Workgroup: OPSAWG
Internet-Draft:
draft-boro-opsawg-teas-attachment-circuit-00
Published: 9 January 2023
Intended Status: Standards Track
Expires: 13 July 2023
Authors: M. Boucadair, Ed. R. Roberts O. G. D. Dios
Orange Juniper Telefonica
S. B. Giraldo B. Wu
Nokia Huawei Technologies

A YANG Service Data Model for Attachment Circuits

Abstract

This document specifies a YANG service data model for Attachment Circuits (ACs). The model can be used for the provisioning of ACs prior or during service provisioning (e.g., Network Slice Service).

The model is designed with the intent to be reusable. Whether a service model reuses structures defined in the AC service model or simply include an AC reference is a design choice of these service models. Relying upon the AC model to manage ACs over which a service is delivered has the merit to decorrelate the management of a service vs. upgrade the AC components to reflect recent AC technologies or features.

Each AC is identified with a unique identifier within a domain. The mapping between this AC and a PE that terminates the AC is hidden to the application/customer that makes use of the AC service model. Such an information is internal to the network controller. Thus, the details about the (network node-specific) attachment interfaces are not exposed in this service model.

Discussion Venues

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

Discussion of this document takes place on the Operations and Management Area Working Group Working Group mailing list (opsawg@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/opsawg/.

Source for this draft and an issue tracker can be found at https://github.com/boucadair/attachment-circuit-model.

Status of This Memo

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

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

working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on 13 July 2023.

Copyright Notice

Table of Contents

1. Introduction

This document specifies a YANG service data model for managing attachment circuits (ACs) that are exposed by a network to its customers (e.g., an enterprise site, a network function, a hosting infrastructure, a peer network provider) . The model can be used for the provisioning of ACs prior or during advanced service provisioning (e.g., Network Slice Service).

Also, the model is designed with the intent to be reusable. Whether a service model reuses structures defined in the AC service model or simply includes an AC reference (that was communicated during AC instantiation) is a design choice of these service models. Relying upon the AC service model to manage ACs over which services are delivered has the merit to decorrelate the management of a service vs. upgrade the AC components to reflect recent AC technologies or new features (e.g., new encryption scheme, additional routing protocol).

Each AC is identified with a unique identifier within a domain. From a network provider standpoint, an AC can be bound to a single or multiple SAPs [I-D.ietf-opsawg-sap]. Likewise, a SAP can be bound to one or multiple ACs. However, the mapping between this AC and a local PE that terminates the AC is hidden to the application that makes use of the AC service model. This information is internal to the Network controller. As such, the details about the (node-specific) attachment interfaces are not exposed in this service model.

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

2. Conventions and Definitions

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

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

This document uses the following terms:

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

Service orchestrator: Refers to a functional entity that interacts with the customer of a network service. The service orchestrator is typically responsible for the attachment circuits, the Provider Edge (PE) selection, and requesting the activation of the requested service to a network controller.

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

Service provider: A service provider that offers network services (e.g., Network Slice Services).

3. Sample Uses of the Attachment Circuit Data Models

Figure 1 depictes two target topology flavors that may host ACs. A CE may be a physical node or a logical entity. The same AC request may include one or multiple ACs that may belong to one or both of these flavors. For the sake of simplfying the illustration, only a subset of these ACs is shown in Figure 1.

CEs may be dedicated to one single service or host multiple services (e.g., service functions [RFC7665]). A single AC (as seen by a network provider) may be bound to one or multiple peer SAPs.

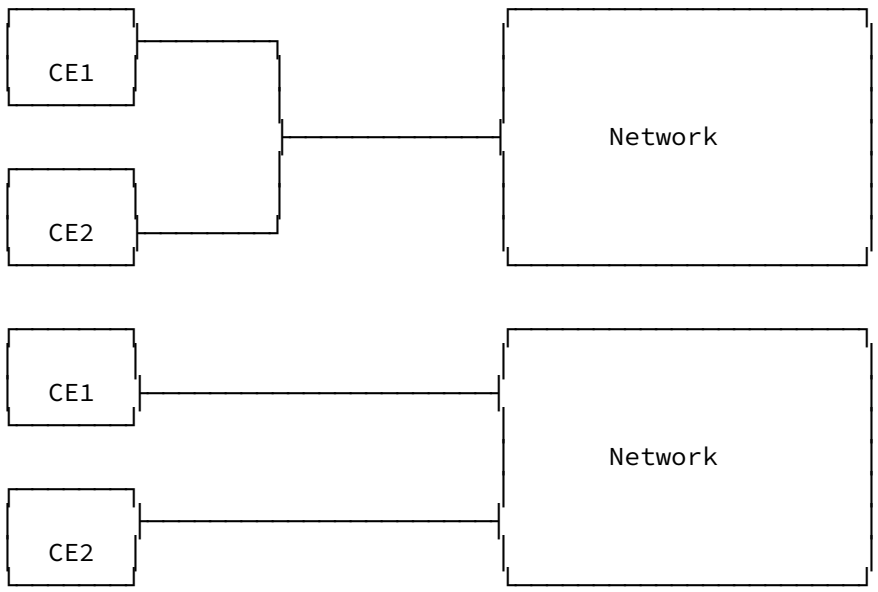

Figure 1: Examples of ACs

3.1. Separate AC Provisioning vs. Actual Service Provisioning

The procedure to provision a service in a service provider network may depend on the practices adopted by a service provider, including the flow put in place for the provisioning of advanced network services and how they are bound to an attachment circuit. For example, the same attachment circuit may be used to host multiple services. In order to avoid service interference and redundant information in various locations, a service provider may expose an interface to manage ACs network-wide. Customers can the request a base attachment circuit to be put in place, and then refer to that base AC when requesting services that are bound to that AC.

