Stream: Internet Engineering Task Force (IETF)
RFC: 9833
Category: Standards Track
Published: August 2025
ISSN: 2070-1721
Authors:
M. Boucadair, Ed. *Orange*
R. Roberts, Ed. *Juniper*
O. Gonzalez de Dios *Telefonica*
S. Barguil Giraldo *Nokia*
B. Wu *Huawei Technologies*

RFC 9833
A Common YANG Data Model for Attachment Circuits

Abstract

The document specifies a common attachment circuits (ACs) YANG data model, which is designed to be reusable by other models. This design is meant to ensure consistent AC structures among models that manipulate ACs. 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

Copyright Notice

Table of Contents

1. Introduction

Connectivity services are provided by networks to customers via dedicated terminating points (e.g., Service Functions (SFs), Customer Premises Equipment (CPE), Autonomous System Border Routers (ASBRs), data center gateways, or Internet Exchange Points (IXPs)). A connectivity service ensures data transfer from (or destined to) a given terminating point to (or originating from) other terminating points. Objectives for such a connectivity service may be negotiated and agreed upon between a customer and 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 connecting the customer's terminating points to the provider network (typically, a Provider Edge (PE)), thereby enabling successful data exchange. This necessary provisioning is referred to in this document as an "attachment circuit" (AC), while the underlying link is referred to as the "bearer".

When a customer requests a new service, that service can be associated with existing attachment circuits or may require the instantiation of new attachment circuits. Whether these attachment circuits are dedicated to a particular service or shared among multiple services depends on the specific deployment.

Examples of attachment circuits are depicted in Figure 1. A Customer Edge (CE) may be realized as a physical node or a logical entity. From the network's perspective, a CE is treated as a peer Service Attachment Point (SAP) [RFC9408]. CEs can be dedicated to a single service (e.g., Layer 3 Virtual Private Network (VPN) or Layer 2 VPN) or can host multiple services (e.g., Service Functions [RFC7665]). A single AC, as viewed by the network provider, may be bound to one or more peer SAPs (e.g., "CE1" and "CE2"). For instance, as discussed in [RFC4364], multiple CEs can attach to a PE over the same attachment circuit. This approach is typically deployed when the Layer 2 infrastructure between the CE and the network supports a multipoint service. A single CE may also terminate multiple ACs (e.g., "CE3" and "CE4"), which may be carried over the same or distinct bearers.

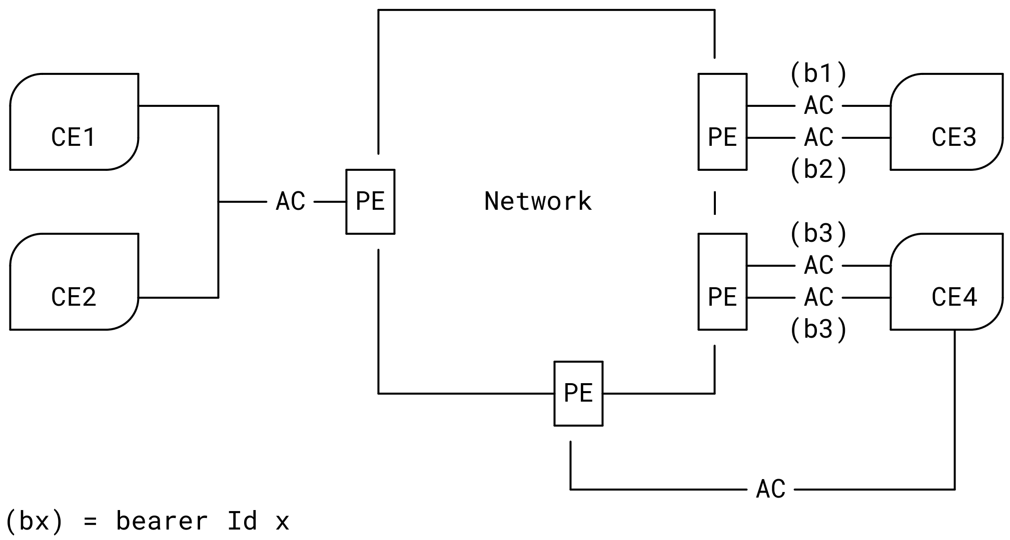

Figure 1: Examples of ACs

This document specifies a common module ("ietf-ac-common") for attachment circuits (Section 5). The module is designed to be reusable by other models, thereby ensuring consistent AC structures among modules that manipulate ACs. For example, the common module can be reused by service models to expose AC-as-a-Service (ACaaS) (e.g., [RFC9834]) or by service models that require binding a service to a set of ACs (e.g., Network Slice Service [YANG-NSS])). It can also be used by network models to provision ACs (e.g., [RFC9835]) and device models, among others.

