CCAMP Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 3, 2018

H. Zheng
Z. Fan
Huawei Technologies
A. Sharma
Google
R. Rao
Infinera
S. Belotti
Nokia
V. Lopez
Telefonica
Y. Li
China Mobile
Y. Xu
CAICT

October 30, 2017

OTN Tunnel YANG Model
draft-ietf-ccamp-otn-tunnel-model-01

Abstract

This document describes the YANG data model for OTN Tunnels.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 3, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction 2
2. Terminology and Notations 2
3. Model Overview 3
3.1. Mux Service in Multi-Domain OTN Network 3
3.2. Bookended and Non-BookEnded OTN Tunnel 4
3.3. Network and Client side tunnel services 4
3.4. OTN Tunnel YANG Tree 4
3.5. OTN Tunnel YANG Code 8
3.6. OTN Types YANG Code 18
4. Security Considerations 26
5. IANA Considerations 26
6. Acknowledgements 26
7. Contributors 26
8. Normative References 27
Authors' Addresses 27

1. Introduction

OTN transport networks can carry various types of client services. In many cases, the client signal is carried over an OTN tunnel across connected domains in a multi-domain network. These OTN services can either be transported or switched in the OTN network. If an OTN tunnel is switched, then additional parameters need to be provided to create a Mux OTN service.

This document provides YANG model for creating OTN tunnel. The model augments the TE Tunnel model, which is an abstract model to create TE Tunnels.

2. Terminology and Notations

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in the YANG data tree presented later in this draft is defined in [I-D.ietf-netmod-yang-tree-diagrams]. They are provided below for reference.

- o Brackets "[" and "]" enclose list keys.
- o Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).
- o Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

3. Model Overview

3.1. Mux Service in Multi-Domain OTN Network

```
                              OTN Mux Service
                   <---------------------------------->

     XXXXXXXXXXX          XXXXXXXXXXXXXXXXXXXXXXXXX         XXXXXXXXX
    XX         XX       XX                        XX      XX         XX
   XX   +---+    X      XX +---+           +---+    X     XX  +---+   XX
   X    |NE1+--------+-----+NE2+-----------+NE3+--------+-X---+NE4|    X
   X    +---+   XX   ^  X  +---+           +---+   XX   ^ XX  +---+    X
    XX         XX    |   XX                        XX    |  XX        XX
     XXXXXXXXXXX     |    XXXXXXXXXXXXXXXXXXXXXXXXX     |   XXXXXXXXX
       Domain-1      |            Domain-2              |    Domain-3
                     +                                  +
         Same OTN Service attributes:      Same OTN Service attributes:
         1. Client Signal                  1. Client Signal
         2. Tributary Port Number          2. Tributary Port Number
         3. Tributary Slot Granularity     3. Tributary Slot Granularity
         4. Tributary Slots                4. Tributary Slots
```

Figure 1: OTN Mux Service in a multi-domain network topology

Figure 1 shows a multi-domain OTN network with three domains. In this example, user wants to setup an end-to-end OTN service that passes through Domain-2. In order to create an OTN mux service in Domain-2, user will need to specify the exact details of the client side LO-ODU on NE2 and NE3, so that these service endpoints can be paired with the LO-ODU endpoints on NE1 and NE4, respectively.

Let's assume that ODU4 is the client side HO-ODU on NE2 and NE3, and the client signal is ODU2. User will need to specify the OTN client signal (ODU2 in this example), the Tributary Port Number (TPN), Tributary Slot Granularities (TSG) and tributary slots to be used. As shown in the figure above, these service parameters must be the same between NE1 and NE2, and NE3 and NE4.

Once the OTN Mux service is setup in Domain-2, the incoming signal from either NE1 and/or NE4 will be switched inside Domain-2, and delivered to NE at the other end.

3.2. Bookended and Non-BookEnded OTN Tunnel

OTN tunnel model provides support for both bookended and non-bookended OTN tunnels.

For bookended tunnels, the same client signal is present on source and destination endpoints. For example, ODU2e bookended tunnel will have the same ODU2e client signal at both source and destination endpoints.

