MEP* [mep-id]
   |           |  +--rw mep-id           MEP-id
   |           |  +--rw mep-name?        string
   |           |  +--rw mep-direction    MEP-direction
   |           |  +--rw context-id?      uint32
   |           |  +--rw ccm-Tx-enable?   boolean
   |           |  +--rw flow-entropy?    binary
   |           |  +--rw mep-address      MEP-address
   |           |  +--rw Interface?       if:interface-ref
   |           |  +--rw session* [user-cookie destination-mepid]
   |           |     +--rw user-cookie           uint32
   |           |     +--rw destination-mepid     MEP-id
   |           |     +--rw ttl?                  uint8
   |           |     +--rw interval?             uint32
   |           |     +--rw enable?               boolean
   |           |     +--rw flow-entropy?         binary
   |           |     +--rw ecmp-choice?          ecmp-choices
   |           |     +--rw outgoing-interface* [interface]
   |           |        +--rw interface    leafref
   |           +--rw remote-MEP* [mep-id]
   |              +--rw mep-id               uint32
   |              +--rw mep-name?            string
   |              +--rw ccm-rx-error-count?   oam-counter32
   +--ro domain-status
      +--ro Domain* [md-name]
         +--ro technology        identityref
         +--ro md-name-format    MD-name-format
         +--ro md-name           binary
         +--ro md-level          int32
         +--ro MAs!
            +--ro ccm-rdi-indicator?    boolean
            +--ro ccm-xcon-count?       oam-counter32
```

```
               +--ro ccm-xcon-Indicator?   boolean
               +--ro MA* [ma-name]
                  +--ro ma-name-format    MA-name-format
                  +--ro ma-name           binary
                  +--ro MEP* [mep-id]
                  |  +--ro ccm-rdi-indicator?        boolean
                  |  +--ro ccm-xcon-count?           oam-counter32
                  |  +--ro ccm-xcon-Indicator?       boolean
                  |  +--ro mep-id                    MEP-id
                  |  +--ro mep-name?                 string
                  |  +--ro interface-oper-status?    leafref
                  |  +--ro interface-admin-status?   leafref
                  |  +--ro session* [user-cookie destination-mepid]
                  |     +--ro user-cookie          uint32
                  |     +--ro destination-mepid    MEP-id
                  |     +--ro session-id?          uint16
                  |     +--ro tx-packt-count?      oam-counter32
                  |     +--ro rx-packet-count?     oam-counter32
                  |     +--ro min-delay?           oam-counter32
                  |     +--ro average-delay?       oam-counter32
                  |     +--ro max-delay?           oam-counter32
                  +--ro remote-MEP* [mep-id]
                     +--ro mep-id                uint32
                     +--ro mep-name?             string
                     +--ro ccm-rdi-indicator?    boolean
                     +--ro ccm-xcon-count?       oam-counter32
                     +--ro ccm-xcon-Indicator?   boolean
   rpcs:
      +---x ping
      |  +--ro input
      |  |  +--ro technology             identityref
      |  |  +--ro md-name-format         MD-name-format
      |  |  +--ro md-name?               binary
      |  |  +--ro md-level               int32
      |  |  +--ro ma-name-format         MA-name-format
      |  |  +--ro ma-name                binary
      |  |  +--ro source-mep-id?         MEP-id
      |  |  +--ro destination-mepid?     MEP-id
      |  |  +--ro ttl?                   uint8
      |  |  +--ro flow-entropy?          binary
      |  |  +--ro ecmp-choice?           ecmp-choices
      |  |  +--ro outgoing-interfaces* [interface]
      |  |     +--ro interface    if:interface-ref
      |  +--ro output
      |     +--ro tx-packt-count?    oam-counter32
      |     +--ro rx-packet-count?   oam-counter32
      |     +--ro min-delay?         oam-counter32
```