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

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

2. Conventions and Definitions

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

LxSM refers to both the Layer 2 Service Model (L2SM) [RFC8466] and the Layer 3 Service Model (L3SM) [RFC8299].

LxNM refers to both the Layer 2 Network Model (L2NM) [RFC9291] and the Layer 3 Network Model (L3NM) [RFC9182].

This document uses the following term:

Bearer: A physical or logical link that connects a CE (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 then 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 Virtual Local Area Networks (VLANs) on the same bearer that is provided by a physical link).

The names of data nodes are prefixed using the prefix associated with the corresponding imported YANG module as shown in Table 1.

Prefix	Module	Reference
inet	ietf-inet-types	Section 4 of [RFC6991]
key-chain	ietf-key-chain	[RFC8177]

Prefix	Module	Reference
nacm	ietf-netconf-acm	[RFC8341]
vpn-common	ietf-vpn-common	[RFC9181]
yang	ietf-yang-types	Section 3 of [RFC6991]

Table 1: Modules and Their Associated Prefixes

3. Relationship to Other AC Data Models

Figure 2 depicts the relationship between the various AC data models:

- "ietf-ac-common" (Section 5)
- "ietf-bearer-svc" (Section 6.1 of [RFC9834])
- "ietf-ac-svc" (Section 6.2 of [RFC9834])
- "ietf-ac-ntw" [RFC9835]
- "ietf-ac-glue" [RFC9836]

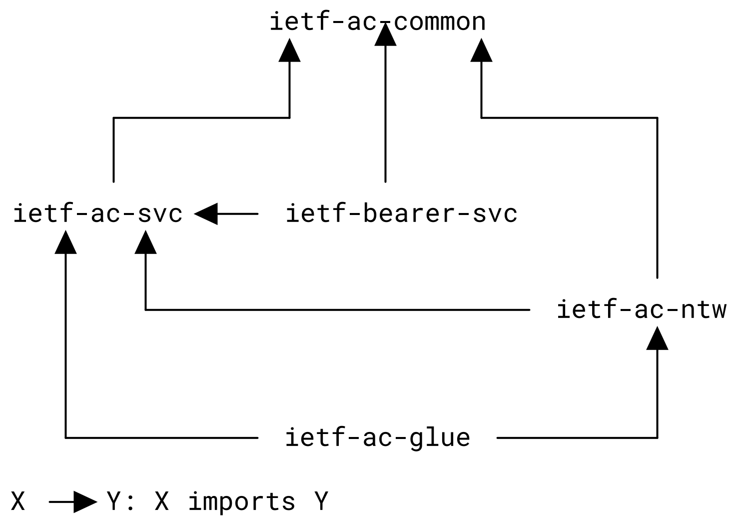

Figure 2: AC Data Models

The "ietf-ac-common" module is imported by the "ietf-bearer-svc", "ietf-ac-svc", and "ietf-ac-ntw" modules. Bearers managed using the "ietf-bearer-svc" module may be referenced by service ACs managed using the "ietf-ac-svc" module. Similarly, a bearer managed using the "ietf-bearer-svc" module may list the set of ACs that use that bearer. To facilitate correlation between an AC service request and the actual AC provisioned in the network, "ietf-ac-ntw" leverages the AC references exposed by the "ietf-ac-svc" module. Furthermore, to bind Layer 2 VPN or Layer 3 VPN services with ACs, the "ietf-ac-glue" module augments the LxSM and LxNM with AC service references exposed by the "ietf-ac-svc" module and AC network references exposed by the "ietf-ac-ntw" module.

4. Description of the AC Common YANG Module

The full tree diagram of the module is provided in Appendix A. Subtrees are provided in the following subsections for the reader's convenience.

4.1. Features

The module defines the following features:

'layer2-ac': Used to indicate support of ACs with Layer 2 properties.

'layer3-ac': Used to indicate support of ACs with Layer 3 properties.

'server-assigned-reference': Used to indicate support of server-generated references to access relevant resources. For example, a server can be a network controller or a router in a provider network.

