Common Control and Measurement Plane
Internet-Draft
Intended status: Standards Track
Expires: 2 January 2026

C. Yu
Huawei Technologies
S. Belotti
Nokia
I. Busi
Huawei Technologies
A. Guo
Futurewei Technologies
D. Beller
Nokia

1 July 2025

A YANG Data Model for Service Path Computation
draft-ybb-ccamp-service-path-computation-02

Abstract

This document defines a YANG data model for client signal service's path computation and path management.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://YuChaode.github.io/draft-ybb-ccamp-service-path-computation/draft-ybb-ccamp-service-path-computation.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ybb-ccamp-service-path-computation/.

Discussion of this document takes place on the Common Control and Measurement Plane Working Group mailing list (mailto:ccamp@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/ccamp/. Subscribe at https://www.ietf.org/mailman/listinfo/ccamp/.

Source for this draft and an issue tracker can be found at https://github.com/YuChaode/draft-ybb-ccamp-service-path-computation.

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/.

Copyright Notice

Table of Contents

1. Introduction 3
  1.1. Terminology and Notations 3
  1.2. Tree Diagram 4
  1.3. Prefix in Data Node Names 4
2. End to End Management of Transport Network Client Signal Service 4
3. Requirements for Service Path Computation 5
  3.1. Top-down Approach 5
  3.2. Multi-layer Path Display 8
  3.3. Path Constraint 10
  3.4. Path Management 10
    3.4.1. Path Reference in Service Provisioning 12
4. Tree Diagram for Service Path Computation 12
5. YANG Data Model for Service Path Computation 17
6. Conventions and Definitions 28
7. Security Considerations 28
8. IANA Considerations 28
9. Normative References 28
Acknowledgments 29
Authors' Addresses 29

1. Introduction

For transport technology, the service’s path is hierarchical, including OTN-layer trails and WDM-layer trails. According to the funcational structure defined by ITU-T, the OTN-layer trails include lower-order ODUk, high-order ODUk and OTUk trails. The WDM-lay trails include OTS, OMS and OCh trail. These trails and the configuration of client-side port configuration form an end-to-end client signal service.

Path computation is a common network management funcation. For traditional transport services, OTS and OMS trails are automatically generated once the fibers are connected and they don’t require any path computation. The path computation is performed on the OCh layer trails and above. Traditionally, the path computation is performed from OCh to low-order ODUk trail layer by layer. This is called bottom-up approach.

One disavantage of bottom-up approach is that the client and server need to interact with each other multiple times. And sometimes, the client layer trail cannot be computed until the server layer trail is setup. This doea not comply with the intention of path computation which will not introduce actual configuration on the network.

In addition, the client prefers to transfer intent-based configuration instead of complex network configuration to achieve path computation and obtain all the layers’ path computation result at one time. The server, usually it is played by the domain controller, can help to deal with the complex network configuration computation. This is called top-down approach.

This document focuses on the top-down path computation for transparent client signal service and non-transparent client signal service (Ethernet service).

We also focus on addressing some undiscussed requirements, such as how to ultize a generic structure to display path information for all the layers’ path at once, and how to specify some parameters on the server layer’s trail in the service path computation request, and how to support more complex path constraints, and how to manage the path computation results and how to reference them in service provisioning request.

1.1. Terminology and Notations

The following terms are defined in [RFC7950] and are not redefined here:

* client
* server
* augment
* data model
* data node

The terminology for describing YANG data models is found in [RFC7950].

1.2. Tree Diagram

A simplified graphical representation of the data model is used in Section 4 of this document. The meaning of the symbols in this diagram is defined in [RFC8340].

1.3. Prefix in Data Node Names

In this document, names of data nodes and other data model objects are prefixed using the standard prefix associated with the corresponding YANG imported modules, as shown in the following table.

Prefix	Yang Module	Reference
clnt-svc	ietf-trans-client-service	[RFCYYYY]
clnt-svc-pc	ietf-trans-client-path-computation	RFC XXXX

Table 1: Prefixes and corresponding YANG modules

RFC Editor Note: Please replace XXXX with the number assigned to the RFC once this draft becomes an RFC. Please replace YYYY with the RFC numbers assigned to [I-D.ietf-ccamp-client-signal-yang]. Please remove this note.

2. End to End Management of Transport Network Client Signal Service

The hierarchical relationship of transport client signal service can reference to Figure 1: (Please note that, the OTN tunnels in this figure and subsequent figures include lower order OTN tunnels, higher order OTN tunnels, and OTU tunnels. The supporting relationship should comply with ITU-T G.XXXX definition. WDM tunnel also include the WDM, flexi-grid scenario and Media channel scenario. The

hierarchical relationship should comply with ITU-T G.XXXX definition.)

