Workgroup: OPSAWG
Internet-Draft:
draft-ietf-opsawg-teas-common-ac-00
Published: 6 November 2023
Intended Status: Standards Track
Expires: 9 May 2024
Authors: M. Boucadair, Ed. R. Roberts, Ed. O. G. D. Dios
Orange Juniper Telefonica
S. B. Giraldo B. Wu
Nokia Huawei Technologies

A Common YANG Data Model for Attachment Circuits

Abstract

The document specifies a common Attachment Circuits (ACs) YANG module, which is designed with the intent to be reusable by other models. For example, this common model can be reused by service models to expose ACs as a service, service models that require binding a service to a set of ACs, network and device models to provision ACs, etc.

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 9 May 2024.

document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

Connectivity services are provided by networks to customers via dedicated terminating points (e.g., service functions, Customer Premises Equipment (CPEs), Autonomous System Border Routers (ASBRs), data centers gateways, Internet Exchange Points). A connectivity service is basically about ensuring data transfer received from (or destined to) a given terminating point to (or from) other terminating points that belong to the same customer/service, an interconnection node, or an ancillary node. A set of objectives for the connectivity service may eventually be negotiated and agreed upon between a customer a network provider. For that data transfer to take place within the provider network, it is assumed that adequate setup is provisioned over the links that connect customer terminating points and a provider network so that data can be successfully exchanged over these links. The required setup is referred to in this document as Attachment Circuits (ACs), while the underlying link is referred to as "bearers".

This document adheres to the definition of an Attachment Circuit as provided in Section 1.2 of [RFC4364], especially:

> Routers can be attached to each other, or to end systems, in a variety of different ways: PPP connections, ATM Virtual Circuits (VCs), Frame Relay VCs, ethernet interfaces, Virtual Local Area Networks (VLANs) on ethernet interfaces, GRE tunnels, Layer 2 Tunneling Protocol (L2TP) tunnels, IPsec tunnels, etc. We will use the term "attachment circuit" to refer generally to some such

> means of attaching to a router. An attachment circuit may be the sort of connection that is usually thought of as a "data link", or it may be a tunnel of some sort; what matters is that it be possible for two devices to be network layer peers over the attachment circuit.

When a customer requests a new value-added service, the service can be bound to existing attachment circuits or trigger the instantiation of new attachment circuits. Whether these AC are specific to a given service or be used to deliver a variety of services is deployment specific.

An example of ACs is depicted in Figure 1. A Customer Edge (CE) may be a physical node or a logical entity. A CE is seen by the network as a peer Service Attachment Point (SAP) [RFC9408]. CEs may be dedicated to one single service (e.g., Layer 3 VPN, Layer 2 VPN) or host multiple services (e.g., Service Functions [RFC7665]). A single AC (as seen by a network provider) may be bound to one or multiple peer SAPs (e.g., CE#1 and CE#2). For example, and as discussed in [RFC4364], multiple CEs can be attached to a PE over the same attachment circuit. This is typically implemented if the layer 2 infrastructure between the CE and the network provides a multipoint service. The same CE may terminate multiple ACs. These ACes may be over the same or distinct bearers.

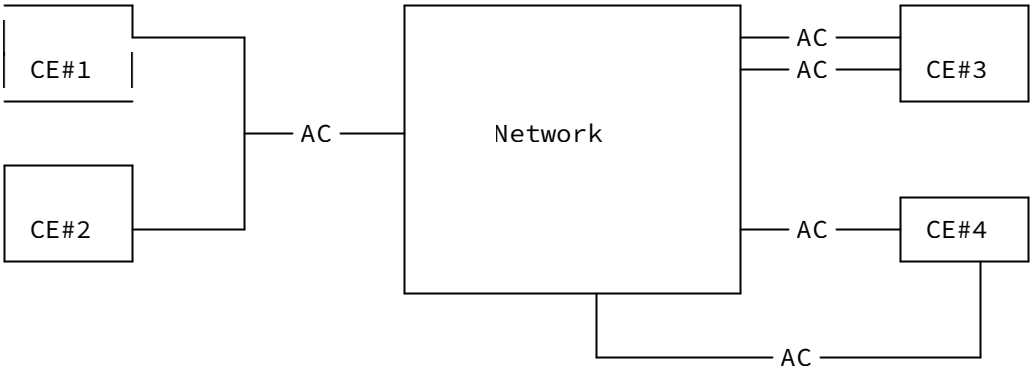

Figure 1: Examples of ACs

This document specifies a common module ("ietf-ac-common") for ACS. The model is designed with the intent to be reusable by other models and, therefore, ensure consistent AC structures among modules that manipulate ACs. For example, the common model can be reused by service models to expose AC as a service (e.g., [I-D.boro-opsawg-teas-attachment-circuit]), service models that require binding a service to a set of ACs (e.g., [I-D.ietf-teas-ietf-network-slice-nbi-yang])), network models to

provision ACs (e.g., [I-D.boro-opsawg-ntw-attachment-circuit]), device models, etc.

The common AC module eases data inheritance between modules (e.g., from service to network models as per [RFC8969]).

The YANG data models in this document conform to the Network Management Datastore Architecture (NMDA) defined in [RFC8342].

2. 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.

The meanings of the symbols in the YANG tree diagrams are defined in [RFC8340].

This document uses the following terms:

Bearer: A physical or logical link that connects a customer node (or site) to a provider network. A bearer can be a wireless or wired link. One or multiple technologies can be used to build a bearer. The bearer type can be specified by a customer.

The operator allocates a unique bearer reference to identify a bearer within its network (e.g., customer line identifier). Such a reference can be retrieved by a customer and used in subsequent service placement requests to unambiguously identify where a service is to be bound.

The concept of bearer can be generalized to refer to the required underlying connection for the provisioning of an attachment circuit. One or multiple attachment circuits may be hosted over the same bearer (e.g., multiple VLANs on the same bearer that is provided by a physical link).

Network controller: Denotes a functional entity responsible for the management of the service provider network.

Service orchestrator: Refers to a functional entity that interacts with the customer of a network service. The service orchestrator is typically responsible for the attachment circuits, the

Provider Edge (PE) selection, and requesting the activation of the requested service to a network controller.

Service provider network: A network that is able to provide network services (e.g., Layer 3 VPN, Layer 2 VPN, and Network Slice Services).

Service provider: A service provider that offers network services (e.g., Layer 3 VPN, Layer 2 VPN, and Network Slice Services).

3. Description of the AC Common YANG Module

The full tree of the "ietf-ac-common" module is available at [AC-Common-Tree].

3.1. Identities

The module defines a set of identities, including the following:

'address-allocation-type': Used to specify the IP address allocation type in an AC. For example, this identity can used to indicate whether the provider network provides DHCP service, DHCP relay, or static addressing. Note that for the IPv6 case, Stateless Address Autoconfiguration (SLAAC) [RFC4862] can be used.

'local-defined-next-hop': Used to specify next hop actions. For example, this identity can be used to indicate an action to discard traffic for a given destination or treat traffic towards addresses within the specified next-hop prefix as though they are connected to a local link.

'l2-tunnel-type': Uses to control the Layer 2 tunnel selection for an AC. The current version supports indicating pseudowire, Virtual Private LAN Service (VPLS), and Virtual eXtensible Local Area Network (VXLAN).

'precedence-type': Used to indicate the redundancy type when requesting ACs. For example, this identity can be used to tag primary and secondary ACs.