For example, a bearer request is first created using a name that is assigned by the client, but if this feature is supported, the request will also include a server-generated reference. That reference can be used when requesting the creation of an AC over the existing bearer.

4.2. 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 is 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': Used 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).

'l3-tunnel-type': Used to control the Layer 3 tunnel selection for an AC. Examples of such type are: IP-in-IP [RFC2003], IPsec [RFC4301], and Generic Routing Encapsulation (GRE) [RFC1701] [RFC1702][RFC7676].

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

'role': Used to indicate the type of an AC: User-to-Network Interface (UNI), Network-to-Network Interface (NNI), or public NNI.

The reader may refer to [MEF6], [MEF17], [RFC6004], or [RFC6215] for examples of discussions regarding the use of UNI and NNI reference points.

New administrative status types: In addition to the status types already defined in [RFC9181], this document defines:

- 'awaiting-validation' to report that a request is pending an administrator approval.
- 'awaiting-processing' to report that a request was approved and validated but is awaiting more processing before activation.
- 'admin-prohibited' to report that a request cannot be handled because of administrative policies.
- 'rejected' to report that a request was rejected due to reasons not covered by the other status types.

'bgp-role': Used to indicate the BGP role when establishing a BGP session per [RFC9234].

4.3. Reusable Groupings

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

'service-status' (Figure 3): Controls the administrative service status and reports the operational service status.

'ac-profile-cfg' (Figure 3): A grouping with a set of valid provider profile identifiers. The following profiles are supported:

'encryption-profile-identifier': Refers to a set of policies related to the encryption setup that can be applied when provisioning an AC.

'qos-profile-identifier': Refers to a set of policies, such as classification, marking, and actions (e.g., [RFC3644]).

'failure-detection-profile-identifier': Refers to a set of failure detection policies (e.g., Bidirectional Forwarding Detection (BFD) policies [RFC5880]) that can be invoked when building an AC.

'forwarding-profile-identifier': Refers to the policies that apply to the forwarding of packets conveyed within an AC. Such policies may consist, for example, of applying Access Control Lists (ACLs).

'routing-profile-identifier': Refers to a set of routing policies that will be invoked (e.g., BGP policies) when building an AC.

'op-instructions' (Figure 3): Defines a set of parameters to specify basic scheduling instructions and report related events for a service request (e.g., AC or bearer) ('service-status'). Advanced scheduling groupings are defined in [YANG-SCHEDULE].

```
  grouping service-status:
    +-- status
       +-- admin-status
       |  +-- status?        identityref
       |  +--ro last-change?   yang:date-and-time
       +--ro oper-status
          +--ro status?        identityref
          +--ro last-change?   yang:date-and-time
  grouping ac-profile-cfg:
    +-- valid-provider-identifiers
       +-- encryption-profile-identifier* [id]
       |  +-- id   string
       +-- qos-profile-identifier* [id]
       |  +-- id   string
       +-- failure-detection-profile-identifier* [id]
       |  +-- id   string
       +-- forwarding-profile-identifier* [id]
       |  +-- id   string
       +-- routing-profile-identifier* [id]
          +-- id   string
  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 3: Service Status, Profiles, and Operational Instructions Groupings

Layer 2 encapsulations (Figure 4): Groupings for the following encapsulation schemes are supported: dot1Q, QinQ, and priority-tagged.

Layer 2 tunnel services (Figure 4): These groupings are used to define Layer 2 tunnel services that may be needed for the activation of an AC. Examples of supported Layer 2 services 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 4: Layer 2 Connection Groupings

Layer 3 address allocation (Figure 5): Defines both IPv4 and IPv6 groupings to specify IP address allocation over an AC. Both dynamic and static address schemes are supported.

For both IPv4 and IPv6, 'address-allocation-type' is used to indicate the IP address allocation mode to activate. When 'address-allocation-type' is set to 'provider-dhcp', DHCP assignments can be made locally or by an external DHCP server. Such behavior is controlled by setting 'dhcp-service-type'.

Note that if 'address-allocation-type' is set to 'slaac', the Prefix Information option of Router Advertisements that will be issued for SLAAC purposes will carry the IPv6 prefix that is determined by 'local-address' and 'prefix-length'.

IP connections (Figure 5): Defines IPv4 and IPv6 groupings 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 5: Layer 3 Connection Groupings

Routing parameters & Operations, Administration, and Maintenance (OAM) (Figure 6): In addition to static routing, the module supports the following routing protocols: BGP [RFC4271], OSPF [RFC4577] [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. The groupings support local specification of authentication keys and the associated authentication algorithm to accommodate legacy implementations that do not support key chains [RFC8177].

Note that this version of the common AC model covers authentication options that are common to both OSPFv2 [RFC4577] and OSPFv3 [RFC6565]; as such, the model does not support [RFC4552].

Similar to [RFC9182], this version of the common AC model assumes that parameters specific to the TCP Authentication Option (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 ('bgp-peer-group-without-name' and 'bgp-peer-group-with-name'): Includes a set of parameters to identify a BGP peer group. Such a group can be defined by providing a local Autonomous System 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 ('bgp-peer-group-with-name').

Basic OSPF and IS-IS parameters ('ospf-basic' and 'isis-basic'): These groupings include the minimal set of routing configuration that is required for the activation of OSPF and IS-IS.

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

The 'redundancy-group' grouping lists the groups to which an AC belongs [RFC9181]. For example, the 'group-id' is used to associate redundancy or protection constraints of ACs.