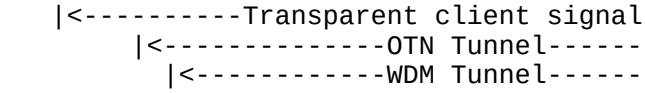

Figure 1: Transparent Client Signal Service and its Server tunnel

For Ethernet client signal service, the client signal can be encapsulated into multiple approach. For example, it can be encapsulated into OTN tunnel directly, or it can be encapsulated into MPLS-TP/SDH tunnels and then into OTN tunnels. Note that the SDH tunnel is considered as an outdated technology and lacks standardization. Therefore, the scenario where Ethernet client signals are encapsulated into the SDH tunnel is not described in this document. The Figure 2 shows the hierarchical relationship of Ethernet service:

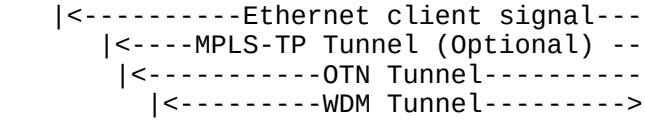

Figure 2: Ethernet Client Signal Service and its Server tunnel

The reference method is defined in the [I-D.ietf-ccamp-client-signal-yang]. The supporting relationship between the tunnels can be also found by the dependency-tunnel structure define by the [I-D.ietf-teas-yang-te].

3. Requirements for Service Path Computation

3.1. Top-down Approach

For the Top-down approach, the path computation request should be based on client signal service. It is needed to specify the source and destination access port of in the request. In addition, some common parameters, such as number of path to be calculated, protection and restoration capabities, path computation policy and path constraint (see Section 3.3).etc. And then the domain controller will trigger the computation on all the layers. The path computation result should contain all the layers' path information. For the OTN tunnel and WDM tunnel in the service path computation result, they can be non-existing before service path compuation and can be totally designed by the domain controller or control plane. The tunnels can also be existing before the service computation request. The domain controller can design to reuse some existing

tunnels based on the consideration of maximum resource utilization. Similar to TE tunnel path computation, service path computation should not create any tunnels on the network during the whole computation process, and will not introduce any other changes on the network.

```
   module: ietf-trans-client-service-path-computation
   rpcs:
      +---x client-service-precompute
         +--ro input
         |  +--ro request* [request-id]
         |     +--ro request-id                        string
         |     +--ro path-count?                       uint8
         |     +--ro te-topology-identifier
         |     +--ro src-access-ports
         |     +--ro dst-access-ports
         |     +--ro tunnel-policy
         |     |  +--ro protection
         |     |  +--ro restoration
         |     |  +--ro optimizations
         |     +--ro explicit-route-exclude-objects
         |     |  +--ro route-object-exclude-object* [index]
         |     +--ro explicit-route-include-objects
         |        +--ro route-object-include-object* [index]
         +--ro output
            +--ro result* [request-id]
               +--ro request-id               string
               +--ro result-code?             enumeration
               +--ro (result-detail)?
                  +--:(failure)
                  |  +--ro failure-reason?          uint32
                  |  +--ro error-message?           string
                  +--:(success)
                     +--ro computed-paths* [path-id]
                     |  +--ro path-id        yang:uuid
                     |  +--ro path-number?   uint8
                     +--ro te-topology-identifier
                     +--ro src-access-ports
                     +--ro dst-access-ports
                     +--ro underlay-tunnels
                        +--ro underlay-tunnel* [index]
                           +--ro index                     uint8
                           +--ro tunnel-name?              leafref
                           +--ro te-topology-identifier
                           +--ro computed-lsp* [lsp-id]
                              +--ro lsp-id              uint8
                              +--ro lsp-type?           enumeration
                              +--ro lsp-metrics
                              |  +--ro lsp-metric* [metric-type]
                              +--ro lsp-route-objects* [index]
```