3.2. Reusable Groupings

The module also defines a set of reusable groupings, including the following:

'op-instructions' (Figure 2): Defines a set of parameters to specify scheduling instructions and report related events for a service request (e.g., AC or bearer).

```
grouping op-instructions
  +-- requested-start?   yang:date-and-time
  +-- requested-stop?    yang:date-and-time
  +--ro actual-start?      yang:date-and-time
  +--ro actual-stop?       yang:date-and-time
```

Figure 2: Operational Instructions Grouping

Layer 2 encapsulations (Figure 3): Groupings for the following encapsulation schemes are supported: dot1Q, QinQ, and priority-tagged.

Layer 2 tunnel services (Figure 3): These grouping are used to define layer 2 tunnel services that may be needed for the activation of an AC. Examples of supported Layer 2 servers are the pseudowire (Section 6.1 of [RFC8077]), VPLS, or VXLAN [RFC7348].

```
grouping dot1q
  +-- tag-type?   identityref
  +-- cvlan-id?   uint16
grouping priority-tagged
  +-- tag-type?   identityref
grouping qinq
  +-- tag-type?   identityref
  +-- svlan-id    uint16
  +-- cvlan-id    uint16
grouping pseudowire
  +-- vcid?      uint32
  +-- far-end?   union
grouping vpls
  +-- vcid?      uint32
  +-- far-end*   union
grouping vxlan
  +-- vni-id             uint32
  +-- peer-mode?         identityref
  +-- peer-ip-address*   inet:ip-address
grouping l2-tunnel-service
  +-- type?         identityref
  +-- pseudowire
  |  +-- vcid?      uint32
  |  +-- far-end?   union
  +-- vpls
  |  +-- vcid?      uint32
  |  +-- far-end*   union
  +-- vxlan
     +-- vni-id             uint32
     +-- peer-mode?         identityref
     +-- peer-ip-address*   inet:ip-address
```

Figure 3: Layer 2 Connection Groupings

Layer 3 address allocation (Figure 4): Defines both IPv4 and IPv6 groupings to specify IP address allocation over an AC. Both dynamic and static address schemes are supported.

IP connections (Figure 4):: Defines IPv4 and IPv6 grouping for managing layer 3 connectivity over an AC. Both basic and more elaborated IP connection groupings are supported.

```
grouping ipv4-allocation-type
  +-- prefix-length?             uint8
  +-- address-allocation-type?   identityref
grouping ipv6-allocation-type
  +-- prefix-length?             uint8
  +-- address-allocation-type?   identityref
grouping ipv4-connection-basic
  +-- prefix-length?                        uint8
  +-- address-allocation-type?              identityref
  +-- (allocation-type)?
     +--:(dynamic)
        +-- (provider-dhcp)?
        |  +--:(dhcp-service-type)
        |     +-- dhcp-service-type?       enumeration
        +-- (dhcp-relay)?
           +--:(customer-dhcp-servers)
              +-- customer-dhcp-servers
                 +-- server-ip-address*   inet:ipv4-address
grouping ipv6-connection-basic
  +-- prefix-length?                        uint8
  +-- address-allocation-type?              identityref
  +-- (allocation-type)?
     +--:(dynamic)
        +-- (provider-dhcp)?
        |  +--:(dhcp-service-type)
        |     +-- dhcp-service-type?       enumeration
        +-- (dhcp-relay)?
           +--:(customer-dhcp-servers)
              +-- customer-dhcp-servers
                 +-- server-ip-address*   inet:ipv6-address
grouping ipv4-connection
  +-- local-address?                            inet:ipv4-address
  +-- virtual-address?                          inet:ipv4-address
  +-- prefix-length?                            uint8
  +-- address-allocation-type?                  identityref
  +-- (allocation-type)?
     +--:(dynamic)
     |  +-- (address-assign)?
     |  |  +--:(number)
     |  |  |  +-- number-of-dynamic-address?   uint16
     |  |  +--:(explicit)
     |  |     +-- customer-addresses
     |  |        +-- address-pool* [pool-id]
     |  |           +-- pool-id?         string
     |  |           +-- start-address    inet:ipv4-address
     |  |           +-- end-address?     inet:ipv4-address
     |  +-- (provider-dhcp)?
     |  |  +--:(dhcp-service-type)
     |  |     +-- dhcp-service-type?           enumeration
```

```
      |  +-- (dhcp-relay)?
      |     +--:(customer-dhcp-servers)
      |        +-- customer-dhcp-servers
      |           +-- server-ip-address*   inet:ipv4-address
      +--:(static-addresses)
         +-- address* [address-id]
            +-- address-id?         string
            +-- customer-address?   inet:ipv4-address
  grouping ipv6-connection
    +-- local-address?                          inet:ipv6-address
    +-- virtual-address?                        inet:ipv6-address
    +-- prefix-length?                          uint8
    +-- address-allocation-type?                identityref
    +-- (allocation-type)?
       +--:(dynamic)
       |  +-- (address-assign)?
       |  |  +--:(number)
       |  |  |  +-- number-of-dynamic-address?   uint16
       |  |  +--:(explicit)
       |  |     +-- customer-addresses
       |  |        +-- address-pool* [pool-id]
       |  |           +-- pool-id?         string
       |  |           +-- start-address    inet:ipv6-address
       |  |           +-- end-address?     inet:ipv6-address
       |  +-- (provider-dhcp)?
       |  |  +--:(dhcp-service-type)
       |  |     +-- dhcp-service-type?           enumeration
       |  +-- (dhcp-relay)?
       |     +--:(customer-dhcp-servers)
       |        +-- customer-dhcp-servers
       |           +-- server-ip-address*   inet:ipv6-address
       +--:(static-addresses)
          +-- address* [address-id]
             +-- address-id?         string
             +-- customer-address?   inet:ipv6-address
```

Figure 4: Layer 3 Connection Groupings

Routing parameters (Figure 5): In addition to static routing, the module supports the following routing protocols: BGP [RFC4271], OSPF [RFC4577] or [RFC6565], IS-IS [ISO10589][RFC1195][RFC5308], and RIP [RFC2453]. For all supported routing protocols, 'address-family' indicates whether IPv4, IPv6, or both address families are to be activated. For example, this parameter is used to determine whether RIPv2 [RFC2453], RIP Next Generation (RIPng),

or both are to be enabled [RFC2080]. More details about supported routing groupings are provided hereafter:

* Authentication: These groupings include the required information to manage the authentication of OSPF, IS-IS, BGP, and RIP. Similar to [RFC9182], this version of the common AC model assumes that parameters specific to the TCP-AO are preconfigured as part of the key chain that is referenced in the model. No assumption is made about how such a key chain is preconfigured. However, the structure of the key chain should cover data nodes beyond those in [RFC8177], mainly SendID and RecvID (Section 3.1 of [RFC5925]).

* BGP peer groups: Includes a set of parameters to identify a BGP peer group. Such a group can be defined by providing a local AS Number (ASN), a customer's ASN, and the address families to be activated for this group. BGP peer groups can be identified by a name.

* Basic parameters: These groupings include the minimal set of routing configuration that is required for the activation of OSPF, IS-IS, BGP, and RIP.