For non-bookended tunnels, different client signals are present on source and destination endpoints. For example, the client signal can be ODU2e on the source endpoint and the handoff at the destination can be 10GbE-LAN client signal.

3.3. Network and Client side tunnel services

The OTN tunnel model provides support for both network to network and client to client tunnels. For network to network tunnel, network termination points on source and destination node represent source and destination endpoints. For client to client tunnel, client termination points on source and destination node represent source and destination endpoints.

If a client to client tunnel needs to use one or more HO (or server) network to network tunnels, ERO and routing constraints, defined in the base TE model, can be used to route the client tunnel over one or more server tunnels.

3.4. OTN Tunnel YANG Tree

```
module: ietf-otn-tunnel
  augment /te:te/te:tunnels/te:tunnel:
    +--rw payload-treatment?          enumeration
    +--rw src-client-signal?          identityref
```

```
    +--rw src-tpn?                    uint16
    +--rw src-tsg?                    identityref
    +--rw src-tributary-slot-count?   uint16
    +--rw src-tributary-slots
    |  +--rw values*   uint8
    +--rw dst-client-signal?          identityref
    +--rw dst-tpn?                    uint16
    +--rw dst-tsg?                    identityref
    +--rw dst-tributary-slot-count?   uint16
    +--rw dst-tributary-slots
       +--rw values*   uint8

  rpcs:
    +---x otn-te-tunnel-path-compute
       +---w input
       |  +---w request* [id]
       |     +---w id                          uint8
       |     +---w type?                       identityref
       |     +---w source?                     inet:ip-address
       |     +---w destination?                inet:ip-address
       |     +---w src-tp-id?                  binary
       |     +---w dst-tp-id?                  binary
       |     +---w switching-layer?            identityref
       |     +---w encoding?                   identityref
       |     +---w protection-type?            identityref
       |     +---w restoration-type?           identityref
       |     +---w provider-id?                te-types:te-global-id
       |     +---w client-id?                  te-types:te-global-id
       |     +---w te-topology-id?             te-types:te-topology-id
       |     +---w setup-priority?             uint8
       |     +---w hold-priority?              uint8
       |     +---w te-path-metric-type?        identityref
       |     +---w odu-type?                   identityref
       |     +---w p2p-primary-paths
       |     |  +---w p2p-primary-path* [name]
       |     |     +---w name                      string
       |     |     +---w te-default-metric?        uint32
       |     |     +---w te-delay-metric?          uint32
       |     |     +---w te-hop-metric?            uint32
       |     |     +---w explicit-route-objects
       |     |        +---w explicit-route-object* [index]
       |     |           +---w explicit-route-usage?   identityref
       |     |           +---w index                   uint32
       |     |           +---w (type)?
       |     |              +--:(numbered)
       |     |              |  +---w numbered-hop
       |     |              |     +---w address?    te-types:te-tp-id
       |     |              |     +---w hop-type?   te-hop-type
```

```
        |     |              +--:(as-number)
        |     |              |  +---w as-number-hop
        |     |              |     +---w as-number?   binary
        |     |              |     +---w hop-type?    te-hop-type
        |     |              +--:(unnumbered)
        |     |              |  +---w unnumbered-hop
        |     |              |     +---w node-id?      te-types:te-node-id
        |     |              |     +---w link-tp-id?   te-types:te-tp-id
        |     |              |     +---w hop-type?     te-hop-type
        |     |              +--:(label)
        |     |              |  +---w label-hop
        |     |              |     +---w value?   rt-types:generalized-label
        |     |              +--:(sid)
        |     |                 +---w sid-hop
        |     |                    +---w sid?   rt-types:generalized-label
        |     +---w p2p-secondary-paths
        |     |  +---w p2p-secondary-path* [name]
        |     |     +---w name                      string
        |     |     +---w te-default-metric?        uint32
        |     |     +---w te-delay-metric?          uint32
        |     |     +---w te-hop-metric?            uint32
        |     |     +---w explicit-route-objects
        |     |        +---w explicit-route-object* [index]
        |     |           +---w explicit-route-usage?   identityref
        |     |           +---w index                   uint32
        |     |           +---w (type)?
        |     |              +--:(numbered)
        |     |              |  +---w numbered-hop
        |     |              |     +---w address?    te-types:te-tp-id