Figure 2 shows the positioning of the AC service model is the overall service delivery process.

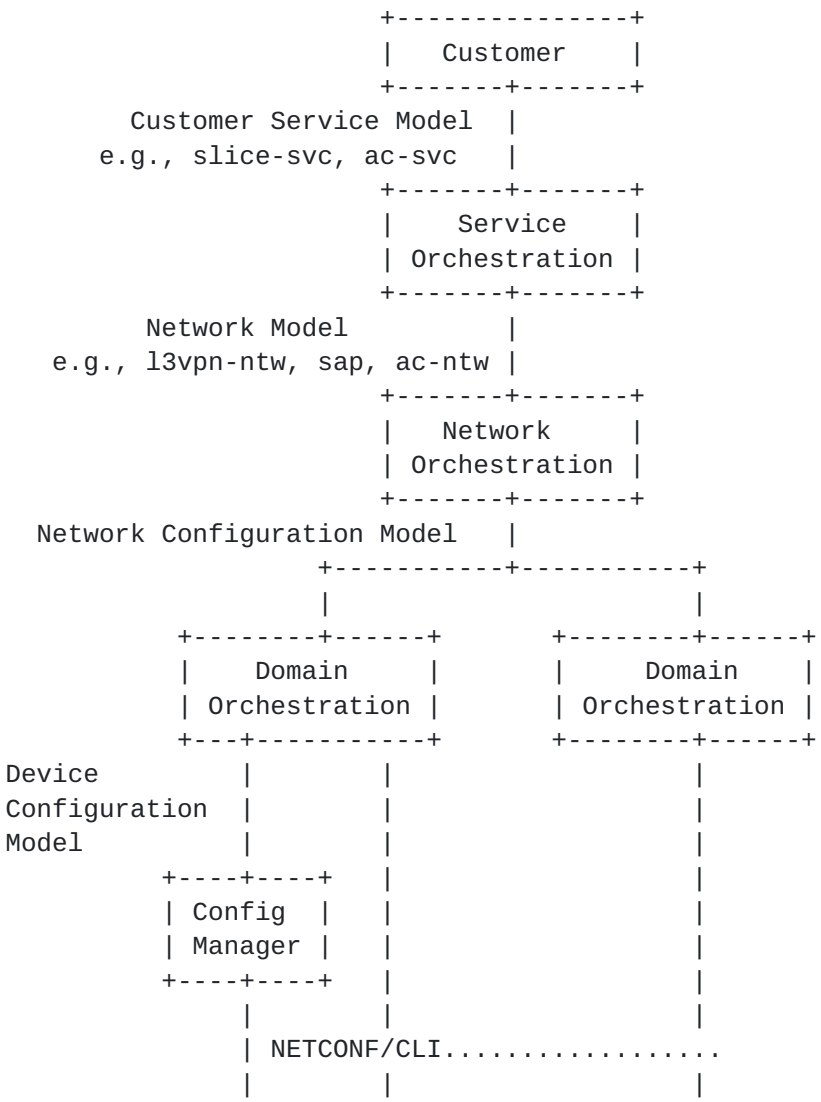

Figure 2: An Example of AC Model Usage

3.2. Examples

This section includes a non-exhaustive list of examples to illustrate the use of the AC service model.

3.3. Request An AC over An Existing Bearer

An example of of a request message body to create a simple AC over an existing bearer is shown in Figure 3. The bearer reference is assumed to be known to both the customer and the network provider. How such reference is shared is out of scope.

```
{
  "ietf-ac-svc:attachment-circuits": {
    "ac": [
      {
        "name": "ac4585",
        "description": "An AC on an existing bearer",
        "requested-ac-start": "2023-12-12T05:00:00.00Z",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "tag-type": "ietf-vpn-common:c-vlan",
              "cvlan-id": 550
            }
          },
          "bearer-reference": "line-156"
        }
      }
    ]
  }
}
```

Figure 3: Example of a Message Body to Request an AC over an Existing Bearer

3.4. Request An AC for a Knwon Peer SAP

An example of a request to create a simple AC, when the peer SAP is known, is shown in Figure 4. In this example, the peer SAP identifier points to an identifier of a service function. The (topological) location of that service function is assumed to be known to the network controller. For example, this can be determined as part of an on-demand procedure to instantiate a service function in a cloud. That instantiated service function can be granted a connectivity service via the provider network.

```
{
  "ietf-ac-svc:attachment-circuits": {
    "ac": [
      {
        "name": "ac4585",
        "description": "An AC on an existing bearer",
        "requested-ac-start": "2023-12-12T05:00:00.00Z",
        "peer-sap-id": [
          "nf-termination-ip"
        ],
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "tag-type": "ietf-vpn-common:c-vlan",
              "cvlan-id": 550
            }
          }
        }
      }
    ]
  }
}
```