```
 grouping bgp-authentication:
   +-- authentication
      +-- enabled?          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
      +-- enabled?          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
      +-- enabled?          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
      +-- enabled?          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:
   +-- local-as?         inet:as-number
   +-- peer-as?          inet:as-number
   +-- address-family?   identityref
   +-- role?             identityref
 grouping bgp-peer-group-with-name:
   +-- name?             string
   +-- local-as?         inet:as-number
   +-- peer-as?          inet:as-number
   +-- address-family?   identityref
   +-- role?             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
          |  +--ro 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
          |  +--ro last-change?   yang:date-and-time
          +--ro oper-status
             +--ro status?        identityref
             +--ro last-change?   yang:date-and-time
  grouping bfd:
    +-- holdtime?   uint32
  grouping redundancy-group:
    +-- group* [group-id]
       +-- group-id?     string
       +-- precedence?   identityref
```

Figure 6: Routing & OAM Groupings

Bandwidth parameters (Figure 7): 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. Type values are taken from [RFC9181]. For example, the following values can be used:

'bw-per-cos': The bandwidth is per Class of Service (CoS).

'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 7: Bandwidth Groupings

5. Common Attachment Circuit YANG Module

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

```
<CODE BEGINS> file "ietf-ac-common@2025-08-11.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 common attachment circuit (AC)
     YANG module with a set of reusable features, types,
     identities, and groupings.

     Copyright (c) 2025 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 9833; see the
     RFC itself for full legal notices.";

  revision 2025-08-11 {
```

```
    description
      "Initial revision.";
    reference
      "RFC 9833: A Common YANG Data Model for Attachment Circuits";
  }

  /****************************Features************************/

  feature layer2-ac {
    description
      "Indicates support of Layer 2 ACs.";
  }

  feature layer3-ac {
    description
      "Indicates support of Layer 3 ACs.";
  }

  feature server-assigned-reference {
    description
      "Indicates support for server-generated references and use
       of such references to access related resources.";
  }

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

  identity address-allocation-type {
    description
      "Base identity for address allocation type on 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
         infrastructure.";
    }

    // 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.";
   }

   // Layer 3 tunnel types

   identity l3-tunnel-type {
     description
       "Base identity for Layer 3 tunnel selection for an AC.";
   }

   identity ip-in-ip {
     base l3-tunnel-type;
     description
       "IP-in-IP tunneling.";
     reference
       "RFC 2003: IP Encapsulation within IP";
   }

   identity ipsec {
     base l3-tunnel-type;
     description
       "IP Security (IPsec).";
     reference
       "RFC 4301: Security Architecture for the Internet
                  Protocol";
   }

   identity gre {
     base l3-tunnel-type;
     description
       "Generic Routing Encapsulation (GRE).";
     reference
       "RFC 1701: Generic Routing Encapsulation (GRE)
        RFC 1702: Generic Routing Encapsulation over IPv4 networks
        RFC 7676: IPv6 Support for Generic Routing Encapsulation
                  (GRE)";
   }

   // Tagging precedence

   identity precedence-type {
     description
       "Redundancy type.  Attachment to a network 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 a secondary attachment circuit.";
    }