```
      |     +--ro average-delay?     oam-counter32
      |     +--ro max-delay?         oam-counter32
      +---x trace-route
      |  +--ro input
      |  |  +--ro technology             identityref
      |  |  +--ro md-name-format         MD-name-format
      |  |  +--ro md-name?               binary
      |  |  +--ro md-level               int32
      |  |  +--ro ma-name-format         MA-name-format
      |  |  +--ro ma-name                binary
      |  |  +--ro source-mepid?          MEP-id
      |  |  +--ro destination-mepid?     MEP-id
      |  |  +--ro flow-entropy?          binary
      |  |  +--ro ecmp-choice?           ecmp-choices
      |  |  +--ro outgoing-interfaces* [interface]
      |  |     +--ro interface    if:interface-ref
      |  +--ro output
      |     +--ro response* [ttl]
      |        +--ro ttl                uint8
      |        +--ro remote-mepid?      MEP-id
      |        +--ro tx-packt-count?    oam-counter32
      |        +--ro rx-packet-count?   oam-counter32
      |        +--ro min-delay?         oam-counter32
      |        +--ro average-delay?     oam-counter32
      |        +--ro max-delay?         oam-counter32
      +---x End-station-locator
         +--ro input
         |  +--ro technology             identityref
         |  +--ro md-name-format         MD-name-format
         |  +--ro md-name?               binary
         |  +--ro md-level               int32
         |  +--ro ma-name-format         MA-name-format
         |  +--ro ma-name                binary
         |  +--ro source-mepid?          MEP-id
         |  +--ro end-station-address?   End-station-address
         |  +--ro context-identifier?    Context-identifier
         +--ro output
            +--ro devices* [mep-id]
               +--ro mep-id      MEP-id
               +--ro mep-name?   string
   notifications:
      +---n CCM-RDI-notification
         +--ro mep-id?          MEP-id
         +--ro remote-mepid?    MEP-id
         +--ro error-message?   String
```

Figure 5 data hierarchy of OAM

5. OAM YANG module

```
<CODE BEGINS> file "xxx.yang"

module ietf-oam {
  yang-version 1;

  namespace "urn:cisco:params:xml:ns:yang:ietf-oam";
  // name space urn later to be replaced with IANA
  // assigned URI

  prefix ietf-oam;

  import ietf-inet-types {
    prefix inet;
  }
  import ietf-interfaces {
    prefix if;
  }
  import ietf-yang-types {
    prefix yang;
  }

  organization "IETF NETMOD (NETCONF Data Modeling ) Working Group";
  contact
    "Tissa Senevirathne tsenevir@cisco.com";
  description
    "This YANG module defines the generic configuration,
     statistics and rpc for 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.";

  revision 2014-03-28 {
    description
      "Initial revision.";
    reference "draft-tissa-netmod-oam";
  }

  /*
   * identity definitions,
   */

  identity technology-types {
    description
      "this is the base identy of technology types which are
```

```
          vpls, nvo3, TRILL, ipv4, ipv6, mpls";
      }

      identity vpls {
        base technology-types;
        description
          "vpls technology type";
      }

      identity nvo3 {
        base technology-types;
        description
          "nvo3 type";
      }

      identity trill {
        base technology-types;
        description
          "trill type";
      }

      identity ipv4 {
        base technology-types;
        description
          "technology of ipv4";
      }

      identity ipv6 {
        base technology-types;
        description
          "technology of ipv6";
      }

      /*
       * typedef definitions
       */

      typedef MEP-direction {
        type enumeration {
          enum "Up" {
            value 0;
          }
          enum "Down" {
            value 1;
          }
        }
      }
```

```
    typedef vlan {
      type uint16 {
        range "1..4094";
      }
    }

    typedef vni {
      type uint32;
    }

    typedef vpls-id {
      type uint32;
    }

    typedef TRILL-nickname {
      type uint16 {
        range "1..65531";
      }
    }

    typedef MEP-address {
      type union {
        type TRILL-nickname;
        type inet:ipv4-address;
        type inet:ipv6-address;
      }
      description
        "Defines addresses of different MEP types. IPv4, IPv6,
         RBridge nickname";
    }

    typedef End-station-address {
      type union {
        type yang:mac-address;
        type inet:ipv4-address;
        type inet:ipv6-address;
      }
      description
        "Defines addresses of different End stations, MAC, IPv4 or
  IPv6";
      }

    typedef MEP-id {
      type uint32 {
        range "1..8191";
      }
```

```
      description
        "Defines type for MEPIDm range is 1..8191";
    }

    typedef Context-identifier {
      type union {
        type vlan;
        type vni;
        type vpls-id;
        type uint32;
      }
      description
        "defines context identifier types VLAN, VNI, VPLS instence
  etc..";
    }

    typedef CCM-Interval {
      default "interval-1min";
      type enumeration {
        enum "interval-invalid" {
          value 0;
        }
        enum "interval-300hz" {
          value 1;
        }
        enum "interval-10ms" {
          value 2;
        }
        enum "interval-100ms" {
          value 3;
        }
        enum "interval-1s" {
          value 4;
        }
        enum "interval-10s" {
          value 5;
        }
        enum "interval-1min" {
          value 6;
        }
        enum "interval-10min" {
          value 7;
        }
      }
      reference
        "802.2Q Rev5 or 802.ag, all of the above
         are standard enumeration from the 802.1Q";
```