Figure 4: Example of a Message Body to Request an AC with a Peer SAP

3.5. One CE, Two ACs

An example of a request to create two ACs to service the same CE on the same link is shown in Figure 5. This example assumes that static addressing is used for both ACs.

```
{
  "ietf-ac-svc:attachment-circuits": {
    "ac": [
      {
        "name": "ac1",
        "description": "a first ac with a same peer node",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "cvlan-id": 1
            }
          }
        },
        "ip-connection": {
          "ipv4": {
            "local-address": "192.0.2.1",
            "prefix-length": 30,
            "address": [
              {
                "address-id": "1",
                "customer-address": "192.0.2.2"
              }
            ]
          },
          "ipv6": {
            "local-address": "2001:db8::1",
            "prefix-length": 64,
            "address": [
              {
                "address-id": "1",
                "customer-address": "2001:db8::2"
              }
            ]
          }
        },
        "routing-protocols": {
          "routing-protocol": [
            {
              "id": "1",
              "type": "ietf-vpn-common:direct-routing"
            }
          ]
        }
      },
      {
        "name": "ac2",
        "description": "a second ac with a same peer node",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "cvlan-id": 2
```

```
            }
          }
        },
        "ip-connection": {
          "ipv4": {
            "local-address": "192.0.2.1",
            "prefix-length": 30,
            "address": [
              {
                "address-id": "1",
                "customer-address": "192.0.2.2"
              }
            ]
          },
          "ipv6": {
            "local-address": "2001:db8::1",
            "prefix-length": 64,
            "address": [
              {
                "address-id": "1",
                "customer-address": "2001:db8::2"
              }
            ]
          }
        },
        "routing-protocols": {
          "routing-protocol": [
            {
              "id": "1",
              "type": "ietf-vpn-common:direct-routing"
            }
          ]
        }
      }
    ]
  }
}
```

Figure 5: Example of a Message Body to Request Two ACes on The Same Link

3.6. Illustrate the Use of Global Profiles

An example of a request to create two ACs to service the same CE on the same link is shown in Figure 6. Unlike Figure 5, this example factorizes some of the redundant data.

```
{
  "ietf-ac-svc:attachment-circuits": {
    "ac-global-profile": [
      {
        "id": "simple-profile",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q"
          }
        },
        "routing-protocols": {
          "routing-protocol": [
            {
              "id": "1",
              "type": "ietf-vpn-common:direct-routing"
            }
          ]
        }
      }
    ],
    "ac": [
      {
        "name": "ac1",
        "description": "a first ac with a same peer node",
        "ac-global-profile": ["simple-profile"],
        "l2-connection": {
          "encapsulation": {
            "dot1q": {
              "cvlan-id": 1
            }
          }
        },
        "ip-connection": {
          "ipv4": {
            "local-address": "192.0.2.1",
            "prefix-length": 30,
            "address": [
              {
                "address-id": "1",
                "customer-address": "192.0.2.2"
              }
            ]
          },
          "ipv6": {
            "local-address": "2001:db8::1",
            "prefix-length": 64,
            "address": [
              {
                "address-id": "1",
                "customer-address": "2001:db8::2"
              }
            ]
          }
```

```
        }
      },
      {
        "name": "ac2",
        "description": "a second ac with a same peer node",
        "ac-global-profile": ["simple-profile"],
        "l2-connection": {
          "encapsulation": {
            "dot1q": {
              "cvlan-id": 2
            }
          }
        },
        "ip-connection": {
          "ipv4": {
            "local-address": "192.0.2.1",
            "prefix-length": 30,
            "address": [
              {
                "address-id": "1",
                "customer-address": "192.0.2.2"
              }
            ]
          },
          "ipv6": {
            "local-address": "2001:db8::1",
            "prefix-length": 64,
            "address": [
              {
                "address-id": "1",
                "customer-address": "2001:db8::2"
              }
            ]
          }
        }
      }
    ]
  }
}
```

Figure 6: Example of a Message Body to Request Two ACes on The Same Link (Global Profile)

3.7. Illustrate the Use of Per-Node Profiles

An example of a request to create two ACs to service the same CE on the same link is shown in Figure 7. Unlike Figure 5, this example factorizes all redundant data.

```
{
  "ietf-ac-svc:attachment-circuits": {
    "ac-node-group": [
      {
        "id": "simple-node-profile",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q"
          }
        },
        "ip-connection": {
          "ipv4": {
            "local-address": "192.0.2.1",
            "prefix-length": 30,
            "address": [
              {
                "address-id": "1",
                "customer-address": "192.0.2.2"
              }
            ]
          },
          "ipv6": {
            "local-address": "2001:db8::1",
            "prefix-length": 64,
            "address": [
              {
                "address-id": "1",
                "customer-address": "2001:db8::2"
              }
            ]
          }
        },
        "routing-protocols": {
          "routing-protocol": [
            {
              "id": "1",
              "type": "ietf-vpn-common:direct-routing"
            }
          ]
        }
      }
    ],
    "ac": [
      {
        "name": "ac1",
        "description": "a first ac with a same peer node",
        "ac-node-profile": ["simple-node-profile"],
        "l2-connection": {
          "encapsulation": {
            "dot1q": {
              "cvlan-id": 1
            }
          }
```

```
        }
      },
      {
        "name": "ac2",
        "description": "a second ac with a same peer node",
        "ac-node-profile": ["simple-node-profile"],
        "l2-connection": {
          "encapsulation": {
            "dot1q": {
              "cvlan-id": 2
            }
          }
        }
      }
    ]
  }
}
```

Figure 7: Example of a Message Body to Request Two ACes on The Same Link (Node Profile)

A customer may request adding a new AC by simply referring to an existing per-node AC profile as shown in Figure 8. This AC inherites all the data that was enclosed in the indicated per-node AC profile (IP addressing, routing, etc.).

```
{
  "ietf-ac-svc:attachment-circuits": {
    "ac": [
      {
        "name": "ac3",
        "description": "a third AC with a same peer node",
        "ac-node-profile": [
          "simple-node-profile"
        ],
        "l2-connection": {
          "encapsulation": {
            "dot1q": {
              "cvlan-id": 3
            }
          },
          "bearer-reference": "line-156"
        }
      }
    ]
  }
}
```

Figure 8: Example of a Message Body to Add a new AC over an existing link (Node Profile)

3.8. Multiple CEs

Figure 9 shows an example of CEs that are interconnected by a service provider network.