        |     |              |     +---w hop-type?   te-hop-type
        |     |              +--:(as-number)
        |     |              |  +---w as-number-hop
        |     |              |     +---w as-number?   binary
        |     |              |     +---w hop-type?    te-hop-type
        |     |              +--:(unnumbered)
        |     |              |  +---w unnumbered-hop
        |     |              |     +---w node-id?      te-types:te-node-id
        |     |              |     +---w link-tp-id?   te-types:te-tp-id
        |     |              |     +---w hop-type?     te-hop-type
        |     |              +--:(label)
        |     |              |  +---w label-hop
        |     |              |     +---w value?   rt-types:generalized-label
        |     |              +--:(sid)
        |     |                 +---w sid-hop
        |     |                    +---w sid?   rt-types:generalized-label
        |     +---w payload-treatment?          enumeration
        |     +---w src-client-signal?          identityref
        |     +---w src-tpn?                    uint16
```

```
       |     +---w src-tsg?                    identityref
       |     +---w src-tributary-slot-count?   uint16
       |     +---w src-tributary-slots
       |     |  +---w values*   uint8
       |     +---w dst-client-signal?          identityref
       |     +---w dst-tpn?                    uint16
       |     +---w dst-tsg?                    identityref
       |     +---w dst-tributary-slot-count?   uint16
       |     +---w dst-tributary-slots
       |        +---w values*   uint8
       +--ro output
          +--ro return-code?   enumeration
          +--ro result* [id]
             +--ro id                     uint8
             +--ro p2p-primary-paths
             |  +--ro p2p-primary-path* [name]
             |     +--ro name                      string
             |     +--ro te-default-metric?        uint32
             |     +--ro te-delay-metric?          uint32
             |     +--ro te-hop-metric?            uint32
             |     +--ro explicit-route-objects
             |        +--ro explicit-route-object* [index]
             |           +--ro explicit-route-usage?   identityref
             |           +--ro index                   uint32
             |           +--ro (type)?
             |              +--:(numbered)
             |              |  +--ro numbered-hop
             |              |     +--ro address?    te-types:te-tp-id
             |              |     +--ro hop-type?   te-hop-type
             |              +--:(as-number)
             |              |  +--ro as-number-hop
             |              |     +--ro as-number?   binary
             |              |     +--ro hop-type?    te-hop-type
             |              +--:(unnumbered)
             |              |  +--ro unnumbered-hop
             |              |     +--ro node-id?      te-types:te-node-id
             |              |     +--ro link-tp-id?   te-types:te-tp-id
             |              |     +--ro hop-type?     te-hop-type
             |              +--:(label)
             |              |  +--ro label-hop
             |              |     +--ro value?   rt-types:generalized-label
             |              +--:(sid)
             |                 +--ro sid-hop
             |                    +--ro sid?   rt-types:generalized-label
             +--ro p2p-secondary-paths
                +--ro p2p-secondary-path* [name]
                   +--ro name                      string
                   +--ro te-default-metric?        uint32
```

```
                  +--ro te-delay-metric?          uint32
                  +--ro te-hop-metric?            uint32
                  +--ro explicit-route-objects
                     +--ro explicit-route-object* [index]
                        +--ro explicit-route-usage?   identityref
                        +--ro index                   uint32
                        +--ro (type)?
                           +--:(numbered)
                           |  +--ro numbered-hop
                           |     +--ro address?    te-types:te-tp-id
                           |     +--ro hop-type?   te-hop-type
                           +--:(as-number)
                           |  +--ro as-number-hop
                           |     +--ro as-number?   binary
                           |     +--ro hop-type?    te-hop-type
                           +--:(unnumbered)
                           |  +--ro unnumbered-hop
                           |     +--ro node-id?      te-types:te-node-id
                           |     +--ro link-tp-id?   te-types:te-tp-id
                           |     +--ro hop-type?     te-hop-type
                           +--:(label)
                           |  +--ro label-hop
                           |     +--ro value?   rt-types:generalized-label
                           +--:(sid)
                              +--ro sid-hop
                                 +--ro sid?   rt-types:generalized-label
```