```
       description
         "IntervalInvalid - value 0
          Interval300Hz - Value 1
          Intervale10ms - value 2
          Interval100ms - value3
          Interval1s    - value 4
          Interval10s  - value 5
          Interval1min - value 6
          Interval10min - value 7";
     }

     typedef ecmp-choices {
       type enumeration {
         enum "ecmp-use-platform-hash" {
           value 0;
         }
         enum "ecmp-use-round-robin" {
           value 1;
         }
       }
     }

     typedef MD-name-format {
       type enumeration {
         enum "ieee-reserved" {
           value 0;
         }
         enum "none" {
           value 1;
         }
         enum "dns-like-name" {
           value 2;
         }
         enum "mac-address-and-uint" {
           value 3;
           reference "802.1Q Rev5";
           description
             "Domain name 3 specifies domain name is mac-address + 2
   octets.";
         }
       }
       reference "802.1Q";
       description
         "defines the domain name format";
     }

     typedef MA-name-format {
```

```
       type enumeration {
         enum "ieee-reserved" {
           value 0;
         }
         enum "primary-vid" {
           value 1;
         }
         enum "char-string" {
           value 2;
         }
         enum "unsigned-int16" {
           value 3;
         }
         enum "rfc2865-vpnid" {
           value 4;
         }
       }
       reference "802.1Q";
       description
         "Defines Format of MA-names";
     }

     typedef flow-entropy {
       type binary {
         length "96";
       }
     }

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

     /*
      * grouping definitions
      */

     grouping maintenance-domain {
       status current;
       description
         "Defines the MA-domain group";
       reference "802.1Q Rev5";
       leaf technology {
         mandatory true;
         status current;
         type identityref {
```

```
          base technology-types;
        }
        description
          "Defines the technology";
      }
      leaf md-name-format {
        mandatory true;
        status current;
        description
          "Defines the maintenance domain name";
        type MD-name-format;
        reference "802.1Q Rev5";
      }
      leaf md-name {
        status current;
        description
          "Defines the MA-Domain name. This is a binary (octet) string
           of 43 bytes";
        type binary {
          length "1..43";
        }
        reference "802.1Q Rev5";
      }
      leaf md-level {
        mandatory true;
        status current;
        description
          "Defines the MD-Level";
        type int32 {
          range "0..7";
        }
        reference "802.1Q Rev5 or 802.1ag";
      }
    }

    grouping ma-identifier {
      description
        "ma-identifier defines MAID parameters as defined in 8021Q";
      reference "IEEE 802.1Q Rev5";
      leaf ma-name-format {
        mandatory true;
        status current;
        description
          "This defines the MA name format 1 is no format,
           2 - dnslikename, 3- macaddress 4-CharString";
        type MA-name-format;
        reference "IEEE 802.1Q Rev 5";
```

```
      }
      leaf ma-name {
        mandatory true;
        description
          "Define the MA-Name according to the specified format.
           This is 43 byte string.";
        type binary {
          length "1..45";
        }
        reference "802.1Q Rve 5 or 8021ag Clause 21.6.5";
      }
    }

    grouping MEP {
      status current;
      description
        "Defines elements withing the MEP";
      reference "802.1Q Rev5";
      leaf mep-id {
        mandatory true;
        status current;
        description
          "Assigm MEPID in the range of 1..8191";
        type MEP-id {
          range "1..8191";
        }
        reference "802.1Q Rev5";
      }
      leaf mep-name {
        type string;
        description
          "Defines textual name for MEP. This is not specified in IEEE
  but
           defined in IETF OAM for ease of use";
      }
      leaf mep-direction {
        type MEP-direction;
        mandatory true;
      }
      leaf context-id {
        type uint32;
        description
          "This contain VLAN-ID, VNI-ID, VPLS-ID on which MEP is
  applied";
      }
      leaf ccm-Tx-enable {
        type boolean;
```