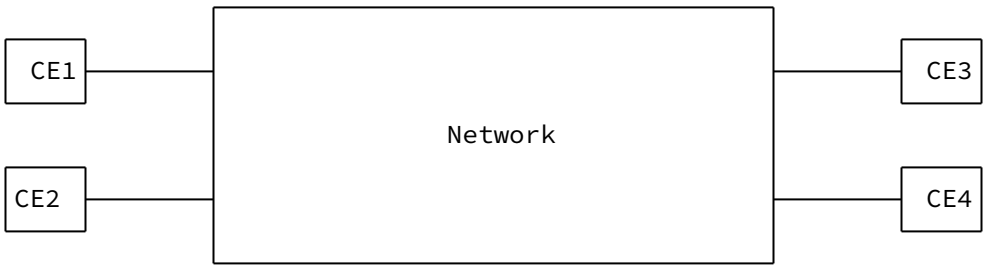

Figure 9: Network Topology Example

Figure 10 depicts an example of the message body of a request to instantiate the various ACs that are shown in Figure 9.

```
{
  "ietf-ac-svc:attachment-circuits": {
    "ac-global-profile": [
      {
        "id": "simple-profile",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "cvlan-id": 1
            }
          }
        }
      }
    ],
    "ac": [
      {
        "name": "ac1",
        "description": "Connect a first site",
        "ac-global-profile": [
          "simple-node-profile"
        ],
        "l2-connection": {
          "bearer-reference": "ce1-network"
        }
      },
      {
        "name": "ac2",
        "description": "Connect a second Site",
        "ac-global-profile": [
          "simple-node-profile"
        ],
        "l2-connection": {
          "bearer-reference": "ce2-network"
        }
      },
      {
        "name": "ac3",
        "description": "Connect a third site",
        "ac-global-profile": [
          "simple-node-profile"
        ],
        "l2-connection": {
          "bearer-reference": "ce3-network"
        }
      },
      {
        "name": "ac4",
        "description": "Connect a forth site",
        "ac-global-profile": [
          "simple-node-profile"
        ],
        "l2-connection": {
```

```
          "bearer-reference": "ce4-network"
        }
      }
    ]
  }
}
```

Figure 10: Example of a Message Body to of Creating Multiple ACs bound to Multiple CEs

4. Description of the Attachment Circuit YANG Module

4.1. Overall Structure of the Module

The overall tree structure of the AC service module is shown in Figure 11.

```
module: ietf-ac-svc
  +--rw specific-provisioning-profiles
  |  ...
  +--rw service-provisioning-profiles
  |  ...
  +--rw attachment-circuits
     +--rw ac-global-profile* [id]
     |  ...
     +--rw ac-node-group* [id]
     |  ...
     +--rw ac* [id]
        +--rw peer-sap-id*         string
        +--rw profile-id*
        |       -> /attachment-circuits/ac-global-profile/id
        +--rw id                   string
        +--rw l2-connection
        |  ...
        +--rw ip-connection
        |  ...
        +--rw routing-protocols
        |  ...
        +--rw oam
        |  ...
        +--rw security
           ...
```

Figure 11: Overall AC Service Tree Structure

The full tree structure is provided in Appendix A.

Each AC is identified with a unique identifier within a domain. The mapping between this AC and a local PE that terminates the AC is hidden to the application that makes use of the AC service model. This information is internal to the Network controller. As such, the details about the (node-specific) attachment interfaces are not exposed in this service model.

4.2. Service Profiles

4.2.1. Description

The 'specific-provisioning-profiles' container (Figure 12) can be used by a service provider to maintain a set of specific profiles

that are similar to those defined in [RFC9181]. The exact definition of the profiles is local to each service provider. The model only includes an identifier for these profiles in order to facilitate identifying and binding local policies when building an AC.

```
module: ietf-ac-svc
  +--rw specific-provisioning-profiles
  |  +--rw valid-provider-identifiers
  |     +--rw external-connectivity-identifier* [id]
  |     |       {external-connectivity}?
  |     |  +--rw id    string
  |     +--rw encryption-profile-identifier* [id]
  |     |  +--rw id    string
  |     +--rw qos-profile-identifier* [id]
  |     |  +--rw id    string
  |     +--rw bfd-profile-identifier* [id]
  |     |  +--rw id    string
  |     +--rw forwarding-profile-identifier* [id]
  |     |  +--rw id    string
  |     +--rw routing-profile-identifier* [id]
  |        +--rw id    string
  +--rw service-provisioning-profiles
  |  +--rw service-profile-identifier* [id]
  |     +--rw id    string
  +--rw attachment-circuits
     +--rw ac-global-profile* [id]
     |  ...
     +--rw ac-node-group* [id]
     |  ...
     +--rw ac* [name]
        ...
```

Figure 12: Service Proviles

As shown in Figure 12, two profile types can be defined: 'specific-provisioning-profiles' and 'service-provisioning-profiles'. Whether only specific profiles, service profiles, or a combination thereff are used is local to each service provider.

The following specific rovisioning profiles can be defined:

'external-connectivity-identifier': Refers to a profile that defines the external connectivity provided to a site that is connected via an AC. External connectivity may be access to the

Internet or restricted connectivity, such as access to a public/ private cloud.

'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]).

'bfd-profile-identifier': Refers to a set of 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.