3.2. Multi-layer Path Display

For the MDSC, if it wants to show the whole information of computed path, the path computation result provided by the domain controller must contain tunnels at all the layers. The number of these tunnels is not fixed. For example, for OTN tunnel, the client signal can be encapsulated into a low order OTN tunnel at first, and then be multiplexed into a higher OTN tunnel. The client signal can be encapsulated into a high order OTN tunnel directly without the multiplexing scenario. There is another common scenario that an OTN tunnel is supported by multiple WDM tunnels. This document provides a generic structure in a list to present the actual path information regardless which layer it is. The tunnels will be ordered by index from 0 for the topmost tunnel. The correlation with server tunnel will be provided by the "server-tunnel" attribute in the LSP hop.

```
   ..........................
   +--ro underlay-tunnels
      +--ro underlay-tunnel* [index]
         +--ro index                    uint8
         +--ro tunnel-name?             leafref
         +--ro te-topology-identifier
         |  +--ro provider-id?   te-types:te-global-id
         |  +--ro client-id?     te-types:te-global-id
         |  +--ro topology-id?   te-types:te-topology-id
         +--ro computed-lsp* [lsp-id]
            +--ro lsp-id               uint8
            +--ro lsp-type?            enumeration
            +--ro lsp-metrics
            |  +--ro lsp-metric* [metric-type]
            |     +--ro metric-type     identityref
            |     +--ro metric-value?   uint32
            |     +--ro metric-unit?    string
            +--ro lsp-route-objects* [index]
               +--ro index            uint8
               +--ro node-id?         te-types:te-node-id
               +--ro node-uri-id?     yang:uuid
               +--ro link-tp-id?      te-types:te-tp-id
               +--ro ltp-uri-id?      yang:uuid
               +--ro te-label
               |  +--ro (technology)?
               |  |  +--:(wson)
               |  |  |  +--ro (grid-type)?
               |  |  |     +--:(dwdm)
               |  |  |     |  +--ro (single-or-super-channel)?
               |  |  |     |     +--:(single)
               |  |  |     |     |  +--ro dwdm-n?
               l0-types:dwdm-n
               |  |  |     |     +--:(super)
               |  |  |     |        +--ro subcarrier-dwdm-n*
               l0-types:dwdm-n
               |  |  |     +--:(cwdm)
               |  |  |        +--ro cwdm-n?              l0-types:cwdm-n
               |  |  +--:(otn)
               |  |     +--ro otn
               |  |        +--ro tpn?       otn-tpn
               |  |        +--ro tsg?       identityref
               |  |        +--ro ts-list?   string
               |  +--ro direction?           te-types:te-label-direction
               +--ro server-tunnel?   leafref
```

3.3. Path Constraint

It is common for service path computation request to specify path constrain like node/link-tp inclusion/exclusion like TE tunnel path computation. And service path computation needs to support some more kind of path constraint, such as to specify service/tunnel/path included/excluded. There are also scenarios to specify path constrain across layers. For example, some people would like to specify a WDM node included/excluded or wavelength in the service path computation.

```
.............................
+--ro route-object-include-object* [index]
   +--ro index                 uint8
   +--ro node-id?              te-types:te-node-id
   +--ro node-uri-id?          yang:uuid
   +--ro link-tp-id?           te-types:te-tp-id
   +--ro ltp-uri-id?           yang:uuid
   +--ro te-label
   |  +--ro (technology)?
   |  |  +--:(wson)
   |  |  |  +--ro (grid-type)?
   |  |  |     +--:(dwdm)
   |  |  |     |  +--ro (single-or-super-channel)?
   |  |  |     |     +--:(single)
   |  |  |     |     |  +--ro dwdm-n?              l0-types:dwdm-n
   |  |  |     |     +--:(super)
   |  |  |     |        +--ro subcarrier-dwdm-n*   l0-types:dwdm-n
   |  |  |     +--:(cwdm)
   |  |  |        +--ro cwdm-n?              l0-types:cwdm-n
   |  |  +--:(otn)
   |  |     +--ro otn
   |  |        +--ro tpn?       otn-tpn
   |  |        +--ro tsg?       identityref
   |  |        +--ro ts-list?   string
   |  +--ro direction?           te-types:te-label-direction
   +--ro server-tunnel-name?   leafref
   +--ro lsp-type?             enumeration
```

3.4. Path Management

Storage of Path Computation Result It is useful to save the path computation results after they are return. Sometimes, people from operators will make the decision on the results in a short time. If the path is not saved in the domain controller, the MDSC needs to send the full path in the service creation request which is complex. So in this document, we recommend that the domain controller should be capable of saving path computation results to make it possible

that the MDSC can reference the path computation result in service creation request. The path information in the path management structure should be same with the output of service path computation RPC. Once there is a RPC succeed in operating, the path management structure will add one more item. It is noted that the path computation result is not required to be saved forever in the domain controller. How long could it be saved is implementation-specific. But if the path has been adopted by a service creation request, including path inclusion/exclusion, the path can not be deleted from data store.