    // AC type

    identity role {
      description
        "Base identity for the network role of an AC.";
    }

    identity uni {
      base role;
      description
        "User-to-Network Interface (UNI).";
    }

    identity nni {
      base role;
      description
        "Network-to-Network Interface (NNI).";
    }

    identity public-nni {
      base role;
      description
        "Public peering.  This is typically set using a shared
         network, such as an Internet Exchange Point (IXP).";
    }

    // More Admin status types

    identity awaiting-validation {
      base vpn-common:administrative-status;
      description
        "This administrative status reflects that a request is
         pending an administrator approval.";
    }

    identity awaiting-processing {
      base vpn-common:administrative-status;
      description
        "This administrative status reflects that a request was
         approved and validated but is awaiting more processing
         before activation.";
    }

    identity admin-prohibited {
      base vpn-common:administrative-status;
      description
        "This administrative status reflects that a request cannot
         be handled because of administrative policies.";
```

```
  }

  identity rejected {
    base vpn-common:administrative-status;
    description
      "This administrative status reflects that a request was
       rejected because, e.g., there are no sufficient resources
       or other reasons not covered by the other status types.";
  }

  // BGP role

  identity bgp-role {
    description
      "Used to indicate the BGP role when establishing a BGP
       session.";
    reference
      "RFC 9234: Route Leak Prevention and Detection Using
                 Roles in UPDATE and OPEN Messages, Section 4";
  }

  identity provider {
    base bgp-role;
    description
      "The local AS is a transit provider of the remote AS.";
  }

  identity client {
    base bgp-role;
    description
      "The local AS is a transit customer of the remote AS.";
  }

  identity rs {
    base bgp-role;
    description
      "The local AS is a Route Server (RS).";
  }

  identity rs-client {
    base bgp-role;
    description
      "The local AS is a client of an RS, and the RS is the
       remote AS.";
  }

  identity peer {
    base bgp-role;
    description
      "The local and remote ASes have a peering relationship.";
  }

  /***************************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********************/
    /**** Service Status ****/

    grouping service-status {
      description
        "Service status grouping.";
      container status {
        description
          "Service status.";
        container admin-status {
          description
            "Administrative service status.";
          leaf status {
            type identityref {
              base vpn-common:administrative-status;
            }
            description
              "Administrative service status.";
          }
          leaf last-change {
            type yang:date-and-time;
            config false;
            description
              "Indicates the actual date and time of the service status
               change.";
          }
        }
        container oper-status {
          config false;
          description
            "Operational service status.";
          uses vpn-common:oper-status-timestamp;
        }
      }
    }

    /**** 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 failure-detection-profile-identifier {
          key "id";
          description
            "List of BFD profile identifiers.";
          leaf id {
            type string;
            description
              "Identification of the failure detection (e.g., 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
        "Indicates the actual date and time when the service
         actually was enabled.";
    }
    leaf actual-stop {
      type yang:date-and-time;
      config false;
      description
        "Indicates 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 {
          range "1..4094";
        }
        description
          "Service VLAN (S-VLAN) identifier.";
      }
      leaf cvlan-id {
        type uint16 {
          range "1..4094";
        }
        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;
      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, 'ac-common: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, 'ac-common: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, 'ac-common: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(), 'ac-common:slaac') "
         + "or derived-from-or-self(current(), "
         + "'ac-common:provider-dhcp-slaac'))" {
        error-message "SLAAC is only applicable to IPv6.";
      }
      description
        "Defines how IPv4 addresses are allocated to the peer
```

```
         termination points.";
    }
  }

  // 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
         termination points.";
    }
  }

  // Basic parameters for an IPv4 connection

  grouping ipv4-connection-basic {
    description
      "Basic set for 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, 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
               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 an IPv6 connection

  grouping ipv6-connection-basic {
    description
      "Basic set for 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, 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
               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 connection 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 address 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 {
          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 dynamically
```