* Static routing: Parameters to configure an entry of a list of IP static routing entries.

```
grouping bgp-authentication
  +-- authentication
     +-- enable?            boolean
     +-- keying-material
        +-- (option)?
           +--:(ao)
           |  +-- enable-ao?          boolean
           |  +-- ao-keychain?        key-chain:key-chain-ref
           +--:(md5)
           |  +-- md5-keychain?       key-chain:key-chain-ref
           +--:(explicit)
              +-- key-id?             uint32
              +-- key?                string
              +-- crypto-algorithm?   identityref
grouping ospf-authentication
  +-- authentication
     +-- enable?            boolean
     +-- keying-material
        +-- (option)?
           +--:(auth-key-chain)
           |  +-- key-chain?          key-chain:key-chain-ref
           +--:(auth-key-explicit)
              +-- key-id?             uint32
              +-- key?                string
              +-- crypto-algorithm?   identityref
grouping isis-authentication
  +-- authentication
     +-- enable?            boolean
     +-- keying-material
        +-- (option)?
           +--:(auth-key-chain)
           |  +-- key-chain?          key-chain:key-chain-ref
           +--:(auth-key-explicit)
              +-- key-id?             uint32
              +-- key?                string
              +-- crypto-algorithm?   identityref
grouping rip-authentication
  +-- authentication
     +-- enable?            boolean
     +-- keying-material
        +-- (option)?
           +--:(auth-key-chain)
           |  +-- key-chain?          key-chain:key-chain-ref
           +--:(auth-key-explicit)
              +-- key?                string
              +-- crypto-algorithm?   identityref
grouping bgp-peer-group-without-name
  +--ro local-as?         inet:as-number
  +-- peer-as?          inet:as-number
```

```
  +-- address-family?   identityref
grouping bgp-peer-group-with-name
  +-- name?             string
  +--ro local-as?         inet:as-number
  +-- peer-as?          inet:as-number
  +-- address-family?   identityref
grouping ospf-basic
  +-- address-family?   identityref
  +-- area-id           yang:dotted-quad
  +-- metric?           uint16
grouping isis-basic
  +-- address-family?   identityref
  +-- area-address      area-address
grouping ipv4-static-rtg-entry
  +-- lan?        inet:ipv4-prefix
  +-- lan-tag?    string
  +-- next-hop?   union
  +-- metric?     uint32
grouping ipv4-static-rtg
  +-- ipv4-lan-prefixes* [lan next-hop] {vpn-common:ipv4}?
     +-- lan?        inet:ipv4-prefix
     +-- lan-tag?    string
     +-- next-hop?   union
     +-- metric?     uint32
     +-- status
        +-- admin-status
        |  +-- status?        identityref
        |  +-- last-change?   yang:date-and-time
        +--ro oper-status
           +--ro status?        identityref
           +--ro last-change?   yang:date-and-time
grouping ipv6-static-rtg-entry
  +-- lan?        inet:ipv6-prefix
  +-- lan-tag?    string
  +-- next-hop?   union
  +-- metric?     uint32
grouping ipv6-static-rtg
  +-- ipv6-lan-prefixes* [lan next-hop] {vpn-common:ipv6}?
     +-- lan?        inet:ipv6-prefix
     +-- lan-tag?    string
     +-- next-hop?   union
     +-- metric?     uint32
     +-- status
        +-- admin-status
        |  +-- status?        identityref
        |  +-- last-change?   yang:date-and-time
        +--ro oper-status
           +--ro status?        identityref
           +--ro last-change?   yang:date-and-time
```

```
grouping bfd
  +-- holdtime?   uint32
```

Figure 5: Layer 3 Connection Groupings

Bandwidth parameters (Figure 6): Bandwidth parameters can be represented using the Committed Information Rate (CIR), the Excess Information Rate (EIR), or the Peak Information Rate (PIR).

These parameters can be provided per bandwidth type. The following types, defined in [RFC9181], can be used to indicate the bandwidth type:

'bw-per-cos': The bandwidth is per Class of Service (CoS).
'bw-per-port': The bandwidth is per AC.
'bw-per-site': The bandwidth is to all ACs that belong to the same site

```
grouping bandwidth-parameters
  +-- cir?   uint64
  +-- cbs?   uint64
  +-- eir?   uint64
  +-- ebs?   uint64
  +-- pir?   uint64
  +-- pbs?   uint64
grouping bandwidth-per-type
  +-- bandwidth* [bw-type]
     +-- bw-type?     identityref
     +-- (type)?
        +--:(per-cos)
        |  +-- cos* [cos-id]
        |     +-- cos-id?   uint8
        |     +-- cir?      uint64
        |     +-- cbs?      uint64
        |     +-- eir?      uint64
        |     +-- ebs?      uint64
        |     +-- pir?      uint64
        |     +-- pbs?      uint64
        +--:(other)
           +-- cir?   uint64
           +-- cbs?   uint64
           +-- eir?   uint64
           +-- ebs?   uint64
           +-- pir?   uint64
           +-- pbs?   uint64
```