Note: the service path computation request is defined as an RPC, which is stateless. According to the requirement of RESTCONF, RPC should not generate any impact on the data model. So it is recommended to discuss with RESTCONF protocol expert to find a workaround solution.

```
module: ietf-trans-client-service-path-computation
   +--rw path-management
      +--rw path* [path-id]
         +--rw path-id             yang:uuid
         +--rw creation-time?      yang:date-and-time
         +--rw validity-period?    uint8
         +--rw underlay-tunnels
            +--rw underlay-tunnel* [index]
               +--rw index                      uint8
               +--rw tunnel-name?               leafref
               +--rw te-topology-identifier
               |  +--rw provider-id?   te-types:te-global-id
               |  +--rw client-id?     te-types:te-global-id
               |  +--rw topology-id?   te-types:te-topology-id
               +--rw computed-lsp* [lsp-id]
                  +--rw lsp-id               uint8
                  +--rw lsp-type?            enumeration
                  +--rw lsp-metrics
                  |  +--rw lsp-metric* [metric-type]
                  |     +--rw metric-type     identityref
                  |     +--rw metric-value?   uint32
                  |     +--rw metric-unit?    string
                  +--rw lsp-route-objects* [index]
                     +--rw index            uint8
                     +--rw node-id?         te-types:te-node-id
                     +--rw node-uri-id?     yang:uuid
                     +--rw link-tp-id?      te-types:te-tp-id
                     +--rw ltp-uri-id?      yang:uuid
                     +--rw te-label
                     |  +--rw (technology)?
                     |  |  +--:(wson)
```

```
                        |  |  |  +--rw (grid-type)?
                        |  |  |     +--:(dwdm)
                        |  |  |     |  +--rw (single-or-super-channel)?
                        |  |  |     |     +--:(single)
                        |  |  |     |     |  +--rw dwdm-n?
                        l0-types:dwdm-n
                        |  |  |     |     +--:(super)
                        |  |  |     |        +--rw subcarrier-dwdm-n*
                        l0-types:dwdm-n
                        |  |  |     +--:(cwdm)
                        |  |  |        +--rw cwdm-n?
                        l0-types:cwdm-n
                        |  |  +--:(otn)
                        |  |     +--rw otn
                        |  |        +--rw tpn?       otn-tpn
                        |  |        +--rw tsg?       identityref
                        |  |        +--rw ts-list?   string
                        |  +--rw direction?
                        te-types:te-label-direction
                        +--rw server-tunnel?   leafref
```

3.4.1. Path Reference in Service Provisioning

In the current service provisioning approach, the MDSC needs to specify the correlation of tunnel underlay. If the path computation result is saved in the domain controller. It is much easier to reference the path computation result instead of specifying tunnel underlay to do the provisioning.