```
                   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 of 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 connection parameters.";
    leaf local-address {
      type inet:ipv6-address;
      description
        "IPv6 address of the provider side.";
    }
    leaf virtual-address {
      type inet:ipv6-address;
      description
        "This address 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 {
          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 dynamically
                   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 by the customer.";
        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 enabled {
          type boolean;
          description
            "Enables or disables authentication.";
        }
        container keying-material {
          when "../enabled = '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
                  "Specifies a reference to the MD5 key chain.";
                reference
                  "RFC 8177: YANG Data Model for Key Chains";
              }
            }
            case explicit {
              leaf key-id {
                type uint32;
                description
                  "Specifies a 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 enabled {
        type boolean;
        description
          "Enables or disables authentication.";
      }
      container keying-material {
        when "../enabled = 'true'";
        description
          "Container for describing how an OSPF session is to be
           secured for an AC.";
        choice option {
          description
            "Options for OSPF authentication.";
          case auth-key-chain {
            leaf key-chain {
              type key-chain:key-chain-ref;
              description
                "Specifies the name of the key chain.";
            }
          }
          case auth-key-explicit {
            leaf key-id {
              type uint32;
              description
                "Specifies a 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 enabled {
          type boolean;
          description
            "Enables or disables authentication.";
        }
        container keying-material {
          when "../enabled = 'true'";
          description
            "Describes 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
                  "Specifies the name of the key chain.";
              }
            }
            case auth-key-explicit {
              leaf key-id {
                type uint32;
                description
                  "Indicates a 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
        "Includes RIP authentication parameters.";
      leaf enabled {
        type boolean;
        description
          "Enables or disables authentication.";
      }
      container keying-material {
        when "../enabled = 'true'";
        description
          "Describes how a RIP session is to be secured
           on an AC.";
        choice option {
          description
            "Specifies the authentication scheme.";
          case auth-key-chain {
            leaf key-chain {
              type key-chain:key-chain-ref;
              description
                "Indicates the name of the key chain.";
            }
          }
          case auth-key-explicit {
            leaf key {
              type string;
              description
                "Specifies a 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;
      description
        "Indicates a local Autonomous System 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.";
    }
    leaf role {
      type identityref {
        base ac-common:bgp-role;
      }
      description
        "Specifies the BGP role (provider, customer, peer, etc.).";
      reference
        "RFC 9234: Route Leak Prevention and Detection Using
                   Roles in UPDATE and OPEN Messages, Section 4";
    }
  }

  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
        "Specifies the name of the BGP peer-group.";
    }
    uses bgp-peer-group-without-name;
  }

  grouping ospf-basic {
    description
      "Includes 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
        "Specifies an 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;
      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
        "Specifies an area address.";
    }
  }

  // Static routing

  grouping ipv4-static-rtg-entry {
    description
      "Parameters to configure a specific IPv4 static routing
       entry.";
    leaf lan {
      type inet:ipv4-prefix;
      description
        "Indicates an IPv4 LAN prefix.";
    }
    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
        "A set of parameters 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 ac-common:service-status;
      }
    }

    grouping ipv6-static-rtg-entry {
      description
        "Parameters to configure a specific IPv6 static routing
         entry.";
      leaf lan {
        type inet:ipv6-prefix;
        description
          "Indicates an IPv6 LAN prefix.";
      }
      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
      "A set of parameters 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 customer-terminating points.";
      uses ipv6-static-rtg-entry;
      uses ac-common:service-status;
    }
  }

  // OAM

  grouping bfd {
    description
      "Groups a set of basic BFD parameters.";
    leaf holdtime {
      type uint32;
      units "milliseconds";
      description
        "Specifies the 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";
    }
  }

  // redundancy

  grouping redundancy-group {
    description
      "A grouping for redundancy group.";
    list group {
      key "group-id";
      description
        "Specifies a list of group identifiers.";
      leaf group-id {
        type string;
        description
          "Indicates the group-id to which an AC belongs.";
      }
      leaf precedence {
        type identityref {
          base ac-common:precedence-type;
        }
        description
          "Defines redundancy of an AC.";
      }
    }
```