Figure 6: Bandwidth Groupings

4. Common Attachment Circuit YANG Module

This module uses types defined in [RFC6991], [RFC8177], and [RFC9181].

```
<CODE BEGINS> file ietf-ac-common@2022-11-30.yang
module ietf-ac-common {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ac-common";
  prefix ac-common;

  import ietf-vpn-common {
    prefix vpn-common;
    reference
      "RFC 9181: A Common YANG Data Model for Layer 2 and Layer 3
                 VPNs";
  }
  import ietf-netconf-acm {
    prefix nacm;
    reference
      "RFC 8341: Network Configuration Access Control Model";
  }
  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types, Section 4";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types, Section 3";
  }
  import ietf-key-chain {
    prefix key-chain;
    reference
      "RFC 8177: YANG Data Model for Key Chains";
  }

  organization
    "IETF OPSAWG (Operations and Management Area Working Group)";
  contact
    "WG Web:   <https://datatracker.ietf.org/wg/opsawg/>
     WG List:  <mailto:opsawg@ietf.org>

     Editor:   Mohamed Boucadair
               <mailto:mohamed.boucadair@orange.com>
     Author:   Richard Roberts
               <mailto:rroberts@juniper.net>
     Author:   Oscar Gonzalez de Dios
               <mailto:oscar.gonzalezdedios@telefonica.com>
     Author:   Samier Barguil
               <mailto:ssamier.barguil_giraldo@nokia.com>
     Author:   Bo Wu
               <mailto:lana.wubo@huawei.com>";
```

```
description
  "This YANG module defines a YANG model common to attachment
   circuits.

   Copyright (c) 2023 IETF Trust and the persons identified as
   authors of the code.  All rights reserved.

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

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

revision 2022-11-30 {
  description
    "Initial revision.";
  reference
    "RFC xxxx: A Common YANG Data Model for Attachment Circuits";
}

/***************************Identities***********************/
// IP address allocation types

identity address-allocation-type {
  description
    "Base identity for address allocation type in the AC.";
}

identity provider-dhcp {
  base address-allocation-type;
  description
    "The provider's network provides a DHCP service to the
     customer.";
}

identity provider-dhcp-relay {
  base address-allocation-type;
  description
    "The provider's network provides a DHCP relay service to the
     customer.";
}

identity provider-dhcp-slaac {
  if-feature "vpn-common:ipv6";
  base address-allocation-type;
  description
```

```
    "The provider's network provides a DHCP service to the customer
     as well as IPv6 Stateless Address Autoconfiguration (SLAAC).";
  reference
    "RFC 4862: IPv6 Stateless Address Autoconfiguration";
}

identity static-address {
  base address-allocation-type;
  description
    "The provider's network provides static IP addressing to the
     customer.";
}

identity slaac {
  if-feature "vpn-common:ipv6";
  base address-allocation-type;
  description
    "The provider's network uses IPv6 SLAAC to provide addressing
     to the customer.";
  reference
    "RFC 4862: IPv6 Stateless Address Autoconfiguration";
}

identity dynamic-infra {
  base address-allocation-type;
  description
    "The IP address is dynamically allocated by the hosting
     infrastrcture.";
}

// next-hop actions

identity local-defined-next-hop {
  description
    "Base identity of local defined next hops.";
}

identity discard {
  base local-defined-next-hop;
  description
    "Indicates an action to discard traffic for the corresponding
     destination. For example, this can be used to black-hole
     traffic.";
}

identity local-link {
  base local-defined-next-hop;
  description
    "Treat traffic towards addresses within the specified next-hop
```

```
     prefix as though they are connected to a local link.";
}

// Layer 2 tunnel types

identity l2-tunnel-type {
  description
    "Base identity for Layer 2 tunnel selection for an AC.";
}

identity pseudowire {
  base l2-tunnel-type;
  description
    "Pseudowire tunnel termination for the AC.";
}

identity vpls {
  base l2-tunnel-type;
  description
    "Virtual Private LAN Service (VPLS) tunnel termination for
     the AC.";
}

identity vxlan {
  base l2-tunnel-type;
  description
    "Virtual eXtensible Local Area Network (VXLAN) tunnel
     termination for the AC.";
}

// Tagging precedence

identity precedence-type {
  description
    "Redundancy type. The service can be created with primary and
     secondary tagging.";
}

identity primary {
  base precedence-type;
  description
    "Identifies the main attachment circuit.";
}

identity secondary {
  base precedence-type;
  description
    "Identifies the secondary attachment circuit.";
}
```

```
/***************************Typedefs************************/

typedef predefined-next-hop {
  type identityref {
    base local-defined-next-hop;
  }
  description
    "Predefined next-hop designation for locally generated routes.";
}

typedef area-address {
  type string {
    pattern '[0-9A-Fa-f]{2}(\.[0-9A-Fa-f]{4}){0,6}';
  }
  description
    "This type defines the area address format.";
}

/************************Reusable groupings*******************/

/**** A set of profiles ****/

grouping ac-profile-cfg {
  description
    "Grouping for AC profile configuration.";
 container valid-provider-identifiers {
  description
    "Container for valid provider profile identifiers.
     The profiles only have significance within the service
     provider's administrative domain.";
   list encryption-profile-identifier {
    key "id";
    description
      "List of encryption profile identifiers.";
    leaf id {
      type string;
      description
        "Identification of the encryption profile to be used.";
    }
  }
  list qos-profile-identifier {
    key "id";
      description
        "List of QoS profile identifiers.";
      leaf id {
        type string;
        description
          "Identification of the QoS profile to be used.";
      }
```

```
    }
    list bfd-profile-identifier {
      key "id";
      description
        "List of BFD profile identifiers.";
      leaf id {
        type string;
        description
          "Identification of the BFD profile to be used.";
      }
    }
    list forwarding-profile-identifier {
      key "id";
      description
        "List of forwarding profile identifiers.";
      leaf id {
        type string;
        description
          "Identification of the forwarding profile to be used.";
      }
    }
    list routing-profile-identifier {
      key "id";
      description
        "List of routing profile identifiers.";
      leaf id {
        type string;
        description
          "Identification of the routing profile to be used by
           the routing protocols over an AC.";
      }
    }
    nacm:default-deny-write;
  }
}

/**** Operational instructions ****/

grouping op-instructions {
  description
    "Scheduling instructions.";
  leaf requested-start {
    type yang:date-and-time;
    description
      "Indicates the requested date and time when the service is
       expected to be active.";
  }
  leaf requested-stop {
    type yang:date-and-time;
```

```
    description
      "Indicates the requested date and time when the service is
       expected to be disabled.";
  }
  leaf actual-start {
    type yang:date-and-time;
    config false;
    description
      "Indciates the actual date and time when the service
       actually was enabled.";
  }
  leaf actual-stop {
    type yang:date-and-time;
    config false;
    description
      "Indciates the actual date and time when the service
       actually was disabled.";
  }
}

/**** Layer 2 encapsulations ****/
// Dot1q

grouping dot1q {
  description
    "Defines a grouping for tagged interfaces.";
  leaf tag-type {
    type identityref {
      base vpn-common:tag-type;
    }
    description
      "Tag type.";
  }
  leaf cvlan-id {
    type uint16 {
      range "1..4094";
    }
    description
      "VLAN identifier.";
  }
}

// priority-tagged

grouping priority-tagged {
  description
    "Priority tagged.";
  leaf tag-type {
    type identityref {
```

```
      base vpn-common:tag-type;
    }
    description
      "Tag type.";
  }
}

// QinQ

grouping qinq {
  description
    "Includes QinQ parameters.";
  leaf tag-type {
    type identityref {
      base vpn-common:tag-type;
    }
    description
      "Tag type.";
  }
  leaf svlan-id {
    type uint16;
    mandatory true;
    description
      "Service VLAN (S-VLAN) identifier.";
  }
  leaf cvlan-id {
    type uint16;
    mandatory true;
    description
      "Customer VLAN (C-VLAN) identifier.";
  }
}

/**** Layer 2 tunnel services ****/
// pseudowire (PW)

grouping pseudowire {
  description
    "Includes pseudowire termination parameters.";
  leaf vcid {
    type uint32;
    description
      "Indicates a PW or virtual circuit (VC) identifier.";
  }
  leaf far-end {
    type union {
      type uint32;
      type inet:ip-address;
    }
```

```
    description
      "Neighbor reference.";
    reference
      "RFC 8077: Pseudowire Setup and Maintenance Using the Label
                 Distribution Protocol (LDP), Section 6.1";