3.5. OTN Tunnel YANG Code

```
<CODE BEGINS>file "ietf-otn-tunnel@2017-10-30.yang"

module ietf-otn-tunnel {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-otn-tunnel";
  prefix "otn-tunnel";

  import ietf-te {
    prefix "te";
  }

  import ietf-otn-types {
    prefix "otn-types";
  }
```

```
     import ietf-te-types {
       prefix "te-types";
     }

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

     organization
       "IETF CCAMP Working Group";
     contact
       "WG Web: <http://tools.ietf.org/wg/ccamp/>
        WG List: <mailto:ccamp@ietf.org>

        Editor: Haomian Zheng
                <mailto:zhenghaomian@huawei.com>

        Editor: Zheyu Fan
                <mailto:fanzheyu2@huawei.com>

        Editor: Anurag Sharma
                <mailto:ansha@google.com>

        Editor: Rajan Rao
                <mailto:rrao@infinera.com>

        Editor: Sergio Belotti
                <mailto:sergio.belotti@nokia.com>

        Editor: Victor Lopez
                <mailto:victor.lopezalvarez@telefonica.com>

        Editor: Yunbo Li
                <mailto:liyunbo@chinamobile.com>

        Editor: Yunbin Xu
                <mailto:xuyunbin@ritt.cn>";

     description
       "This module defines a model for OTN Tunnel Services.";

     revision "2017-10-30" {
       description
         "Revision 0.4";
       reference
         "draft-ietf-ccamp-otn-tunnel-model-01.txt";
     }
```

```
     grouping otn-tunnel-endpoint {
       description "Parameters for OTN tunnel";

       leaf payload-treatment {
         type enumeration {
           enum switching {
             description "Client signal is switched to another tunnel
                          in this domain";
           }
           enum transport {
             description "Client signal is transparently transmitted
                          in this domain";
           }
         }
         default switching;
         description
           "Treatment of the incoming payload. Payload can be switched
            or transported.";
       }

       leaf src-client-signal {
         type identityref {
           base otn-types:client-signal;
         }
         description
           "Client signal at the source endpoint of the tunnel";
       }

       leaf src-tpn {
         type uint16 {
           range "0..4095";
         }
         description
           "Tributary Port Number. Applicable in case of mux services";
         reference
           "RFC7139: GMPLS Signaling Extensions for Control of Evolving
            G.709 Optical Transport Networks";
       }

       leaf src-tsg {
         type identityref {
           base otn-types:tributary-slot-granularity;
         }
         description
           "Tributary slot granularity.
            Applicable in case of mux services";
         reference
           "G.709/Y.1331, February 2016: Interfaces for the
```

```
             Optical Transport Network (OTN)";
       }

       leaf src-tributary-slot-count {
         type uint16;
         description
           "Number of tributary slots used at the source.";
       }

       container src-tributary-slots {
         description
           "A list of tributary slots used by the client service.
            Applicable in case of mux services";
         leaf-list values {
           type uint8;
             description
               "Tributary tributary slot value";
             reference
               "G.709/Y.1331, February 2016: Interfaces for the
                Optical Transport Network (OTN)";
         }
       }

       leaf dst-client-signal {
         type identityref {
           base otn-types:client-signal;
         }
         description
           "Client signal at the destination endpoint of the tunnel";
       }

       leaf dst-tpn {
         type uint16 {
           range "0..4095";
         }
         description
           "Tributary Port Number. Applicable in case of mux services";
         reference
           "RFC7139: GMPLS Signaling Extensions for Control of Evolving
            G.709 Optical Transport Networks.";
       }

       leaf dst-tsg {
         type identityref {
           base otn-types:tributary-slot-granularity;
         }
         description
           "Tributary slot granularity.
```

```
            Applicable in case of mux services";
        reference
          "G.709/Y.1331, February 2016: Interfaces for the
           Optical Transport Network (OTN)";
      }

      leaf dst-tributary-slot-count {
        type uint16;
        description
          "Number of tributary slots used at the destination.";
      }

      container dst-tributary-slots {
        description
          "A list of tributary slots used by the client service.
           Applicable in case of mux services";
        leaf-list values {
          type uint8;
          description
            "Tributary slot value";
          reference
            "G.709/Y.1331, February 2016: Interfaces for the
             Optical Transport Network (OTN)";
        }
      }
    }

    /*
    Note: Comment has been given to authors of TE Tunnel model to add
    list of endpoints under config to support P2MP tunnel.
    */
    augment "/te:te/te:tunnels/te:tunnel" {
      description
        "Augment with additional parameters required for OTN service";
      uses otn-tunnel-endpoint;
    }

    grouping p2p-path-ero {
      description
        "TE tunnel ERO configuration grouping";

      leaf te-default-metric {
        type uint32;
        description
          "Traffic engineering metric.";
      }
      leaf te-delay-metric {
        type uint32;
```

```
          description
            "Traffic engineering delay metric.";
        }
        leaf te-hop-metric {
          type uint32;
          description
            "Traffic engineering hop metric.";
        }
        container explicit-route-objects {
          description "Explicit route objects container";
          list explicit-route-object {
            key "index";
            description
              "List of explicit route objects";
            leaf explicit-route-usage {
              type identityref {
                base te-types:route-usage-type;
              }
              description "An explicit-route hop action.";
            }
            uses te-types:explicit-route-hop_config;
          }
        }
      }

      rpc otn-te-tunnel-path-compute {
        description "OTN TE tunnel path computation";
        input {
          list request {