4.2.2. Referencing Service/Specific Profiles

All these profiles are uniquely identified by the NETCONF/RESTCONF server by an identifier. To ease referencing these profiles by other data models, specific typedefs are defined for each of these profiles. Likewise, an attachment circuit referenc typedef is defiened when referencing a (global) attachment circuit by its name is required. These typedefs **SHOULD** be used when other modules need a reference to one of these profiles or attahment circuits.

4.3. Attachment Circuits Profiles

4.4. Node-Specific Profiles

4.5. Attachment Circuits

The structure of 'attachment-circuits' is shown in Figure 13.

```
module: ietf-ac-svc
  +--rw specific-provisioning-profiles
  |  ...
  +--rw service-provisioning-profiles
  |  ...
  +--rw attachment-circuits
     +--rw ac-global-profile* [id]
     |  ...f
     +--rw ac-node-group* [id]
     |  ...
     +--rw ac* [name]
        +--rw customer-name?        string
        +--rw description?          string
        +--rw requested-ac-start?   yang:date-and-time
        +--rw requested-ac-stop?    yang:date-and-time
        +--ro actual-ac-start?      yang:date-and-time
        +--ro actual-ac-stop?       yang:date-and-time
        +--rw peer-sap-id*          string
        +--rw ac-global-profile*    ac-global-profile-reference
        +--rw ac-node-profile*      ac-node-group-reference
        +--rw name                  string
        +--rw l2-connection
        |  ...
        +--rw ip-connection
        |  ...
        +--rw routing-protocols
        |  ...
        +--rw oam
        |  ...
        +--rw security
           ...
```

Figure 13: Overall Attachment Circuits Tree Structure

4.5.1. Layer 2 Connection Structure

As shown in the tree depicted in Figure 14, the 'l2-connection' container defines service parameters to enable such connectivity at Layer 2.

```
module: ietf-ac-svc
  +--rw specific-provisioning-profiles
  |  ...
  +--rw service-provisioning-profiles
  |  ...
  +--rw attachment-circuits
     +--rw ac-global-profile* [id]
     |  ...f
     +--rw ac-node-group* [id]
     |  ...
     +--rw ac* [name]
        ...
        +--rw name                  string
        +--rw l2-connection
        |  +--rw encapsulation
        |  |  +--rw type?              identityref
        |  |  +--rw dot1q
        |  |  |  +--rw tag-type?   identityref
        |  |  |  +--rw cvlan-id?   uint16
        |  |  +--rw priority-tagged
        |  |  |  +--rw tag-type?   identityref
        |  |  +--rw qinq
        |  |     +--rw tag-type?   identityref
        |  |     +--rw svlan-id    uint16
        |  |     +--rw cvlan-id    uint16
        |  +--rw (l2-service)?
        |  |  +--:(l2-tunnel-service)
        |  |  |  +--rw l2-tunnel-service
        |  |  |     +--rw type?         identityref
        |  |  |     +--rw pseudowire
        |  |  |     |  +--rw vcid?      uint32
        |  |  |     |  +--rw far-end?   union
        |  |  |     +--rw vpls
        |  |  |     |  +--rw vcid?      uint32
        |  |  |     |  +--rw far-end*   union
        |  |  |     +--rw vxlan
        |  |  |        +--rw vni-id             uint32
        |  |  |        +--rw peer-mode?         identityref
        |  |  |        +--rw peer-ip-address*   inet:ip-address
        |  |  +--:(l2vpn)
        |  |     +--rw l2vpn-id?           vpn-common:vpn-id
        |  +--rw bearer-reference?         string {vpn-common:bearer-re
        +--rw ip-connection
        |  ...
        +--rw routing-protocols
        |  ...
        +--rw oam
        |  ...
        +--rw security
           ...
```

Figure 14: Layer 2 Connection Tree Structure

4.5.2. Layer 3 Connection Structure

As shown in the tree depicted in Figure 15, the 'l3-connection' container defines service parameters to enable Layer 3 connectivity for an AC.

```
module: ietf-ac-svc
  +--rw specific-provisioning-profiles
  |  ...
  +--rw service-provisioning-profiles
  |  ...
  +--rw attachment-circuits
     +--rw ac-global-profile* [id]
     |  ...f
     +--rw ac-node-group* [id]
     |  ...
     +--rw ac* [name]
        ...
        +--rw name                  string
        +--rw l2-connection
        |  ...
        +--rw ip-connection
        |  +--rw ipv4 {vpn-common:ipv4}?
        |  |  +--rw local-address?                          inet:ipv4-a
        |  |  +--rw prefix-length?                          uint8
        |  |  +--rw address-allocation-type?                identityref
        |  |  +--rw (allocation-type)?
        |  |     +--:(dynamic)
        |  |     |  +--rw (address-assign)?
        |  |     |  |  +--:(number)
        |  |     |  |  |  +--rw number-of-dynamic-address?   uint16
        |  |     |  |  +--:(explicit)
        |  |     |  |     +--rw customer-addresses
        |  |     |  |        +--rw address-pool* [pool-id]
        |  |     |  |           +--rw pool-id          string
        |  |     |  |           +--rw start-address    inet:ipv4-address
        |  |     |  |           +--rw end-address?     inet:ipv4-address
        |  |     |  +--rw (provider-dhcp)?
        |  |     |  |  +--:(dhcp-service-type)
        |  |     |  |     +--rw dhcp-service-type?           enumeration
        |  |     |  +--rw (dhcp-relay)?
        |  |     |     +--:(customer-dhcp-servers)
        |  |     |        +--rw customer-dhcp-servers
        |  |     |           +--rw server-ip-address*   inet:ipv4-addres
        |  |     +--:(static-addresses)
        |  |        +--rw address* [address-id]
        |  |           +--rw address-id          string
        |  |           +--rw customer-address?   inet:ipv4-address
        |  +--rw ipv6 {vpn-common:ipv6}?
        |     +--rw local-address?                          inet:ipv6-a
        |     +--rw prefix-length?                          uint8
        |     +--rw address-allocation-type?                identityref
        |     +--rw (allocation-type)?
        |        +--:(dynamic)
        |        |  +--rw (address-assign)?
        |        |  |  +--:(number)
        |        |  |  |  +--rw number-of-dynamic-address?   uint16
        |        |  |  +--:(explicit)
        |        |  |     +--rw customer-addresses
```

```
|        |  |        +--rw address-pool* [pool-id]
|        |  |           +--rw pool-id          string
|        |  |           +--rw start-address    inet:ipv6-address
|        |  |           +--rw end-address?     inet:ipv6-address
|        |  +--rw (provider-dhcp)?
|        |  |  +--:(dhcp-service-type)
|        |  |     +--rw dhcp-service-type?           enumeration
|        |  +--rw (dhcp-relay)?
|        |     +--:(customer-dhcp-servers)
|        |        +--rw customer-dhcp-servers
|        |           +--rw server-ip-address*   inet:ipv6-addres
|        +--:(static-addresses)
|           +--rw address* [address-id]
|              +--rw address-id          string
|              +--rw customer-address?   inet:ipv6-address
+--rw routing-protocols
|  ...
+--rw oam
|  ...
+--rw security
   ...
```