```
  }

  // 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 the sum of the 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 parameters.";
        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 Class of Service (CoS).";
            list cos {
              key "cos-id";
              description
                "List of CoSes.";
              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>
```

6. Security Considerations

This section is modeled after the template described in Section 3.7 of [YANG-GUIDELINES].

The "ietf-ac-common" YANG module defines a data model that is designed to be accessed via YANG-based management protocols, such as NETCONF [RFC6241] and RESTCONF [RFC8040]. These protocols have to use a secure transport layer (e.g., SSH [RFC4252], TLS [RFC8446], and QUIC [RFC9000]) and have to use mutual authentication.

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 YANG 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. As such, there are no additional security issues related to the YANG module that need to be considered.

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]).

7. IANA Considerations

IANA has registered 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 has registered the following YANG module in the "YANG Module Names" subregistry [RFC6020] within the "YANG Parameters" registry:

Name: ietf-ac-common
Maintained by IANA? N
Namespace: urn:ietf:params:xml:ns:yang:ietf-ac-common
Prefix: ac-common
Reference: RFC 9833

8. References

8.1. Normative References

[ISO10589]

ISO/IEC, "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)", ISO/IEC 10589:2002, November 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/info/rfc1195>.

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

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/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/info/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/info/rfc4577>.

[RFC5308] Hopps, C., "Routing IPv6 with IS-IS", RFC 5308, DOI 10.17487/RFC5308, October 2008, <https://www.rfc-editor.org/info/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/info/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/info/rfc6020>.

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

[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/info/rfc6565>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/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/info/rfc7348>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/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/info/rfc8077>.

[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/info/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/info/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/info/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/info/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/info/rfc9181>.

8.2. Informative References

[MEF17] The Metro Ethernet Forum, "Service OAM Requirements & Framework - Phase 1", MEF Technical Specification, MEF 17, April 2007, <https://www.mef.net/wp-content/uploads/2015/04/MEF-17.pdf>.

[MEF6] The Metro Ethernet Forum, "Ethernet Services Definitions - Phase I", MEF Technical Specification, MEF 6, August 2004, <https://www.mef.net/Assets/Technical_Specifications/PDF/MEF_6.pdf>.

[RFC1701] Hanks, S., Li, T., Farinacci, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 1701, DOI 10.17487/RFC1701, October 1994, <https://www.rfc-editor.org/info/rfc1701>.

[RFC1702] Hanks, S., Li, T., Farinacci, D., and P. Traina, "Generic Routing Encapsulation over IPv4 networks", RFC 1702, DOI 10.17487/RFC1702, October 1994, <https://www.rfc-editor.org/info/rfc1702>.

[RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, DOI 10.17487/RFC2003, October 1996, <https://www.rfc-editor.org/info/rfc2003>.

[RFC3644] Snir, Y., Ramberg, Y., Strassner, J., Cohen, R., and B. Moore, "Policy Quality of Service (QoS) Information Model", RFC 3644, DOI 10.17487/RFC3644, November 2003, <https://www.rfc-editor.org/info/rfc3644>.

[RFC4252] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, DOI 10.17487/RFC4252, January 2006, <https://www.rfc-editor.org/info/rfc4252>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[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/info/rfc4364>.

[RFC4552] Gupta, M. and N. Melam, "Authentication/Confidentiality for OSPFv3", RFC 4552, DOI 10.17487/RFC4552, June 2006, <https://www.rfc-editor.org/info/rfc4552>.

[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/info/rfc4862>.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <https://www.rfc-editor.org/info/rfc5880>.

[RFC6004] Berger, L. and D. Fedyk, "Generalized MPLS (GMPLS) Support for Metro Ethernet Forum and G.8011 Ethernet Service Switching", RFC 6004, DOI 10.17487/RFC6004, October 2010, <https://www.rfc-editor.org/info/rfc6004>.

[RFC6215] Bocci, M., Levrau, L., and D. Frost, "MPLS Transport Profile User-to-Network and Network-to-Network Interfaces", RFC 6215, DOI 10.17487/RFC6215, April 2011, <https://www.rfc-editor.org/info/rfc6215>.

[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/info/rfc7665>.

[RFC7676] Pignataro, C., Bonica, R., and S. Krishnan, "IPv6 Support for Generic Routing Encapsulation (GRE)", RFC 7676, DOI 10.17487/RFC7676, October 2015, <https://www.rfc-editor.org/info/rfc7676>.

[RFC8299] Wu, Q., Ed., Litkowski, S., Tomotaki, L., and K. Ogaki, "YANG Data Model for L3VPN Service Delivery", RFC 8299, DOI 10.17487/RFC8299, January 2018, <https://www.rfc-editor.org/info/rfc8299>.

[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/info/rfc8340>.

[RFC8466] Wen, B., Fioccola, G., Ed., Xie, C., and L. Jalil, "A YANG Data Model for Layer 2 Virtual Private Network (L2VPN) Service Delivery", RFC 8466, DOI 10.17487/RFC8466, October 2018, <https://www.rfc-editor.org/info/rfc8466>.

[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/info/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/info/rfc8969>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[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/info/rfc9182>.

[RFC9234] Azimov, A., Bogomazov, E., Bush, R., Patel, K., and K. Sriram, "Route Leak Prevention and Detection Using Roles in UPDATE and OPEN Messages", RFC 9234, DOI 10.17487/RFC9234, May 2022, <https://www.rfc-editor.org/info/rfc9234>.

[RFC9291] Boucadair, M., Ed., Gonzalez de Dios, O., Ed., Barguil, S., and L. Munoz, "A YANG Network Data Model for Layer 2 VPNs", RFC 9291, DOI 10.17487/RFC9291, September 2022, <https://www.rfc-editor.org/info/rfc9291>.

[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/info/rfc9408>.

[RFC9834] Boucadair, M., Ed., Roberts, R., Ed., Gonzalez de Dios, O., Barguil Giraldo, S., and B. Wu, "YANG Data Models for Bearers and 'Attachment Circuits'-as-a-Service (ACaaS)", RFC 9834, August 2025, <https://www.rfc-editor.org/info/rfc9834>.

[RFC9835] Boucadair, M., Roberts, R., Gonzalez de Dios, O., Barguil Giraldo, S., and B. Wu, "A Network YANG Data Model for Attachment Circuits", RFC 9835, August 2025, <https://www.rfc-editor.org/info/rfc9835>.

[RFC9836] Boucadair, M., Ed., Roberts, R., Barguil Giraldo, S., and O. Gonzalez de Dios, "A YANG Data Model for Augmenting VPN Service and Network Models with Attachment Circuits", RFC 9836, August 2025, <https://www.rfc-editor.org/info/rfc9836>.

[YANG-GUIDELINES] Bierman, A., Boucadair, M., Ed., and Q. Wu, "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-netmod-rfc8407bis-22, 14 January 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-rfc8407bis-22>.

[YANG-NSS] Wu, B., Dhody, D., Rokui, R., Saad, T., and J. Mullooly, "A YANG Data Model for the RFC 9543 Network Slice Service", Work in Progress, Internet-Draft, draft-ietf-teas-ietf-network-slice-nbi-yang-25, 9 May 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-teas-ietf-network-slice-nbi-yang-25>.

[YANG-SCHEDULE] Ma, Q., Ed., Wu, Q., Boucadair, M., Ed., and D. King, "A Common YANG Data Model for Scheduling", Work in Progress, Internet-Draft, draft-ietf-netmod-schedule-yang-04, 7 February 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-schedule-yang-04>.

Appendix A. Full Tree