            key "id";
            description "A list of path computation requests.";

            leaf id {
              type uint8;
              description
                "Request ID.";
            }
            leaf type {
              type identityref {
                base te-types:tunnel-type;
              }
              description "TE tunnel type.";
            }
            leaf source {
              type inet:ip-address;
              description
                "TE tunnel source address.";
            }
```

```
            leaf destination {
              type inet:ip-address;
              description
                "TE tunnel destination address";
            }
            leaf src-tp-id {
              type binary;
              description
                "TE tunnel source termination point identifier.";
            }
            leaf dst-tp-id {
              type binary;
              description
                "TE tunnel destination termination point identifier.";
            }
            leaf switching-layer {
              type identityref {
                base te-types:switching-capabilities;
              }
              description
                "Switching layer where the requests are computed.";
            }
            leaf encoding {
              type identityref {
                base te-types:lsp-encoding-types;
              }
              description "LSP encoding type";
            }
            leaf protection-type {
              type identityref {
                base te-types:lsp-protection-type;
              }
              description "LSP protection type";
            }
            leaf restoration-type {
              type identityref {
                base te-types:lsp-restoration-type;
              }
              description "LSP restoration type";
            }
            leaf provider-id {
              type te-types:te-global-id;
              description
                "An identifier to uniquely identify a provider.";
            }
            leaf client-id {
              type te-types:te-global-id;
              description
```

```
                "An identifier to uniquely identify a client.";
            }
            leaf te-topology-id {
              type te-types:te-topology-id;
              description
                "It is presumed that a datastore will contain many
                 topologies. To distinguish between topologies it is
                 vital to have UNIQUE topology identifiers.";
            }
            leaf setup-priority {
              type uint8 {
                range "0..7";
              }
              description
                "TE LSP setup priority";
            }
            leaf hold-priority {
              type uint8 {
                range "0..7";
              }
              description
                "TE LSP hold priority";
            }
            leaf te-path-metric-type {
              type identityref {
                base te-types:path-metric-type;
              }
              default te-types:path-metric-te;
              description
                "The tunnel path metric type.";
            }

            leaf odu-type {
              type identityref{
                base otn-types:tributary-protocol-type;
              }
              description "Type of ODU";
            }
            container p2p-primary-paths {
              description "Set of P2P primary paths container";
              list p2p-primary-path {
                key "name";
                description
                  "List of primary paths for this tunnel.";
                leaf name {
                  type string;
                  description "TE path name";
                }
```

```
              uses p2p-path-ero;
            }
          }
          container p2p-secondary-paths {
            description "Set of P2P secondary paths container";
            list p2p-secondary-path {
              key "name";
              description
                "List of secondary paths for this tunnel.";
              leaf name {
                type string;
                description "TE path name";
              }
              uses p2p-path-ero;
            }
          }
          uses otn-tunnel-endpoint;
        }
      }
      output {
        leaf return-code {
          type enumeration {
            enum success {
              description "success";
            }
            enum aborted {
              description "aborted";
            }
            enum destination-not-found {
              description "destination-not-found";
            }
            enum invalid-argument {
              description "invalid-argument";
            }
            enum no-memory {
              description "no-memory";
            }
            enum no-path-found {
              description "no-path-found";
            }
            enum other-error {
              description "other-error";
            }
            enum some-path-not-found {
              description "some-path-not-found";
            }
            enum source-not-found {
              description "source-not-found";
```

```
            }
            enum topology-error {
              description "topology-error";
            }
          }
          description
            "Return code";
        }
        list result {
          key "id";
          description
            "A list of results for all requests.";

          leaf id {
            type uint8;
            description
              "Request ID";
          }
          container p2p-primary-paths {
            description "Set of P2P primary paths container";
            list p2p-primary-path {
              key "name";
              description
                "List of resultant primary paths for this tunnel.";
              leaf name {
                type string;
                description "TE path name";
              }
              uses p2p-path-ero;
            }
          }
          container p2p-secondary-paths {
            description "Set of P2P secondary paths container";
            list p2p-secondary-path {
              key "name";
              description
                "List of resultant secondary paths for this tunnel.";
              leaf name {
                type string;
                description "TE path name";
              }
              uses p2p-path-ero;
            }
          }
        }
      }
    }
  }
```

```
<CODE ENDS>
```