Figure 15: Layer 3 Connection Tree Structure

4.5.3. Routing

As shown in the tree depicted in Figure 16, the 'routing-protocols' container defines th erequired parameters to enable the required routing features for an AC. One or more routing protocols can be associated with an AC. Such routing protocols are then enabled between a PE and the CE. Each routing instance is uniquely identified to accommodate scenarios where multiple instances of the same routing protocol have to be configured on the same link.

In addition to static routing, the module supports the following routing protocols:

*BGP [RFC4271]

*OSPF [RFC4577] or [RFC6565]

*IS-IS [ISO10589][RFC1195][RFC5308]

*RIP [RFC2453]

The model also supports the Virtual Router Redundancy Protocol (VRRP) [RFC5798] on an AC.

```
module: ietf-ac-svc
  +--rw specific-provisioning-profiles
  |  ...
  +--rw service-provisioning-profiles
  |  ...
  +--rw attachment-circuits
     +--rw ac-global-profile* [id]
     |  ...f
     +--rw ac-node-group* [id]
     |  ...
     +--rw ac* [name]
        ...
        +--rw name                  string
        +--rw l2-connection
        |  ...
        +--rw ip-connection
        |  ...
        +--rw routing-protocols
        |  +--rw routing-protocol* [id]
        |     +--rw id                  string
        |     +--rw type?               identityref
        |     +--rw routing-profiles* [id]
        |     |  +--rw id      routing-profile-reference
        |     |  +--rw type?   identityref
        |     +--rw static
        |     |  +--rw cascaded-lan-prefixes
        |     |     +--rw ipv4-lan-prefixes* [lan next-hop] {vpn-common:
        |     |     |  +--rw lan         inet:ipv4-prefix
        |     |     |  +--rw lan-tag?    string
        |     |     |  +--rw next-hop    union
        |     |     |  +--rw status
        |     |     |     +--rw admin-status
        |     |     |     |  +--rw status?        identityref
        |     |     |     |  +--rw last-change?   yang:date-and-time
        |     |     |     +--ro oper-status
        |     |     |        +--ro status?        identityref
        |     |     |        +--ro last-change?   yang:date-and-time
        |     |     +--rw ipv6-lan-prefixes* [lan next-hop] {vpn-common:
        |     |        +--rw lan         inet:ipv6-prefix
        |     |        +--rw lan-tag?    string
        |     |        +--rw next-hop    union
        |     |        +--rw status
        |     |           +--rw admin-status
        |     |           |  +--rw status?        identityref
        |     |           |  +--rw last-change?   yang:date-and-time
        |     |           +--ro oper-status
        |     |              +--ro status?        identityref
        |     |              +--ro last-change?   yang:date-and-time
        |     +--rw bgp
        |     |  +--rw peer-groups
        |     |  |  +--rw peer-group* [name]
        |     |  |     +--rw name              string
        |     |  |     +--ro local-address?    inet:ip-address
```

```
|     |  |     +--rw peer-as           inet:as-number
|     |  |     +--rw address-family?   identityref
|     |  +--rw neighbor* [remote-address]
|     |     +--rw remote-address    inet:ip-address
|     |     +--ro local-address?    inet:ip-address
|     |     +--rw peer-group?       -> ../../peer-groups/peer-gr
|     |     +--rw peer-as           inet:as-number
|     |     +--rw address-family?   identityref
|     |     +--rw status
|     |        +--rw admin-status
|     |        |  +--rw status?        identityref
|     |        |  +--rw last-change?   yang:date-and-time
|     |        +--ro oper-status
|     |           +--ro status?        identityref
|     |           +--ro last-change?   yang:date-and-time
|     +--rw ospf
|     |  +--rw address-family?   identityref
|     |  +--rw area-id           yang:dotted-quad
|     |  +--rw metric?           uint16
|     |  +--rw status
|     |     +--rw admin-status
|     |     |  +--rw status?        identityref
|     |     |  +--rw last-change?   yang:date-and-time
|     |     +--ro oper-status
|     |        +--ro status?        identityref
|     |        +--ro last-change?   yang:date-and-time
|     +--rw isis
|     |  +--rw address-family?   identityref
|     |  +--rw area-address      area-address
|     |  +--rw status
|     |     +--rw admin-status
|     |     |  +--rw status?        identityref
|     |     |  +--rw last-change?   yang:date-and-time
|     |     +--ro oper-status
|     |        +--ro status?        identityref
|     |        +--ro last-change?   yang:date-and-time
|     +--rw rip
|     |  +--rw address-family?   identityref
|     |  +--rw status
|     |     +--rw admin-status
|     |     |  +--rw status?        identityref
|     |     |  +--rw last-change?   yang:date-and-time
|     |     +--ro oper-status
|     |        +--ro status?        identityref
|     |        +--ro last-change?   yang:date-and-time
|     +--rw vrrp
|        +--rw address-family?   identityref
|        +--rw status
|           +--rw admin-status
|           |  +--rw status?        identityref
|           |  +--rw last-change?   yang:date-and-time
|           +--ro oper-status
|              +--ro status?        identityref
```

```
|              +--ro last-change?   yang:date-and-time
+--rw oam
|  ...
+--rw security
   ...
```