To be added

4. Tree Diagram for Service Path Computation

```
module: ietf-trans-client-service-path-computation
  +--rw path-management
     +--rw path* [path-id]
        +--rw path-id             yang:uuid
        +--rw creation-time?      yang:date-and-time
        +--rw validity-period?    uint8
        +--rw underlay-tunnels
           +--rw underlay-tunnel* [index]
              +--rw index                    uint8
              +--rw tunnel-name?
              |       -> /te:te/tunnels/tunnel/name
              +--rw te-topology-identifier
              |  +--rw provider-id?   te-global-id
              |  +--rw client-id?     te-global-id
              |  +--rw topology-id?   te-topology-id
```

```
            +--rw computed-lsp* [lsp-id]
               +--rw lsp-id               uint8
               +--rw lsp-type?            enumeration
               +--rw lsp-metrics
               |  +--rw lsp-metric* [metric-type]
               |     +--rw metric-type     identityref
               |     +--rw metric-value?   uint32
               |     +--rw metric-unit?    string
               +--rw lsp-route-objects* [index]
                  +--rw index            uint8
                  +--rw node-id?         te-types:te-node-id
                  +--rw node-uri-id?     yang:uuid
                  +--rw link-tp-id?      te-types:te-tp-id
                  +--rw ltp-uri-id?      yang:uuid
                  +--rw te-label
                  |  +--rw (technology)?
                  |  |  +--:(wson)
                  |  |  |  +--rw (grid-type)?
                  |  |  |     +--:(dwdm)
                  |  |  |     |  +--rw (single-or-super-channel)?
                  |  |  |     |     +--:(single)
                  |  |  |     |     |  +--rw dwdm-n?
                  |  |  |     |     |           l0-types:dwdm-n
                  |  |  |     |     +--:(super)
                  |  |  |     |        +--rw subcarrier-dwdm-n*
                  |  |  |     |                 l0-types:dwdm-n
                  |  |  |     +--:(cwdm)
                  |  |  |        +--rw cwdm-n?
                  |  |  |                 l0-types:cwdm-n
                  |  |  +--:(otn)
                  |  |     +--rw otn-label
                  |  |        +--rw tpn?       otn-tpn
                  |  |        +--rw tsg?       identityref
                  |  |        +--rw ts-list?   string
                  |  +--rw direction?
                  |           te-types:te-label-direction
                  +--rw server-tunnel?   -> ../../../index

  rpcs:
    +---x client-service-precompute
       +---w input
       |  +---w request* [request-id]
       |     +---w request-id                        string
       |     +---w path-count?                       uint8
       |     +---w te-topology-identifier
       |     |  +---w provider-id?   te-global-id
       |     |  +---w client-id?     te-global-id
       |     |  +---w topology-id?   te-topology-id
```

```
|     +---w src-access-ports
|     |  +---w access-node-id?    te-types:te-node-id
|     |  +---w access-node-uri?   nw:node-id
|     |  +---w access-ltp-id?     te-types:te-tp-id
|     |  +---w access-ltp-uri?    nt:tp-id
|     |  +---w client-signal?     identityref
|     +---w dst-access-ports
|     |  +---w access-node-id?    te-types:te-node-id
|     |  +---w access-node-uri?   nw:node-id
|     |  +---w access-ltp-id?     te-types:te-tp-id
|     |  +---w access-ltp-uri?    nt:tp-id
|     |  +---w client-signal?     identityref
|     +---w tunnel-policy
|     |  +---w protection
|     |  |  +---w protection-type?                identityref
|     |  |  +---w protection-reversion-disable?   boolean
|     |  |  +---w hold-off-time?                  uint32
|     |  |  +---w wait-to-revert?                 uint16
|     |  |  +---w aps-signal-id?                  uint8
|     |  +---w restoration
|     |  |  +---w restoration-type?                identityref
|     |  |  +---w restoration-scheme?              identityref
|     |  |  +---w restoration-reversion-disable?   boolean
|     |  |  +---w hold-off-time?                   uint32
|     |  |  +---w wait-to-restore?                 uint16
|     |  |  +---w wait-to-revert?                  uint16
|     |  +---w share-timeslot?   boolean
|     |  +---w optimizations
|     |     +---w (algorithm)?
|     |     |  +--:(metric)
|     |     |     +---w optimization-metric* [metric-type]
|     |     |        +---w metric-type    identityref
|     |     |        +---w weight?        uint8
|     |     +---w lsp-type?                    enumeration
|     |     +---w path-metric-bounds
|     |        +---w path-metric-bound* [metric-type]
|     |           +---w metric-type    identityref
|     |           +---w upper-bound?   uint64
|     +---w explicit-route-exclude-objects
|     |  +---w route-object-exclude-object* [index]
|     |     +---w index                uint8
|     |     +---w node-id?             te-types:te-node-id
|     |     +---w node-uri-id?         yang:uuid
|     |     +---w link-tp-id?          te-types:te-tp-id
|     |     +---w ltp-uri-id?          yang:uuid
|     |     +---w te-label
|     |     |  +---w (technology)?
|     |     |  |  +--:(wson)
```

```
        |     |     |  |  |  +---w (grid-type)?
        |     |     |  |  |     +--:(dwdm)
        |     |     |  |  |     |  +---w (single-or-super-channel)?
        |     |     |  |  |     |     +--:(single)
        |     |     |  |  |     |     |  +---w dwdm-n?
        |     |     |  |  |     |     |          l0-types:dwdm-n
        |     |     |  |  |     |     +--:(super)
        |     |     |  |  |     |        +---w subcarrier-dwdm-n*
        |     |     |  |  |     |                l0-types:dwdm-n
        |     |     |  |  |     +--:(cwdm)