  }
}

// VPLS

grouping vpls {
  description
    "VPLS termination parameters.";
  leaf vcid {
    type uint32;
    description
      "VC identifier.";
  }
  leaf-list far-end {
    type union {
      type uint32;
      type inet:ip-address;
    }
    description
      "Neighbor reference.";
  }
}

// VXLAN

grouping vxlan {
  description
    "VXLAN termination parameters.";
  leaf vni-id {
    type uint32;
    mandatory true;
    description
      "VXLAN Network Identifier (VNI).";
  }
  leaf peer-mode {
    type identityref {
      base vpn-common:vxlan-peer-mode;
    }
    description
      "Specifies the VXLAN access mode.  By default,
       the peer mode is set to 'static-mode'.";
  }
  leaf-list peer-ip-address {
    type inet:ip-address;
```

```
    description
      "List of a peer's IP addresses.";
  }
}

// Layer 2 Tunnel service

grouping l2-tunnel-service {
  description
    "Defines a Layer 2 tunnel termination.";
  leaf type {
    type identityref {
      base l2-tunnel-type;
    }
    description
      "Selects the tunnel termination type for an AC.";
  }
  container pseudowire {
    when "derived-from-or-self(../type, 'pseudowire')" {
      description
        "Only applies when the Layer 2 service type is
         'pseudowire'.";
    }
    description
      "Includes pseudowire termination parameters.";
    uses pseudowire;
  }
  container vpls {
    when "derived-from-or-self(../type, 'vpls')" {
      description
        "Only applies when the Layer 2 service type is 'vpls'.";
    }
    description
      "VPLS termination parameters.";
    uses vpls;
  }
  container vxlan {
    when "derived-from-or-self(../type, 'vxlan')" {
      description
        "Only applies when the Layer 2 service type is 'vxlan'.";
    }
    description
      "VXLAN termination parameters.";
    uses vxlan;
  }
}

/**** Layer 3 connection *****/
// IPv4 allocation type
```

```
grouping ipv4-allocation-type {
  description
    "IPv4-specific parameters.";
  leaf prefix-length {
    type uint8 {
      range "0..32";
    }
    description
      "Subnet prefix length expressed in bits. It is applied to both
       local and customer addresses.";
  }
  leaf address-allocation-type {
    type identityref {
      base address-allocation-type;
    }
    must "not(derived-from-or-self(current(), 'slaac') or "
       + "derived-from-or-self(current(), "
       + "'provider-dhcp-slaac'))" {
      error-message "SLAAC is only applicable to IPv6.";
    }
    description
      "Defines how IPv4 addresses are allocated to the peer site.";
  }
}

// IPv6 allocation type

grouping ipv6-allocation-type {
  description
    "IPv6-specific parameters.";
  leaf prefix-length {
    type uint8 {
      range "0..128";
    }
    description
      "Subnet prefix length expressed in bits. It is applied to both
       local and customer addresses.";
  }
  leaf address-allocation-type {
    type identityref {
      base address-allocation-type;
    }
    description
      "Defines how IPv6 addresses are allocated to the peer site.";
  }
}

// Basic parameters for IPv4 connection
```

```
grouping ipv4-connection-basic {
  description
    "Basic set fof IPv4-specific parameters for the connection.";
  uses ipv4-allocation-type;
  choice allocation-type {
    description
      "Choice of the IPv4 address allocation.";
    case dynamic {
      description
        "When the addresses are allocated by DHCP or other dynamic
         means local to the infrastructure.";
      choice provider-dhcp {
        description
          "Parameters related to DHCP-allocated addresses. IP
           addresses are allocated by DHCP, that is provided by
           the operator.";
        leaf dhcp-service-type {
          type enumeration {
            enum server {
              description
                "Local DHCP server.";
            }
            enum relay {
              description
                "Local DHCP relay.  DHCP requests are relayed to
                 a provider's server.";
            }
          }
          description
            "Indicates the type of DHCP service to be enabled on
             an AC.";
        }
      }
      choice dhcp-relay {
        description
          "The DHCP relay is provided by the operator.";
        container customer-dhcp-servers {
          description
            "Container for a list of the customer's DHCP servers.";
          leaf-list server-ip-address {
            type inet:ipv4-address;
            description
              "IPv4 addresses of the customer's DHCP server.";
          }
        }
      }
    }
  }
```

```
}

// Basic parameters for IPv6 connection

grouping ipv6-connection-basic {
  description
    "Basic set fof IPv6-specific parameters for the connection.";
  uses ipv6-allocation-type;
  choice allocation-type {
    description
      "Choice of the IPv6 address allocation.";
    case dynamic {
      description
        "When the addresses are allocated by DHCP or other dynamic
         means local to the infrastructure.";
      choice provider-dhcp {
        description
          "Parameters related to DHCP-allocated addresses.
           IP addresses are allocated by DHCP, that is provided
           by the operator.";
        leaf dhcp-service-type {
          type enumeration {
            enum server {
              description
                "Local DHCP server.";
            }
            enum relay {
              description
                "Local DHCP relay.  DHCP requests are relayed to a
                 provider's server.";
            }
          }
          description
            "Indicates the type of DHCP service to be enabled on
             the AC.";
        }
      }
      choice dhcp-relay {
        description
          "The DHCP relay is provided by the operator.";
        container customer-dhcp-servers {
          description
            "Container for a list of the customer's DHCP servers.";
          leaf-list server-ip-address {
            type inet:ipv6-address;
            description
              "IPv6 addresses of the customer's DHCP server.";
          }
        }
```

```
      }
    }
  }
}

// Full parameters for the IPv4 connection

grouping ipv4-connection {
  description
    "IPv4-specific parameters.";
  leaf local-address {
    type inet:ipv4-address;
    description
      "The IP address used at the provider's interface.";
  }
  leaf virtual-address {
    type inet:ipv4-address;
    description
      "This addresss may be used for redundancy purposes.";
  }
  uses ipv4-allocation-type;
  choice allocation-type {
    description
      "Choice of the IPv4 address allocation.";
    case dynamic {
      description
        "When the addresses are allocated by DHCP or other
         dynamic means local to the infrastructure.";
      choice address-assign {
        default "number";
        description
          "A choice for how IPv4 addresses are assigned.";
        case number {
          leaf number-of-dynamic-address {
            type uint16;
            description
              "Specifies the number of IP addresses to be assigned
               to the customer on the AC.";
          }
        }
        case explicit {
          container customer-addresses {
            description
              "Container for customer addresses to be allocated
               using DHCP.";
            list address-pool {
              key "pool-id";
              description
                "Describes IP addresses to be dyncamically allocated.
```

```
            When only 'start-address' is present, it represents a
            single address.

            When both 'start-address' and 'end-address' are
            specified, it implies a range inclusive of both
            addresses.";
          leaf pool-id {
            type string;
            description
              "A pool identifier for the address range from
               'start-address' to 'end-address'.";
          }
          leaf start-address {
            type inet:ipv4-address;
            mandatory true;
            description
              "Indicates the first address in the pool.";
          }
          leaf end-address {
            type inet:ipv4-address;
            description
              "Indicates the last address in the pool.";
          }
        }
      }
    }
  }
  choice provider-dhcp {
    description
      "Parameters related to DHCP-allocated addresses. IP
       addresses are allocated by DHCP, which is provided by
       the operator.";
    leaf dhcp-service-type {
      type enumeration {
        enum server {
          description
            "Local DHCP server.";
        }
        enum relay {
          description
            "Local DHCP relay.  DHCP requests are relayed to
             a provider's server.";
        }
      }
      description
        "Indicates the type of DHCP service to be enabled on
         this AC.";
    }
```

```
      }
      choice dhcp-relay {
        description
          "The DHCP relay is provided by the operator.";
        container customer-dhcp-servers {
          description
            "Container for a list of the customer's DHCP servers.";
          leaf-list server-ip-address {