```
        default "false";
      }
      leaf flow-entropy {
        type binary {
          length "96";
        }
      }
      leaf mep-address {
        type MEP-address;
        mandatory true;
      }
      leaf Interface {
        type if:interface-ref;
        description
          "Interface name as defined by ietf-interfaces";
      }
    }

    grouping CCM-defect-stats {
      description
        "Contains all of the CCM related defect stats";

      leaf ccm-rdi-indicator {
        config false;
        type boolean;
        description
          "True indicate one or more of the MEP have seen RDI
           flag set from remote MEP";
      }

      leaf ccm-xcon-count {
        config false;
        type oam-counter32;
        description
          "Number of times cross connect errors are seen";
      }
      leaf ccm-xcon-Indicator {
        config false;
        type boolean;
        description
          "There is currently cross connect error seen since last
           clearing of the variable";
      }
    }

    grouping monitor-stats {
      leaf tx-packt-count {
        type oam-counter32;
```

description

```
          "Transmitted Packet count";
      }
      leaf rx-packet-count {
        type oam-counter32;
        description
          "Received packet count";
      }
      leaf min-delay {
        units "milliseconds";
        type oam-counter32;
        description
          "Delay is specified in milliseconds";
      }
      leaf average-delay {
        units "milliseconds";
        type oam-counter32;
        description
          "average delay in milliseconds";
      }
      leaf max-delay {
        type oam-counter32;
        units "millisecond";
      }
    }

    /*
     * below config data definitions
     */

    container domains {
      status current;
      config true;
      description
        "Contains configuration related data. Within the container
         is list of fault domains. Wihin each domian has List of MA.";
      list domain {
        uses maintenance-domain {
          status current;
        }
        key "md-name";
        ordered-by system;
        status current;
        config true;
        description
          "Define the list of Domains within the IETF-OAM";
        container MAs {
          presence
```

```
            "Indicates creation of MA within the Domain
             There can be more than one MA within a specified domain";

          status current;
          config true;
          description
            "This container defines MA, within that have multiple MA
             and within MA have MEP, MIP";

          list MA {
            ordered-by system;
            status current;
            config true;
            key "ma-name";
            uses ma-identifier;
            leaf ccm-Interval {
              status current;
              description
                "Defines CCM Interval 0- Means disable
                 1 - CCM are sent 3 1/3 ms
                 2 - CCM are sent every 10 ms
                 3- CCM are sent every 100 ms
                 4- CCM are sent every 1 s
                 5 - CCM are sent every 10 s
                 6 - CCM are sent every 1 minute
                 7- CCM are sent every 10 mins";
              type CCM-Interval;
              reference "802.1Q Rev5 and 802.1ag";
            }
            leaf ccm-loss-threshold {
              default "3";
              type uint32;
              description
                "number of consecutive CCM messages missed before
                 declaring RDI fault. This is monitored per each
                 remote MEP";
            }
            leaf ccm-ttl {
              type uint8;
              default "255";
            }
            list MEP {
              key "mep-id";
              ordered-by system;
              status current;
              config true;
              description
```

```
        "contain list of MEPS";
      uses MEP {
        status current;
      }
      list session {
        key "user-cookie destination-mepid";
        ordered-by user;
        config true;
        description
          "per session basis create the monitoring";
        leaf user-cookie {
          config true;
          type uint32;
          description
            "user need to specify some cookie to identify
             multiple sessions between to MEPs";
        }
        leaf destination-mepid {
          type MEP-id;
          config true;
        }
        leaf ttl {
          config true;
          type uint8;
          default "255";
        }
        leaf interval {
          units "milliseconds";
          default "1000";
          type uint32;
          description
            "In milli seconds. 0 means continous";
        }
        leaf enable {
          default "false";
          config true;
          type boolean;
          description
            "enable or disable a monitor session";
        }
        leaf flow-entropy {
          type binary {
            length "96";
          }
        }
        leaf ecmp-choice {
          config true;
```