3.6. OTN Types YANG Code

```
<CODE BEGINS> file "ietf-otn-types@2017-10-30.yang"

module ietf-otn-types {
  namespace "urn:ietf:params:xml:ns:yang:ietf-otn-types";
  prefix "otn-types";

  organization
    "IETF CCAMP Working Group";
  contact
    "WG Web: <http://tools.ietf.org/wg/ccamp/>
     WG List: <mailto:ccamp@ietf.org>

     Editor: Haomian Zheng
             <mailto:zhenghaomian@huawei.com>

     Editor: Zheyu Fan
             <mailto:fanzheyu2@huawei.com>

     Editor: Anurag Sharma
             <mailto:ansha@google.com>

     Editor: Rajan Rao
             <mailto:rrao@infinera.com>

     Editor: Sergio Belotti
             <mailto:sergio.belotti@nokia.com>

     Editor: Victor Lopez
             <mailto:victor.lopezalvarez@telefonica.com>

     Editor: Yunbo Li
             <mailto:liyunbo@chinamobile.com>

     Editor: Yunbin Xu
             <mailto:xuyunbin@ritt.cn>";

  description
    "This module defines OTN types.";

  revision "2017-10-30" {
    description
      "Revision 0.4";
```

```
      reference
        "draft-ietf-ccamp-otn-tunnel-model-01.txt";
    }

    identity tributary-slot-granularity {
      description
        "Tributary slot granularity";
      reference
        "G.709/Y.1331, February 2016: Interfaces for the Optical
         Transport Network (OTN)";
    }

    identity tsg-1.25G {
      base tributary-slot-granularity;
      description
        "1.25G tributary slot granularity";
    }

    identity tsg-2.5G {
      base tributary-slot-granularity;
      description
        "2.5G tributary slot granularity";
    }
  /*
    identity tsg-1.25Gand2.5G {
      base tributary-slot-granularity;
      description
        "Both 1.25G and 2.5G tributary slot granularity";
    }
  */
    identity tributary-protocol-type {
      description
        "Base identity for protocol framing used by tributary signals";
    }

    identity prot-OTU1 {
      base tributary-protocol-type;
      description
        "OTU1 protocol (2.66G)";
    }
  /*
    identity prot-OTU1e {
      base tributary-protocol-type;
      description
        "OTU1e type (11.04G)";
    }

    identity prot-OTU1f {
```

```
      base tributary-protocol-type;
      description
        "OTU1f type (11.27G)";
    }
  */
    identity prot-OTU2 {
      base tributary-protocol-type;
      description
        "OTU2 type (10.70G)";
    }

    identity prot-OTU2e {
      base tributary-protocol-type;
      description
        "OTU2e type (11.09G)";
    }
  /*
    identity prot-OTU2f {
      base tributary-protocol-type;
      description
        "OTU2f type (11.31G)";
    }
  */
    identity prot-OTU3 {
      base tributary-protocol-type;
      description
        "OTU3 type (43.01G)";
    }
  /*
    identity prot-OTU3e1 {
      base tributary-protocol-type;
      description
        "OTU3e1 type (44.57G)";
    }

    identity prot-OTU3e2 {
      base tributary-protocol-type;
      description
        "OTU3e2 type (44.58G)";
    }
  */
    identity prot-OTU4 {
      base tributary-protocol-type;
      description
        "OTU4 type (111.80G)";
    }

    identity prot-OTUCn {
```

```
      base tributary-protocol-type;
      description
        "OTUCn type (beyond 100G)";
    }

    identity prot-ODU0 {
      base tributary-protocol-type;
      description
        "ODU0 protocol (1.24G)";
    }

    identity prot-ODU1 {
      base tributary-protocol-type;
      description
        "ODU1 protocol (2.49G)";
    }
  /*
    identity prot-ODU1e {
      base tributary-protocol-type;
      description
        "ODU1e protocol (10.35G).";
    }

    identity prot-ODU1f {
      base tributary-protocol-type;
      description
        "ODU1f protocol (10.56G).";
    }
  */
    identity prot-ODU2 {
      base tributary-protocol-type;
      description
        "ODU2 protocol (10.03G)";
    }

    identity prot-ODU2e {
      base tributary-protocol-type;
      description
        "ODU2e protocol (10.39G)";
    }
  /*
    identity prot-ODU2f {
      base tributary-protocol-type;
      description
        "ODU2f protocol (10.60G).";
    }
  */
    identity prot-ODU3 {
```

```
      base tributary-protocol-type;
      description
        "ODU3 protocol (40.31G)";
    }
  /*
    identity prot-ODU3e1 {
      base tributary-protocol-type;
      description
        "ODU3e1 protocol (41.77G).";
    }

    identity prot-ODU3e2 {
      base tributary-protocol-type;
      description
        "ODU3e2 protocol (41.78G).";