            type inet:ipv4-address;
            description
              "IPv4 addresses of the customer's DHCP server.";
          }
        }
      }
    }
    case static-addresses {
      description
        "Lists the IPv4 addresses that are used.";
      list address {
        key "address-id";
        ordered-by user;
        description
          "Lists the IPv4 addresses that are used. The first address o
           the list is the primary address of the connection.";
        leaf address-id {
          type string;
          description
            "An identifier of the static IPv4 address.";
        }
        leaf customer-address {
          type inet:ipv4-address;
          description
            "An IPv4 address of the customer side.";
        }
      }
    }
  }
}

// Full parameters for the IPv6 connection

grouping ipv6-connection {
  description
    "IPv6-specific parameters.";
  leaf local-address {
    type inet:ipv6-address;
    description
      "IPv6 address of the provider side.";
  }
```

```
leaf virtual-address {
  type inet:ipv6-address;
  description
    "This addresss may be used for redundancy purposes.";
}
uses ipv6-allocation-type;
choice allocation-type {
  description
    "Choice of the IPv6 address allocation.";
  case dynamic {
    description
      "When the addresses are allocated by DHCP or other
       dynamic means local to the infrastructure.";
    choice address-assign {
      default "number";
      description
        "A choice for how IPv6 addresses are assigned.";
      case number {
        leaf number-of-dynamic-address {
          type uint16;
          description
            "Specifies the number of IP addresses to be assigned to
             the customer on this access.";
        }
      }
      case explicit {
        container customer-addresses {
          description
            "Container for customer addresses to be allocated
             using DHCP.";
          list address-pool {
            key "pool-id";
            description
              "Describes IP addresses to be dyncamically allocated.

               When only 'start-address' is present, it represents a
               single address.

               When both 'start-address' and 'end-address' are
               specified, it implies a range inclusive of both
               addresses.";
            leaf pool-id {
              type string;
              description
                "A pool identifier for the address range from
                 'start-address' to 'end-address'.";
            }
            leaf start-address {
              type inet:ipv6-address;
```

```
            mandatory true;
            description
              "Indicates the first address in the pool.";
          }
          leaf end-address {
            type inet:ipv6-address;
            description
              "Indicates the last address in the pool.";
          }
        }
      }
    }
  }
  choice provider-dhcp {
    description
      "Parameters related to DHCP-allocated addresses.
       IP addresses are allocated by DHCP, which is provided
       by the operator.";
    leaf dhcp-service-type {
      type enumeration {
        enum server {
          description
            "Local DHCP server.";
        }
        enum relay {
          description
            "Local DHCP relay.  DHCP requests are relayed
             to a provider's server.";
        }
      }
      description
        "Indicates the type of DHCP service to
         be enabled on this access.";
    }
  }
  choice dhcp-relay {
    description
      "The DHCP relay is provided by the operator.";
    container customer-dhcp-servers {
      description
        "Container for a list of the customer's DHCP servers.";
      leaf-list server-ip-address {
        type inet:ipv6-address;
        description
          "IPv6 addresses of the customer's DHCP server.";
      }
    }
  }
}
```

```
    case static-addresses {
      description
        "Lists the IPv6 addresses that are used.";
      list address {
        key "address-id";
        ordered-by user;
        description
          "Lists the IPv6 addresses that are used. The first address
           of the list is the primary IP address of the connection.";
        leaf address-id {
          type string;
          description
            "An identifier of the static IPv6 address.";
        }
        leaf customer-address {
          type inet:ipv6-address;
          description
            "An IPv6 address of the customer side.";
        }
      }
    }
  }
}

/**** Routing ****/
// Routing authentication

grouping bgp-authentication {
  description
    "Grouping for BGP authentication parameters.";
  container authentication {
    description
      "Container for BGP authentication  parameters.";
    leaf enable {
      type boolean;
      default "false";
      description
        "Enables or disables authentication.";
    }
    container keying-material {
      when "../enable = 'true'";
      description
        "Container for describing how a BGP routing session is to
         be secured on an AC.";
      choice option {
        description
          "Choice of authentication options.";
        case ao {
          description
```

```
      "Uses the TCP Authentication Option (TCP-AO).";
    reference
      "RFC 5925: The TCP Authentication Option";
    leaf enable-ao {
      type boolean;
      description
        "Enables the TCP-AO.";
    }
    leaf ao-keychain {
      type key-chain:key-chain-ref;
      description
        "Reference to the TCP-AO key chain.";
      reference
        "RFC 8177: YANG Data Model for Key Chains";
    }
  }
  case md5 {
    description
      "Uses MD5 to secure the session.";
    reference
      "RFC 4364: BGP/MPLS IP Virtual Private Networks
                 (VPNs), Section 13.2";
    leaf md5-keychain {
      type key-chain:key-chain-ref;
      description
        "Reference to the MD5 key chain.";
      reference
        "RFC 8177: YANG Data Model for Key Chains";
    }
  }
  case explicit {
    leaf key-id {
      type uint32;
      description
        "Key identifier.";
    }
    leaf key {
      type string;
      description
        "BGP authentication key.

         This model only supports the subset of keys that
         are representable as ASCII strings.";
    }
    leaf crypto-algorithm {
      type identityref {
        base key-chain:crypto-algorithm;
      }
      description
```

```
              "Indicates the cryptographic algorithm associated
               with the key.";
          }
        }
      }
    }
  }
}

grouping ospf-authentication {
  description
    "Authentication configuration.";
  container authentication {
    description
      "Container for OSPF authentication  parameters.";
    leaf enable {
      type boolean;
      default "false";
      description
        "Enables or disables authentication.";
    }
    container keying-material {
      when "../enable = 'true'";
      description
        "Container for describing how an OSPF session is to be secured
         for this AC.";
      choice option {
        description
          "Options for OSPF authentication.";
        case auth-key-chain {
          leaf key-chain {
            type key-chain:key-chain-ref;
            description
              "Name of the key chain.";
          }
        }
        case auth-key-explicit {
          leaf key-id {
            type uint32;
            description
              "Key identifier.";
          }
          leaf key {
            type string;
            description
              "OSPF authentication key.

               This model only supports the subset of keys that
               are representable as ASCII strings.";
```

```
          }
          leaf crypto-algorithm {
            type identityref {
              base key-chain:crypto-algorithm;
            }
            description
              "Indicates the cryptographic algorithm associated with
               the key.";
          }
        }
      }
    }
  }
}

grouping isis-authentication {
  description
    "IS-IS authentication configuration.";
  container authentication {
    description
      "Container for IS-IS authentication  parameters.";
    leaf enable {
      type boolean;
      default "false";
      description
        "Enables or disables authentication.";
    }
    container keying-material {
      when "../enable = 'true'";
      description
        "Container for describing how an IS-IS session is secured
         over an AC.";
      choice option {
        description
          "Options for IS-IS authentication.";
        case auth-key-chain {
          leaf key-chain {
            type key-chain:key-chain-ref;
            description
              "Name of the key chain.";
          }
        }
        case auth-key-explicit {
          leaf key-id {
            type uint32;
            description
              "Key identifier.";
          }
          leaf key {
```

```
            type string;
            description
              "IS-IS authentication key.

               This model only supports the subset of keys that
               are representable as ASCII strings.";
          }