```
                type ecmp-choices;
                description
                  "0 means use the specified interface
                   1 means use round robin";
              }
              list outgoing-interface {
                config true;
                key "interface";
                leaf interface {
                  type leafref {
                    path "/if:interfaces/if:interface/if:name";
                  }
                  config true;
                }
              }
            }
          }
          list remote-MEP {
            key "mep-id";
            ordered-by system;
            status current;
            config true;
            description
              "list all of the remote MEP within the MA";
            leaf mep-id {
              mandatory true;
              status current;
              description
                "Assigm MEPID in the range of 1..8191";
              config true;
              type uint32;
              reference "802.1Q Rev5";
            }
            leaf mep-name {
              type string;
              description
                "Defines textual name for MEP. This is not
                 specified in IEEE but defined in IETF OAM
                 for ease of use";
            }
            leaf ccm-rx-error-count {
              type oam-counter32;
              description
                "counts number of CCM packets that was
                 expected but not received";
            }
          }
```

```
          }
        }
      }
    } // end of container domain that defines config data

    /*
     * read only data definitions below
     */
    container domain-status {
      config false;
      description
        "This container carries status and statistics";
      list Domain {
        uses maintenance-domain {
          status current;
        }
        key "md-name";
        ordered-by system;
        status current;
        description
          "Define the list of Domains within the IETF-OAM";
        container MAs {
          presence
            "Indicates creation of MA within the Domain
             There can be more than one MA within a specified domain";
          status current;
          description
            "This container defines MA, within that have
             multiple MA and within MA have MEP, MIP";

          uses CCM-defect-stats;
          list MA {
            ordered-by system;
            status current;
            uses ma-identifier;
            key "ma-name";
            list MEP {
              key "mep-id";
              ordered-by system;
              status current;
              description
                "contain list of MEPS";
              uses CCM-defect-stats;
              leaf mep-id {
                type MEP-id;
              }
              leaf mep-name {
```

```
                type string;
                description
                  "Defines textual name for MEP. This is not
                   specified in IEEE but
                   defined in IETF OAM for ease of use";
              }
              leaf interface-oper-status {
                type leafref {
                  path
                   "/if:interfaces-state/if:interface/if:oper-status";
                }
              }
              leaf interface-admin-status {
                type leafref {
                  path
                    "/if:interfaces-state/if:interface/if:oper-status";
                }
              }
              list session {
                key "user-cookie destination-mepid";
                ordered-by user;
                description
                  "";
                leaf user-cookie {
                  type uint32;
                }
                leaf destination-mepid {
                  type MEP-id;
                }
                leaf session-id {
                  type uint16;
                  description
                    "This is system generated key to uniquely identify
                     the session. May be useful for isolation of system
                     specific problems or further trouble shooting";
                }
                uses monitor-stats;
              }
            }
            list remote-MEP {
              key "mep-id";
              ordered-by system;
              status current;
              description
                "list all of the remote MEP within the MA";
              leaf mep-id {
                mandatory true;
```

```
                status current;
                description
                  "Assigm MEPID in the range of 1..8191";
                type uint32;
                reference "802.1Q Rev5";
              }
              leaf mep-name {
                type string;
                description
                  "Defines textual name for MEP. This is not specified
                   in IEEE but
                   defined in IETF OAM for ease of use";
              }
              uses CCM-defect-stats {
                description
                  "indicates CCM defects received from this
                   remote MEP (end point)";
              }
            }
          }
        }
      } // end of domain-status

      /*
       * below is definition of notifications
       */

      notification CCM-RDI-notification {
        description
          "When RDI is received this notificiation is sent";
        leaf mep-id {
          type MEP-id;
          description
            "Indicate which MEP is seeing the error";
        }
        leaf remote-mepid {
          type MEP-id;
          description
            "Who is seeing the error (if known) if unknown make it 0.";
        }
        leaf error-message {
          type string {
            length "0..255";
          }
          description
            "Error message to indicate more details.";
```

```
      }
    } // end of notification

    /*
     * below is definitions of rpc commands
     */

    rpc ping {
      description
        "Generates Ping and return response";
      input {
        uses maintenance-domain {
          description
            "Specifies the MA-domain";
        }
        uses ma-identifier {
          description
            "identfies the Maintenance association";
        }
        leaf source-mep-id {
          type MEP-id;
        }
        leaf destination-mepid {
          type MEP-id;
        }
        leaf ttl {
          type uint8;
          default "255";
        }
        leaf flow-entropy {
          type binary {
            length "96";
          }
        }
        leaf ecmp-choice {
          type ecmp-choices;
          description
            "0 means use the specified interface
             1 means use round robin";
        }
        list outgoing-interfaces {
          key "interface";
          leaf interface {
            type if:interface-ref;
          }
        }
      }
```