Figure 16: Routing Tree Structure

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].

4.5.4. OAM

As shown in the tree depicted in Figure 17, the 'oam' container defines OAM-related parameters of an AC.

```
module: ietf-ac-svc
  +--rw specific-provisioning-profiles
  |  ...
  +--rw service-provisioning-profiles
  |  ...
  +--rw attachment-circuits
     +--rw ac-global-profile* [id]
     |  ...f
     +--rw ac-node-group* [id]
     |  ...
     +--rw ac* [name]
        ...
        +--rw name                  string
        +--rw l2-connection
        |  ...
        +--rw ip-connection
        |  ...
        +--rw routing-protocols
        |  ...
        +--rw oam
        |  +--rw bfd {vpn-common:bfd}?
        |     +--rw holdtime?   uint32
        |     +--rw status
        |        +--rw admin-status
        |        |  +--rw status?        identityref
        |        |  +--rw last-change?   yang:date-and-time
        |        +--ro oper-status
        |           +--ro status?        identityref
        |           +--ro last-change?   yang:date-and-time
        +--rw security
           ...
```

Figure 17: OAM Tree Structure

4.5.5. Security

As shown in the tree depicted in Figure 18, the 'security' container defines sercurity parameters of an AC.

```
module: ietf-ac-svc
  +--rw specific-provisioning-profiles
  |  ...
  +--rw service-provisioning-profiles
  |  ...
  +--rw attachment-circuits
     +--rw ac-global-profile* [id]
     |  ...f
     +--rw ac-node-group* [id]
     |  ...
     +--rw ac* [name]
        ...
        +--rw name                  string
        +--rw l2-connection
        |  ...
        +--rw ip-connection
        |  ...
        +--rw routing-protocols
        |  ...
        +--rw oam
        |  ...
        +--rw security
           +--rw encryption {vpn-common:encryption}?
           |  +--rw enabled?   boolean
           |  +--rw layer?     enumeration
           +--rw encryption-profile
              +--rw (profile)?
                 +--:(customer-profile)
                    +--rw customer-key-chain?   key-chain:key-chain-ref
```

Figure 18: Security Tree Structure

5. YANG Module

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

```
<CODE BEGINS> file "ietf-ac-svc@2022-11-30.yang"

module ietf-ac-svc {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ac-svc";
  prefix ac-svc;

  import ietf-vpn-common {
    prefix vpn-common;
    reference
      "RFC 9181: A Common YANG Data Model for Layer 2 and Layer 3
                 VPNs";
  }
  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>

     Author:   Mohamed Boucadair
               <mailto:mohamed.boucadair@orange.com>
     Author:   Richard Roberts
               <mailto:rroberts@juniper.net>";
  description
    "This YANG module defines a generic YANG model for
     the attachment circuits exposed outside a network.

     Copyright (c) 2022 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

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

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

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

// Identities

identity address-allocation-type {
  description
    "Base identity for address allocation type in the
     Provider Edge to Customer Edge (PE-CE) link.";
}

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

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

identity provider-dhcp-slaac {
  if-feature "vpn-common:ipv6";
  base address-allocation-type;
  description
    "The provider's network provides a DHCP service to the
     customer as well as IPv6 Stateless Address
     Autoconfiguration (SLAAC).";
  reference
    "RFC 4862: IPv6 Stateless Address Autoconfiguration";
}

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

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

```
  description
    "The provider's network uses IPv6 SLAAC to provide
     addressing to the customer.";
  reference
    "RFC 4862: IPv6 Stateless Address Autoconfiguration";
}

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

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.";
}

identity l2-tunnel-type {
  description
    "Base identity for Layer 2 tunnel selection under the VPN
     network access.";
}

identity pseudowire {
  base l2-tunnel-type;
  description
    "Pseudowire tunnel termination in the VPN network access.";
}

identity vpls {
  base l2-tunnel-type;
  description
    "Virtual Private LAN Service (VPLS) tunnel termination in
     the VPN network access.";
}
```

```
identity vxlan {
  base l2-tunnel-type;
  description
    "Virtual eXtensible Local Area Network (VXLAN) tunnel
     termination in the VPN network access.";
}