        |     |     |  |  |        +---w cwdm-n?
        |     |     |  |  |                l0-types:cwdm-n
        |     |     |  |  +--:(otn)
        |     |     |  |     +---w otn-label
        |     |     |  |        +---w tpn?       otn-tpn
        |     |     |  |        +---w tsg?       identityref
        |     |     |  |        +---w ts-list?   string
        |     |     |  +---w direction?
        |     |     |          te-types:te-label-direction
        |     |     +---w server-tunnel-name?
        |     |     |       -> /te:te/tunnels/tunnel/name
        |     |     +---w lsp-type?             enumeration
        |     +---w explicit-route-include-objects
        |        +---w route-object-include-object* [index]
        |           +---w index                 uint8
        |           +---w node-id?              te-types:te-node-id
        |           +---w node-uri-id?          yang:uuid
        |           +---w link-tp-id?           te-types:te-tp-id
        |           +---w ltp-uri-id?           yang:uuid
        |           +---w te-label
        |           |  +---w (technology)?
        |           |  |  +--:(wson)
        |           |  |  |  +---w (grid-type)?
        |           |  |  |     +--:(dwdm)
        |           |  |  |     |  +---w (single-or-super-channel)?
        |           |  |  |     |     +--:(single)
        |           |  |  |     |     |  +---w dwdm-n?
        |           |  |  |     |     |          l0-types:dwdm-n
        |           |  |  |     |     +--:(super)
        |           |  |  |     |        +---w subcarrier-dwdm-n*
        |           |  |  |     |                l0-types:dwdm-n
        |           |  |  |     +--:(cwdm)
        |           |  |  |        +---w cwdm-n?
        |           |  |  |                l0-types:cwdm-n
        |           |  |  +--:(otn)
        |           |  |     +---w otn-label
        |           |  |        +---w tpn?       otn-tpn
        |           |  |        +---w tsg?       identityref
```

```
       |           |  |        +---w ts-list?   string
       |           |  +---w direction?
       |           |           te-types:te-label-direction
       |           +---w server-tunnel-name?
       |           |       -> /te:te/tunnels/tunnel/name
       |           +---w lsp-type?             enumeration
       +--ro output
          +--ro result* [request-id]
             +--ro request-id                      string
             +--ro result-code?                    enumeration
             +--ro (result-detail)?
                +--:(failure)
                |  +--ro failure-reason?           uint32
                |  +--ro error-message?            string
                +--:(success)
                   +--ro computed-paths* [path-id]
                   |  +--ro path-id        yang:uuid
                   |  +--ro path-number?   uint8
                   +--ro te-topology-identifier
                   |  +--ro provider-id?   te-global-id
                   |  +--ro client-id?     te-global-id
                   |  +--ro topology-id?   te-topology-id
                   +--ro src-access-ports
                   |  +--ro access-node-id?    te-types:te-node-id
                   |  +--ro access-node-uri?   nw:node-id
                   |  +--ro access-ltp-id?     te-types:te-tp-id
                   |  +--ro access-ltp-uri?    nt:tp-id
                   |  +--ro client-signal?     identityref
                   +--ro dst-access-ports
                   |  +--ro access-node-id?    te-types:te-node-id
                   |  +--ro access-node-uri?   nw:node-id
                   |  +--ro access-ltp-id?     te-types:te-tp-id
                   |  +--ro access-ltp-uri?    nt:tp-id
                   |  +--ro client-signal?     identityref
                   +--ro underlay-tunnels
                      +--ro underlay-tunnel* [index]
                         +--ro index                      uint8
                         +--ro tunnel-name?
                         |       -> /te:te/tunnels/tunnel/name
                         +--ro te-topology-identifier
                         |  +--ro provider-id?   te-global-id
                         |  +--ro client-id?     te-global-id
                         |  +--ro topology-id?   te-topology-id
                         +--ro computed-lsp* [lsp-id]
                            +--ro lsp-id               uint8
                            +--ro lsp-type?            enumeration
                            +--ro lsp-metrics
                            |  +--ro lsp-metric* [metric-type]
```

```
|     +--ro metric-type     identityref
|     +--ro metric-value?   uint32
|     +--ro metric-unit?    string
+--ro lsp-route-objects* [index]
   +--ro index            uint8
   +--ro node-id?
   |       te-types:te-node-id
   +--ro node-uri-id?     yang:uuid
   +--ro link-tp-id?
   |       te-types:te-tp-id
   +--ro ltp-uri-id?      yang:uuid
   +--ro te-label
   |  +--ro (technology)?
   |  |  +--:(wson)
   |  |  |  +--ro (grid-type)?
   |  |  |     +--:(dwdm)
   |  |  |     |  +--ro (single-or-super-channel)?
   |  |  |     |     +--:(single)
   |  |  |     |     |  +--ro dwdm-n?
   |  |  |     |     |          l0-types:dwdm-n
   |  |  |     |     +--:(super)
   |  |  |     |        +--ro subcarrier-dwdm-n*
   |  |  |     |                l0-types:dwdm-n
   |  |  |     +--:(cwdm)
   |  |  |        +--ro cwdm-n?
   |  |  |                l0-types:cwdm-n
   |  |  +--:(otn)
   |  |     +--ro otn-label
   |  |        +--ro tpn?       otn-tpn
   |  |        +--ro tsg?
   |  |        |       identityref
   |  |        +--ro ts-list?   string
   |  +--ro direction?
   |          te-types:te-label-direction
   +--ro server-tunnel?
           -> ../../../index
```