          leaf crypto-algorithm {
            type identityref {
              base key-chain:crypto-algorithm;
            }
            description
              "Indicates the cryptographic algorithm associated with
               the key.";
          }
        }
      }
    }
  }
}

grouping rip-authentication {
  description
    "RIP authentication configuration.";
  container authentication {
    description
      "Container for RIP authentication  parameters.";
    leaf enable {
      type boolean;
      default "false";
      description
        "Enables or disables authentication.";
    }
    container keying-material {
      when "../enable = 'true'";
      description
        "Container for describing how a RIP session is to be secured
         on this AC.";
      choice option {
        description
          "Specifies the authentication
           scheme.";
        case auth-key-chain {
          leaf key-chain {
            type key-chain:key-chain-ref;
            description
              "Name of the key chain.";
          }
        }
```

```
        case auth-key-explicit {
          leaf key {
            type string;
            description
              "RIP authentication key.

               This model only supports the subset of keys that
               are representable as ASCII strings.";
          }
          leaf crypto-algorithm {
            type identityref {
              base key-chain:crypto-algorithm;
            }
            description
              "Indicates the cryptographic algorithm associated with
               the key.";
          }
        }
      }
    }
  }
}

// Basic routing parameters

grouping bgp-peer-group-without-name {
  description
    "Identifies a BGP peer-group configured on the local system.";
  leaf local-as {
    type inet:as-number;
    config false;
    description
      "Indicates a local AS Number (ASN). This ASN is exposed
       to a customer so that it knows which ASN to use
       to set up a BGP session.";
  }
  leaf peer-as {
    type inet:as-number;
    description
      "Indicates the customer's ASN when the customer
       requests BGP routing.";
  }
  leaf address-family {
    type identityref {
      base vpn-common:address-family;
    }
    description
      "This node contains the address families to be activated.
       'dual-stack' means that both IPv4 and IPv6 will be activated.";
```

```
  }
}

grouping bgp-peer-group-with-name {
  description
    "Identifies a BGP peer-group configured on the local system -
     identified by a peer-group name";
  leaf name {
    type string;
    description
      "Name of the BGP peer-group";
  }
  uses bgp-peer-group-without-name;
}

grouping ospf-basic {
  description
    "Configuration specific to OSPF.";
  leaf address-family {
    type identityref {
      base vpn-common:address-family;
    }
    description
      "Indicates whether IPv4, IPv6, or both are to be activated.";
  }
  leaf area-id {
    type yang:dotted-quad;
    mandatory true;
    description
      "Area ID.";
    reference
      "RFC 4577: OSPF as the Provider/Customer Edge Protocol
                 for BGP/MPLS IP Virtual Private Networks
                 (VPNs), Section 4.2.3
       RFC 6565: OSPFv3 as a Provider Edge to Customer Edge
                 (PE-CE) Routing Protocol, Section 4.2";
  }
  leaf metric {
    type uint16;
    default "1";
    description
      "Metric of the AC.  It is used in the routing state
       calculation and path selection.";
  }
}

grouping isis-basic {
  description
    "Basic configuration specific to IS-IS.";
```

```
  leaf address-family {
    type identityref {
      base vpn-common:address-family;
    }
    description
      "Indicates whether IPv4, IPv6, or both are to be activated.";
  }
  leaf area-address {
    type area-address;
    mandatory true;
    description
      "Area address.";
  }
}

// Static routing

grouping ipv4-static-rtg-entry {
  description
    "Paramters to configure a specific IPv4 static routing entry.";
  leaf lan {
    type inet:ipv4-prefix;
    description
      "LAN prefixes.";
  }
  leaf lan-tag {
    type string;
    description
      "Internal tag to be used in service policies.";
  }
  leaf next-hop {
    type union {
      type inet:ip-address;
      type predefined-next-hop;
    }
    description
      "The next hop that is to be used for the static route.
       This may be specified as an IP address or a
       predefined next-hop type (e.g., 'discard' or
       'local-link').";
  }
  leaf metric {
    type uint32;
    description
      "Indicates the metric associated with the static route.";
  }
}

grouping ipv4-static-rtg {
```

```
  description
    "Configuration specific to IPv4 static routing.";
  list ipv4-lan-prefixes {
    if-feature "vpn-common:ipv4";
    key "lan next-hop";
    description
      "List of LAN prefixes for the site.";
    uses ipv4-static-rtg-entry;
    uses vpn-common:service-status;
  }
}

grouping ipv6-static-rtg-entry {
  description
    "Paramters to configure a specific IPv6 static routing entry.";
  leaf lan {
    type inet:ipv6-prefix;
    description
      "LAN prefixes.";
  }
  leaf lan-tag {
    type string;
    description
      "Internal tag to be used in service (e.g., VPN) policies.";
  }
  leaf next-hop {
    type union {
      type inet:ip-address;
      type predefined-next-hop;
    }
    description
      "The next hop that is to be used for the static route.
       This may be specified as an IP address or a predefined
       next-hop type (e.g., 'discard' or 'local-link').";
  }
  leaf metric {
    type uint32;
    description
      "Indicates the metric associated with the static route.";
  }
}

grouping ipv6-static-rtg {
  description
    "Configuration specific to IPv6 static routing.";
  list ipv6-lan-prefixes {
    if-feature "vpn-common:ipv6";
    key "lan next-hop";
    description
```

```
      "List of LAN prefixes for the site.";
    uses ipv6-static-rtg-entry;
    uses vpn-common:service-status;
  }
}

// OAM: maintain or remove?

grouping bfd {
  description
    "Container for BFD.";
  leaf holdtime {
    type uint32;
    units "milliseconds";
    description
      "Expected BFD holdtime.
       The customer may impose some fixed values
       for the holdtime period if the provider allows
       the customer to use this function.
       If the provider doesn't allow the customer to
       use this function, fixed values will not be set.";
    reference
      "RFC 5880: Bidirectional Forwarding Detection (BFD),
                 Section 6.8.18";
  }
}

// QoS

grouping bandwidth-parameters {
  description
    "A grouping for bandwidth parameters.";
  leaf cir {
    type uint64;
    units "bps";
    description
      "Committed Information Rate (CIR). The maximum number of bits
       that a port can receive or send during one second over
       an interface.";
  }
  leaf cbs {
    type uint64;
    units "bytes";
    description
      "Committed Burst Size (CBS). CBS controls the bursty nature of
       the traffic.  Traffic that does not use the configured CIR
       accumulates credits until the credits reach the configured
       CBS.";
  }
```

```
  leaf eir {
    type uint64;
    units "bps";
    description
      "Excess Information Rate (EIR), i.e., excess frame delivery
       allowed not subject to a Service Level Agreement (SLA).
       The traffic rate can be limited by EIR.";
  }
  leaf ebs {
    type uint64;
    units "bytes";
    description
      "Excess Burst Size (EBS).  The bandwidth available for burst
       traffic from the EBS is subject to the amount of bandwidth
       that is accumulated during periods when traffic allocated
       by the EIR policy is not used.";
  }
  leaf pir {
    type uint64;
    units "bps";
    description
      "Peak Information Rate (PIR), i.e., maximum frame delivery
       allowed. It is equal to or less than sum of CIR and EIR.";
  }
  leaf pbs {
    type uint64;
    units "bytes";
    description
      "Peak Burst Size (PBS).";
  }
}

grouping bandwidth-per-type{
  description
    "Grouping for bandwidth per type.";
  list bandwidth {
    key "bw-type";
    description
      "List for bandwidth per type data nodes.";
    leaf bw-type {
      type identityref {
        base vpn-common:bw-type;
      }
      description
        "Indicates the bandwidth type.";
    }
    choice type {
      description
        "Choice based upon bandwidth type.";
```

```
        case per-cos {
          description
            "Bandwidth per CoS.";
          list cos {
            key "cos-id";
            description
              "List of Class of Services.";
            leaf cos-id {
              type uint8;
              description
                "Identifier of the CoS, indicated by a Differentiated
                 Services Code Point (DSCP) or a CE-CLAN CoS (802.1p)
                 value in the service frame.";
              reference
                "IEEE Std 802.1Q: Bridges and Bridged Networks";
            }
            uses bandwidth-parameters;
          }
        }
        case other {
          description
            "Other bandwidth types.";
          uses bandwidth-parameters;
        }
      }
    }
  }
}
```