    }
  */
    identity prot-ODU4 {
      base tributary-protocol-type;
      description
        "ODU4 protocol (104.79G)";
    }

    identity prot-ODUFlex-cbr {
      base tributary-protocol-type;
      description
        "ODU Flex CBR protocol for transporting constant bit rate
         signal";
    }

    identity prot-ODUFlex-gfp {
      base tributary-protocol-type;
      description
        "ODU Flex GFP protocol for transporting stream of packets
         using Generic Framing Procedure";
    }

    identity prot-ODUCn {
      base tributary-protocol-type;
      description
        "ODUCn protocol (beyond 100G)";
    }

    identity prot-1GbE {
      base tributary-protocol-type;
      description
        "1G Ethernet protocol";
    }
```

```
     identity prot-10GbE-LAN {
       base tributary-protocol-type;
       description
         "10G Ethernet LAN protocol";
     }

     identity prot-40GbE {
       base tributary-protocol-type;
       description
         "40G Ethernet protocol";
     }

     identity prot-100GbE {
       base tributary-protocol-type;
       description
         "100G Ethernet protocol";
     }

     identity client-signal {
       description
         "Base identity from which specific client signals for the
          tunnel are derived";
     }

     identity client-signal-1GbE {
       base client-signal;
       description
         "Client signal type of 1GbE";
     }

     identity client-signal-10GbE-LAN {
       base client-signal;
       description
         "Client signal type of 10GbE LAN";
     }

     identity client-signal-10GbE-WAN {
       base client-signal;
       description
         "Client signal type of 10GbE WAN";
     }

     identity client-signal-40GbE {
       base client-signal;
       description
         "Client signal type of 40GbE";
     }
```

```
identity client-signal-100GbE {
  base client-signal;
  description
    "Client signal type of 100GbE";
}

identity client-signal-OC3_STM1 {
  base client-signal;
  description
    "Client signal type of OC3 & STM1";
}

identity client-signal-OC12_STM4 {
  base client-signal;
  description
    "Client signal type of OC12 & STM4";
}

identity client-signal-OC48_STM16 {
  base client-signal;
  description
    "Client signal type of OC48 & STM16";
}

identity client-signal-OC192_STM64 {
  base client-signal;
  description
    "Client signal type of OC192 & STM64";
}

identity client-signal-OC768_STM256 {
  base client-signal;
  description
    "Client signal type of OC768 & STM256";
}

identity client-signal-ODU0 {
  base client-signal;
  description
    "Client signal type of ODU0 (1.24G)";
}

identity client-signal-ODU1 {
  base client-signal;
  description
    "ODU1 protocol (2.49G)";
}
```

```
     identity client-signal-ODU2 {
       base client-signal;
       description
         "Client signal type of ODU2 (10.03G)";
     }

     identity client-signal-ODU2e {
       base client-signal;
       description
         "Client signal type of ODU2e (10.39G)";
     }

     identity client-signal-ODU3 {
       base client-signal;
       description
         "Client signal type of ODU3 (40.31G)";
     }
   /*
     identity client-signal-ODU3e2 {
       base client-signal;
       description
         "Client signal type of ODU3e2 (41.78G)";
     }
   */
     identity client-signal-ODU4 {
       base client-signal;
       description
         "Client signal type of ODU4 (104.79G)";
     }

     identity client-signal-ODUflex-cbr {
       base client-signal;
       description
         "Client signal type of ODU Flex CBR";
     }

     identity client-signal-ODUflex-gfp {
       base client-signal;
       description
         "Client signal type of ODU Flex GFP";
     }

     identity client-signal-ODUCn {
       base client-signal;
       description
         "Client signal type of ODUCn (beyond 100G)";
     }
```

```
  identity client-signal-FC400 {
    base client-signal;
    description
      "Client signal type of Fibre Channel FC400";
  }

  identity client-signal-FC800 {
    base client-signal;
    description
      "Client signal type of Fibre Channel FC800";
  }

  identity client-signal-FICON-4G {
    base client-signal;
    description
      "Client signal type of Fibre Connection 4G";
  }

  identity client-signal-FICON-8G {
    base client-signal;
    description
      "Client signal type of Fibre Connection 8G";
  }
}

<CODE ENDS>
```