Figure 3: Service path computation tree diagram

5. YANG Data Model for Service Path Computation

```
<CODE BEGINS>
  file "ietf-trans-client-service-path-computation@2024-07-07.yang"
 module ietf-trans-client-service-path-computation {
   /* TODO: FIXME */
     yang-version 1.1;

   namespace "urn:ietf:params:xml:ns:yang:ietf-trans-client-service-path-computation";
```

```
      prefix "clntsvc-cpt";

      import ietf-trans-client-service {
        prefix "clntsvc";
        reference "draft-ietf-ccamp-client-signal-yang";
      }

      import ietf-te-types {
        prefix "te-types";
        reference "RFC 8776 - Traffic Engineering Common YANG Types";
      }

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

     import ietf-layer1-types {
       prefix l1-types;
       reference "RFC ZZZZ - A YANG Data Model for Layer 1 Types";
     }

     import ietf-layer0-types {
       prefix l0-types;
     }

     import ietf-te {
       prefix "te";
     }

      organization
        "Internet Engineering Task Force (IETF) CCAMP WG";
      contact
        "
          ID-draft editor:
            Chaode Yu (yuchaode@huawei.com);
            Sergio Belotti (sergio.belotti@nokia.com);
            Italo Busi (italo.busi@huawei.com);
            Aihua Guo (aihuaguo.ietf@gmail.com);
            Dieter Beller (dieter.beller@nokia.com);
        ";

      description
        "This module defines a YANG data model for describing
         transport network client services. The model fully conforms
         to the Network Management Datastore Architecture (NMDA).

         Copyright (c) 2021 IETF Trust and the persons
```

```
         identified as authors of the code.  All rights reserved.

         Redistribution and use in source and binary forms, with or
         without modification, is permitted pursuant to, and subject
         to the license terms contained in, the Simplified BSD License
         set forth in Section 4.c of the IETF Trust's Legal Provisions
         Relating to IETF Documents
         (https://trustee.ietf.org/license-info).
         This version of this YANG module is part of RFC XXXX; see
         the RFC itself for full legal notices.";

      revision 2024-07-07 {
         description
           "initial version";
         reference
           "draft-ybb-ccamp-service-path-computation";
       }


      container path-management {
        list path {
          key path-id;

          leaf path-id {
            type yang:uuid;
          }

          leaf creation-time {
            type yang:date-and-time;
          }

          leaf validity-period {
            type uint8;
            units "minute";
          }

          container underlay-tunnels {
            list underlay-tunnel {
              key index;

              leaf index {
                description
                  "The tunnels underlay should be ordered by  their
                  supproting relationship. The client layer tunnels
                  should use smaller index.";
                  type uint8;
              }
```

```
            leaf tunnel-name {
              type leafref {
                path "/te:te/te:tunnels/te:tunnel/te:name";
              }

              description
                "the computed result can route with some existing
                TE tunnels. The tunnel-name is the identifier of
                tunnel. If the tunnel is not created, this
                parameter is not needed to provide.";
            }

            uses te-types:te-topology-identifier;

            list computed-lsp {
              key lsp-id;
              leaf lsp-id {
                type uint8;
              }

              leaf lsp-type {
                type enumeration {
                  enum forwarding-working;
                  enum forwarding-protection;
                  enum reverse-working;
                  enum reverse-protection;
                }
              }

              container lsp-metrics {
                list lsp-metric {
                  key metric-type;

                  leaf metric-type {
                    type identityref {
                      base te-types:path-metric-type;
                    }
                  }

                  leaf metric-value {
                    type uint32;
                  }

                  leaf metric-unit {
                    type string;
                  }
                }
              }
```

```
              list lsp-route-objects {
                key index;

                leaf index {
                  type uint8;
                }

                uses hop-infomation;

                leaf server-tunnel {
                  type leafref {
                    path "../../../index";
                  }
                }
              }
            }
          }
        }
      }
    }


    rpc client-service-precompute {
      input {
        list request {
          key "request-id";

          leaf request-id {
            type string;
          }

          leaf path-count {
            type uint8 {
              range "1..10";
            }
          }

          uses te-types:te-topology-identifier;

          container src-access-ports {
            description
              "Source access port of a client service.";
            uses clntsvc:client-svc-access-parameters;
          }

          container dst-access-ports {
            description
                  "Destination access port of a client service.";
```

```
            uses clntsvc:client-svc-access-parameters;
          }

          uses tunnel-policy-grouping;
          uses path-constraints;
        }
      }
      output {
        list result {
          key "request-id";

          leaf request-id {
            type string;
          }

          leaf result-code {
            type enumeration {
              enum success {
              }
              enum failure {
              }
            }
          }
          choice result-detail {
            case failure {
              uses failure-info-grouping;
            }



            case success {
              list computed-paths {
                key path-id;

                leaf path-id {
                  type yang:uuid;
                }

                leaf path-number {
                  type uint8;
                  description
                    "when the client requestes to compute multiple paths
                     for a service, this path-number can be used to rank
                     the path result, based on the path computation
                     policy. The path-number starts with 0 and 0
                     indicates the best option. The better path's
                     path-number should be in lower number.";
                }
```

```
            }

            uses te-types:te-topology-identifier;
            container src-access-ports {
              description
                "Source access port of a client service.";
              uses clntsvc:client-svc-access-parameters;
            }

            container dst-access-ports {
              description
                    "Destination access port of a client service.";
              uses clntsvc:client-svc-access-parameters;
            }

            container underlay-tunnels {
              list underlay-tunnel {
                key index;

                description
                  "The server could support all the layers of tunnels
                  under the client signal service. If it cannot
                  support that, it should return its topmost layer
                  tunnel's path infomation";

                leaf index {
                  description
                    "The tunnels underlay should be ordered by  their
                    supproting relationship. The client layer tunnels
                    should use smaller index.";
                  type uint8;
                }

                leaf tunnel-name {
                  type leafref {
                    path "/te:te/te:tunnels/te:tunnel/te:name";
                  }

                  description
                    "the computed result can route with some existing
                    TE tunnels. The tunnel-name is the identifier of
                    tunnel. If the tunnel is not created, this
                    parameter is not needed to provide.";
                }

                uses te-types:te-topology-identifier;

                list computed-lsp {
```

```
                    key lsp-id;

                    leaf lsp-id {
                      type uint8;
                    }

                    leaf lsp-type {
                      type enumeration {
                        enum forwarding-working;
                        enum forwarding-protection;
                        enum reverse-working;
                        enum reverse-protection;
                      }
                    }

                    container lsp-metrics {
                      list lsp-metric {
                        key metric-type;

                        leaf metric-type {
                          type identityref {
                            base te-types:path-metric-type;
                          }
                        }

                        leaf metric-value {
                          type uint32;
                        }

                        leaf metric-unit {
                          type string;
                        }
                      }
                    }

                    list lsp-route-objects {
                      key index;

                      leaf index {
                        type uint8;
                      }

                      uses hop-infomation;

                      leaf server-tunnel {
                        type leafref {
                          path "../../../index";
                        }
```

```
                        }
                      }
                    }

                  }
                }
              }
            }
          }
        }

      grouping hop-infomation {
        leaf node-id {
            type te-types:te-node-id;
            description
              "The identifier of a node in the TE topology.";
          }

          leaf node-uri-id {
            type yang:uuid;
            description
              "Another identifier of TE node. This uuid of node may be acquired by invento
ry.";
          }

          leaf link-tp-id {
            type te-types:te-tp-id;
              description
                "TE link termination point identifier, used
                 together with te-node-id to identify the
                 link termination point";
          }

          leaf ltp-uri-id {
            type yang:uuid;
            description
              "another identifier of link TP. This uuid of TP may be acquired by inventory
.";
          }

          container te-label {
            description
              "Container that specifies TE label.";
            choice technology {
              description
                "Data plane technology type.";
              case wson {
                uses l0-types:wson-label-hop;
              }
```

```
            case otn {
              uses l1-types:otn-label-hop;
            }
          }
          leaf direction {
            type te-types:te-label-direction;
            description "Label direction";
          }
        }

    }

    grouping tunnel-policy-grouping {
      description "Tunnel creation policy which can be also inherited by client service.
";
      container tunnel-policy {
        uses te:protection-restoration-properties;

        leaf share-timeslot {
          type boolean;
        }

        uses path-policy;
      }
    }

    grouping failure-info-grouping {
      leaf failure-reason {
        type uint32;
      }
      leaf error-message {
        type string;
      }
    }

    grouping path-constraints {
      description "";

      container explicit-route-exclude-objects {
        list route-object-exclude-object {
          key index;
          uses hop-include-or-exclude-grouping;
        }
      }

      container explicit-route-include-objects {
        list route-object-include-object {
          key index;
          uses hop-include-or-exclude-grouping;
```

```
        }
      }
    }

    grouping hop-include-or-exclude-grouping {
      leaf index {
        type uint8;
      }

      uses hop-infomation;
      leaf server-tunnel-name {
        type leafref {
          path "/te:te/te:tunnels/te:tunnel/te:name";
        }
      }

      leaf lsp-type {
        type enumeration {
          enum all-paths;
          enum forwarding-working;
          enum forwarding-protection;
          enum reverse-working;
          enum reverse-protection;
        }
      }
    }

    grouping path-policy {
      container optimizations {
        choice algorithm {
          case metric {
            list optimization-metric {
              key "metric-type";
              leaf metric-type {
                type identityref {
                  base te-types:path-metric-type;
                }
                description "TE path metric type";
              }
              leaf weight {
                type uint8;
                description "TE path metric normalization weight";
              }
            }
          }
        }

        leaf lsp-type {
```

```
            type enumeration {
              enum all-paths;
              enum forwarding-working;
              enum forwarding-protection;
              enum reverse-working;
              enum reverse-protection;
            }
          }

          uses te-types:generic-path-metric-bounds;
        }
      }
     }
    <CODE ENDS>
```