```
      output {
        uses monitor-stats {
          description
            "Stats of Ping is same as that of monitor sessions";
        }
      }
    } // end of rpc command ping

    rpc trace-route {
      description
        "Generates Trace-route and return response. Starts with TTL
         of one and increment by one at each hop. Untill destination
         reached or TTL reach max valune";
      input {
        uses maintenance-domain {
          description
            "Specifies the MA-domain";
        }
        uses ma-identifier {
          description
            "identfies the Maintenance association";
        }
        leaf source-mepid {
          type MEP-id;
        }
        leaf destination-mepid {
          type MEP-id;
        }
        leaf flow-entropy {
          type binary {
            length "96";
          }
        }
        leaf ecmp-choice {
          type ecmp-choices;
          description
            "0 means use the specified interface
             1 ECMP selection according to the platform ECMP selection
             algorithm";
        }
        list outgoing-interfaces {
          key "interface";
          leaf interface {
            type if:interface-ref;
          }
        }
      }
```

```
      output {
        list response {
          key "ttl";
          leaf ttl {
            type uint8;
          }
          leaf remote-mepid {
            type MEP-id;
          }
          uses monitor-stats;
        }
      }
    } // end of rpc command traceroute

    rpc End-station-locator {
      description
        "Allows to discover where the end station is located.";
      input {
        uses maintenance-domain {
          description
            "Specifies the MA-domain";
        }
        uses ma-identifier {
          description
            "identfies the Maintenance association";
        }
        leaf source-mepid {
          type MEP-id;
        }
        leaf end-station-address {
          type End-station-address;
          description
            "End station address can be MAC address, IPv4 or IPv6
  address.";
        }
        leaf context-identifier {
          type Context-identifier;
          description
            "This can be either vni, vlan , vpls or any other
  applicable
             conext";
        }
      }
      output {
        list devices {
          key "mep-id";
          leaf mep-id {
```

```
            type MEP-id;
          }
          leaf mep-name {
            type string;
            description
              "End-station locator response MAY return the textual name
               of MEP that owns the end-station.
               If textual name is not available word Unknown SHOULD
               be returned";
          }
        }
      }
    } // end rpc end-station-locator
  }// end ietf-oam module


  <CODE ENDS>
```

Figure 6 YANG module of OAM

6. Security Considerations

TBD

7. IANA Considerations

This document registers the following namespace URI in the IETF XML registry.

URI:TBD

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2234] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

[8021Q] IEEE, "Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks", IEEE Std 802.1Q-2011, August, 2011.

8.2. Informative References

[RFCabab] Faber, T., Touch, J. and W. Yue, "The TIME-WAIT state in TCP and Its Effect on Busy Servers", Proc. Infocom 1999 pp. 1573-1583.

[Y1731] ITU, "OAM functions and mechanisms for Ethernet based networks", ITU-T G.8013/Y.1731, July, 2011.

[TRLOAMFRM] Salam, S., et.al., "TRILL OAM Framework", draft-ietf-trill-oam-framework, Work in Progress, November, 2012.

[RFC6291] Andersson, L., et.al., "Guidelines for the use of the "OAM" Acronym in the IETF" RFC 6291, June 2011.

[RFC6325] Perlman, R., et.al., "Routing Bridges (RBridges): Base Protocol Specification", RFC 6325, July 2011.

9. Acknowledgments

Giles Heron came up with the idea of developing a YANG model as a way of creating a unified OAM API set (interface), work in this document is largely an inspiration of that. Alexander Clemm provided many valuable tips, comments and remarks that helped to refine the YANG model presented in this document.

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Tissa Senevirathne
CISCO Systems
375 East Tasman Drive.
San Jose, CA 95134
USA.

Phone: 408-853-2291
Email: tsenevir@cisco.com

Norman Finn
CISCO Systems
510 McCarthy Blvd
Milpitas, CA 95035.

Email: nfinn@cisco.com

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

Email: dekumar@cisco.com

Samer Salam
CISCO Systems
595 Burrard St. Suite 2123
Vancouver, BC V7X 1J1, Canada

Email: ssalam@cisco.com