4. Security Considerations

TBD.

5. IANA Considerations

TBD.

6. Acknowledgements

TBD.

7. Contributors

Dieter Beller
Nokia
Email: dieter.beller@nokia.com

Yanlei Zheng
China Unicom

Email: zhengyl@dimpt.com

Xian Zhang
Huawei Technologies
Email: zhang.xian@huawei.com

Lei Wang
China Mobile
Email: wangleiyj@chinamobile.com

Oscar Gonzalez de Dios
Telefonica
Email: oscar.gonzalezdedios@telefonica.com

8. Normative References

[I-D.ietf-netmod-yang-tree-diagrams]
Bjorklund, M. and L. Berger, "YANG Tree Diagrams", draft-ietf-netmod-yang-tree-diagrams-02 (work in progress), October 2017.

Authors' Addresses

Haomian Zheng
Huawei Technologies
F3 R&D Center, Huawei Industrial Base, Bantian, Longgang District
Shenzhen, Guangdong 518129
P.R.China

Email: zhenghaomian@huawei.com

Zheyu Fan
Huawei Technologies
F3 R&D Center, Huawei Industrial Base, Bantian, Longgang District
Shenzhen, Guangdong 518129
P.R.China

Email: fanzheyu2@huawei.com

Anurag Sharma
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
USA

Email: ansha@google.com

Rajan Rao
Infinera
169 Java Drive
Sunnyvale, CA 94089
USA

Email: rrao@infinera.com

Sergio Belotti
Nokia

Email: sergio.belotti@nokia.com

Victor Lopez
Telefonica

Email: victor.lopezalvarez@telefonica.com

Yunbo Li
China Mobile

Email: liyunbo@chinamobile.com

Yunbin Xu
CAICT

Email: xuyunbin@ritt.cn