Figure 4: Service Path Computation YANG module

6. Conventions and Definitions

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

7. Security Considerations

TODO Security

8. IANA Considerations

This document has no IANA actions.

9. Normative References

[I-D.ietf-ccamp-client-signal-yang]
Zheng, H., Guo, A., Busi, I., Snitser, A., and C. Yu, "A YANG Data Model for Transport Network Client Signals", Work in Progress, Internet-Draft, draft-ietf-ccamp-client-signal-yang-14, 4 February 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-ccamp-client-signal-yang-14>.

[I-D.ietf-teas-yang-te]
Saad, T., Gandhi, R., Liu, X., Beeram, V. P., and I. Bryskin, "A YANG Data Model for Traffic Engineering Tunnels, Label Switched Paths and Interfaces", Work in Progress, Internet-Draft, draft-ietf-teas-yang-te-38, 29 May 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-teas-yang-te-38>.

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

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

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

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

Acknowledgments

This document was prepared using kramdown.

Authors' Addresses

Chaode Yu
Huawei Technologies
Email: yuchaode@huawei.com

Sergio Belotti
Nokia
Email: sergio.belotti@nokia.com

Italo Busi
Huawei Technologies
Email: italo.busi@huawei.com

Aihua Guo
Futurewei Technologies

Email: aihuaguo.ietf@gmail.com

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