```
module: ietf-ac-common

  grouping service-status:
    +-- status
       +-- admin-status
       |  +-- status?        identityref
       |  +--ro last-change?   yang:date-and-time
       +--ro oper-status
          +--ro status?        identityref
          +--ro last-change?   yang:date-and-time
  grouping ac-profile-cfg:
    +-- valid-provider-identifiers
       +-- encryption-profile-identifier* [id]
       |  +-- id   string
       +-- qos-profile-identifier* [id]
       |  +-- id   string
       +-- failure-detection-profile-identifier* [id]
       |  +-- id   string
       +-- forwarding-profile-identifier* [id]
       |  +-- id   string
       +-- routing-profile-identifier* [id]
          +-- id   string
  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
  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
  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
grouping bgp-authentication:
  +-- authentication
     +-- enabled?           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
```

```
       +-- enabled?           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
       +-- enabled?           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
       +-- enabled?           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:
    +-- local-as?         inet:as-number
    +-- peer-as?          inet:as-number
    +-- address-family?   identityref
    +-- role?             identityref
  grouping bgp-peer-group-with-name:
    +-- name?             string
    +-- local-as?         inet:as-number
    +-- peer-as?          inet:as-number
    +-- address-family?   identityref
    +-- role?             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
          |  +--ro 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
         |  +--ro last-change?   yang:date-and-time
         +--ro oper-status
            +--ro status?        identityref
            +--ro last-change?   yang:date-and-time
 grouping bfd:
   +-- holdtime?   uint32
 grouping redundancy-group:
   +-- group* [group-id]
      +-- group-id      string
      +-- precedence?   identityref
 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
```

Acknowledgments

The document reuses many of the structures that were defined in [RFC9181] and [RFC9182].

Thanks to Ebben Aries for the YANG Doctors review, Andy Smith and Gyanh Mishra for the RTGDIR reviews, Watson Ladd for the SECDIR review, and Behcet Sarikaya for the GENART review.

Thanks to Reza Rokui for the shepherd review.

Thanks to Mahesh Jethanandani for the AD review.

Thanks to Éric Vyncke, Gunter Van de Velde, Orie Steele, and Paul Wouters for the IESG review.

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