<CODE ENDS>

5. Security Considerations

The YANG module specified in this document defines schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

The "ietf-ac-common" module defines a set of identities, types, and groupings. These nodes are intended to be reused by other YANG

modules. The module by itself does not expose any data nodes that are writable, data nodes that contain read-only state, or RPCs.

YANG modules that use the groupings that are defined in this document should identify the corresponding security considerations. For example, reusing some of these groupings will expose privacy-related information (e.g., 'ipv6-lan-prefixes' or 'ipv4-lan-prefixes'). Disclosing such information may be considered a violation of the customer-provider trust relationship.

Several groupings ('bgp-authentication', 'ospf-authentication', 'isis-authentication', and 'rip-authentication') rely upon [RFC8177] for authentication purposes. As such, modules that will reuse these groupings will inherit the security considerations discussed in Section 5 of [RFC8177]. Also, these groupings support supplying explicit keys as strings in ASCII format. The use of keys in hexadecimal string format would afford greater key entropy with the same number of key- string octets. However, such a format is not included in this version of the common AC model, because it is not supported by the underlying device modules (e.g., [RFC8695]).

6. IANA Considerations

IANA is requested to register the following URI in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-ac-common
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

IANA is requested to register the following YANG module in the "YANG Module Names" subregistry [RFC6020] within the "YANG Parameters" registry.

Name: ietf-ac-common
Namespace: urn:ietf:params:xml:ns:yang:ietf-ac-common
Prefix: ac-common
Maintained by IANA? N
Reference: RFC xxxx

7. References

7.1. Normative References

[ISO10589] ISO, "Information technology - Telecommunications and information exchange between systems - Intermediate System to Intermediate System intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode

network service (ISO8473)", 2002, <https://www.iso.org/standard/30932.html>.

[RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and dual environments", RFC 1195, DOI 10.17487/RFC1195, December 1990, <https://www.rfc-editor.org/rfc/rfc1195>.

[RFC2080] Malkin, G. and R. Minnear, "RIPng for IPv6", RFC 2080, DOI 10.17487/RFC2080, January 1997, <https://www.rfc-editor.org/rfc/rfc2080>.

[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>.

[RFC2453] Malkin, G., "RIP Version 2", STD 56, RFC 2453, DOI 10.17487/RFC2453, November 1998, <https://www.rfc-editor.org/rfc/rfc2453>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/rfc/rfc3688>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/rfc/rfc4271>.

[RFC4577] Rosen, E., Psenak, P., and P. Pillay-Esnault, "OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4577, DOI 10.17487/RFC4577, June 2006, <https://www.rfc-editor.org/rfc/rfc4577>.

[RFC5308] Hopps, C., "Routing IPv6 with IS-IS", RFC 5308, DOI 10.17487/RFC5308, October 2008, <https://www.rfc-editor.org/rfc/rfc5308>.

[RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <https://www.rfc-editor.org/rfc/rfc5925>.

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

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol

(NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/rfc/rfc6241>.

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

[RFC6565] Pillay-Esnault, P., Moyer, P., Doyle, J., Ertekin, E., and M. Lundberg, "OSPFv3 as a Provider Edge to Customer Edge (PE-CE) Routing Protocol", RFC 6565, DOI 10.17487/RFC6565, June 2012, <https://www.rfc-editor.org/rfc/rfc6565>.

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

[RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <https://www.rfc-editor.org/rfc/rfc7348>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/rfc/rfc8040>.

[RFC8077] Martini, L., Ed. and G. Heron, Ed., "Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)", STD 84, RFC 8077, DOI 10.17487/RFC8077, February 2017, <https://www.rfc-editor.org/rfc/rfc8077>.

[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>.

[RFC8177] Lindem, A., Ed., Qu, Y., Yeung, D., Chen, I., and J. Zhang, "YANG Data Model for Key Chains", RFC 8177, DOI 10.17487/RFC8177, June 2017, <https://www.rfc-editor.org/rfc/rfc8177>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/rfc/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture

(NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/rfc/rfc8342>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC9181] Barguil, S., Gonzalez de Dios, O., Ed., Boucadair, M., Ed., and Q. Wu, "A Common YANG Data Model for Layer 2 and Layer 3 VPNs", RFC 9181, DOI 10.17487/RFC9181, February 2022, <https://www.rfc-editor.org/rfc/rfc9181>.

7.2. Informative References

[AC-Common-Tree] "Full Common Attachment Circuit Tree Structure", 2023, <https://github.com/boucadair/attachment-circuit-model/blob/main/yang/full-trees/ac-common-with-groupings.txt>.

[I-D.boro-opsawg-ntw-attachment-circuit] Boucadair, M., Roberts, R., de Dios, O. G., Barguil, S., and B. Wu, "A Network YANG Data Model for Attachment Circuits", Work in Progress, Internet-Draft, draft-boro-opsawg-ntw-attachment-circuit-03, 5 September 2023, <https://datatracker.ietf.org/doc/html/draft-boro-opsawg-ntw-attachment-circuit-03>.

[I-D.boro-opsawg-teas-attachment-circuit] Boucadair, M., Roberts, R., de Dios, O. G., Barguil, S., and B. Wu, "YANG Data Models for 'Attachment Circuits'-as-a-Service (ACaaS)", Work in Progress, Internet-Draft, draft-boro-opsawg-teas-attachment-circuit-07, 10 July 2023, <https://datatracker.ietf.org/doc/html/draft-boro-opsawg-teas-attachment-circuit-07>.

[I-D.ietf-teas-ietf-network-slice-nbi-yang] Wu, B., Dhody, D., Rokui, R., Saad, T., and J. Mullooly, "A YANG Data Model for the IETF Network Slice Service", Work in Progress, Internet-Draft, draft-ietf-teas-ietf-network-slice-nbi-yang-08, 23 October 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-teas-ietf-network-slice-nbi-yang-08>.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <https://www.rfc-editor.org/rfc/rfc4364>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <https://www.rfc-editor.org/rfc/rfc4862>.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/rfc/rfc7665>.

[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>.

[RFC8695] Liu, X., Sarda, P., and V. Choudhary, "A YANG Data Model for the Routing Information Protocol (RIP)", RFC 8695, DOI 10.17487/RFC8695, February 2020, <https://www.rfc-editor.org/rfc/rfc8695>.

[RFC8969] Wu, Q., Ed., Boucadair, M., Ed., Lopez, D., Xie, C., and L. Geng, "A Framework for Automating Service and Network Management with YANG", RFC 8969, DOI 10.17487/RFC8969, January 2021, <https://www.rfc-editor.org/rfc/rfc8969>.

[RFC9182] Barguil, S., Gonzalez de Dios, O., Ed., Boucadair, M., Ed., Munoz, L., and A. Aguado, "A YANG Network Data Model for Layer 3 VPNs", RFC 9182, DOI 10.17487/RFC9182, February 2022, <https://www.rfc-editor.org/rfc/rfc9182>.

[RFC9408] Boucadair, M., Ed., Gonzalez de Dios, O., Barguil, S., Wu, Q., and V. Lopez, "A YANG Network Data Model for Service Attachment Points (SAPs)", RFC 9408, DOI 10.17487/RFC9408, June 2023, <https://www.rfc-editor.org/rfc/rfc9408>.

Acknowledgments

TBC.

Contributors

Victor Lopez
Nokia

Email: victor.lopez@nokia.com

Ivan Bykov
Ribbon Communications

Email: Ivan.Bykov@rbbn.com

Qin Wu
Huawei

Email: bill.wu@huawei.com

Kenichi Ogaki
KDDI

Email: ke-oogaki@kddi.com

Luis Angel Munoz
Vodafone

Email: luis-angel.munoz@vodafone.com

Authors' Addresses

Mohamed Boucadair (editor)
Orange

Email: mohamed.boucadair@orange.com

Richard Roberts (editor)
Juniper

Email: rroberts@juniper.net

Oscar Gonzalez de Dios
Telefonica

Email: oscar.gonzalezdedios@telefonica.com

Samier Barguil Giraldo
Nokia

Email: samier.barguil_giraldo@nokia.com

Bo Wu
Huawei Technologies

Email: lana.wubo@huawei.com