/* 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.";
}

typedef attachment-circuit-reference {
  type leafref {
    path "/ac-svc:attachment-circuits/ac-svc:ac/ac-svc:name";
  }
  description
    "Defines a reference to an attachment circuit that can be used
     by other modules.";
}

typedef ac-global-profile-reference {
  type leafref {
    path "/ac-svc:attachment-circuits/ac-svc:ac-global-profile"
        + "/ac-svc:id";
  }
  description
    "Defines a reference to a gloabl attachment circuit that can be us
     by other modules.";
}


typedef ac-node-group-reference {
  type leafref {
    path "/ac-svc:attachment-circuits/ac-svc:ac-node-group"
        + "/ac-svc:id";
  }
  description
```

```
    "Defines a reference to a per-node attachment circuit that can be
     by other modules.";
}

typedef encryption-profile-reference {
  type leafref {
    path "/specific-provisioning-profiles/valid-provider-identifiers"
       + "/encryption-profile-identifier/id";
  }
  description
    "Defines a reference to an encryption profile for referencing
     purposes.";
}

typedef qos-profile-reference {
  type leafref {
    path "/specific-provisioning-profiles/valid-provider-identifiers"
       + "/qos-profile-identifier/id";
  }
  description
    "Defines a reference to a QoS profile for referencing
     purposes.";
}

typedef bfd-profile-reference {
  type leafref {
    path "/specific-provisioning-profiles/valid-provider-identifiers"
       + "/bfd-profile-identifier/id";
  }
  description
    "Defines a reference to a BFD profile for referencing
     purposes.";
}

typedef forwarding-profile-reference {
  type leafref {
    path "/specific-provisioning-profiles/valid-provider-identifiers"
       + "/forwarding-profile-identifier/id";
  }
  description
    "Defines a reference to a forwarding profile for referencing
     purposes.";
}

typedef routing-profile-reference {
  type leafref {
    path "/specific-provisioning-profiles/valid-provider-identifiers"
       + "/routing-profile-identifier/id";
  }
  description
    "Defines a reference to a routing profile for referencing
     purposes.";
}
```

```
// L2 connection

grouping l2-connection-profile {
  description
    "Defines Layer 2 protocols and parameters that can be
     factorized when provisioning Layer 2 connectivity.";
  container encapsulation {
    description
      "Container for Layer 2 encapsulation.";
    leaf type {
      type identityref {
        base vpn-common:encapsulation-type;
      }
      //default "vpn-common:priority-tagged";
      description
        "Encapsulation type.  By default, the type
         of the tagged interface is 'priority-tagged'.";
    }
    container dot1q {
      when "derived-from-or-self(../type, 'vpn-common:dot1q')" {
        description
          "Only applies when the type of the tagged interface
           is 'dot1q'.";
      }
      description
        "Tagged interface.";
      leaf tag-type {
        type identityref {
          base vpn-common:tag-type;
        }
        default "vpn-common:c-vlan";
        description
          "Tag type.  By default, the tag type is 'c-vlan'.";
      }
      leaf cvlan-id {
        type uint16 {
          range "1..4094";
        }
        description
          "VLAN identifier.";
      }
    }
    container qinq {
      when "derived-from-or-self(../type, 'vpn-common:qinq')" {
        description
          "Only applies when the type of the tagged interface
           is 'qinq'.";
      }
      description
        "Includes QinQ parameters.";
      leaf tag-type {
        type identityref {
```

```
          base vpn-common:tag-type;
        }
        default "vpn-common:s-c-vlan";
        description
          "Tag type.";
      }
      leaf svlan-id {
        type uint16;
        mandatory true;
        description
          "Service VLAN (S-VLAN) identifier.";
      }
      leaf cvlan-id {
        type uint16;
        mandatory true;
        description
          "Customer VLAN (C-VLAN) identifier.";
      }
    }
  }
}

grouping l2-connection {
  description
    "Defines Layer 2 protocols and parameters that are used to enable
     AC connectivity.";
  container encapsulation {
    description
      "Container for Layer 2 encapsulation.";
    leaf type {
      type identityref {
        base vpn-common:encapsulation-type;
      }
      //default "vpn-common:priority-tagged";
      description
        "Encapsulation type.  By default, the type of the tagged
         interface is 'priority-tagged'.";
    }
    container dot1q {
      when "derived-from-or-self(../type, 'vpn-common:dot1q')" {
        description
          "Only applies when the type of the tagged interface
           is 'dot1q'.";
      }
      description
        "Tagged interface.";
      leaf tag-type {
        type identityref {
          base vpn-common:tag-type;
        }
        default "vpn-common:c-vlan";
        description
          "Tag type.  By default, the tag type is 'c-vlan'.";
```

```
  }
  leaf cvlan-id {
    type uint16 {
      range "1..4094";
    }
    description
      "VLAN identifier.";
  }
}
container priority-tagged {
  when "derived-from-or-self(../type, "
     + "'vpn-common:priority-tagged')" {
    description
      "Only applies when the type of the tagged interface is
       'priority-tagged'.";
  }
  description
    "Priority tagged.";
  leaf tag-type {
    type identityref {
      base vpn-common:tag-type;
    }
    default "vpn-common:c-vlan";
    description
      "Tag type.  By default, the tag type is 'c-vlan'.";
  }
}
container qinq {
  when "derived-from-or-self(../type, 'vpn-common:qinq')" {
    description
      "Only applies when the type of the tagged interface
       is 'qinq'.";
  }
  description
    "Includes QinQ parameters.";
  leaf tag-type {
    type identityref {
      base vpn-common:tag-type;
    }
    default "vpn-common:s-c-vlan";
    description
      "Tag type.";
  }
  leaf svlan-id {
    type uint16;
    mandatory true;
    description
      "Service VLAN (S-VLAN) identifier.";
  }
  leaf cvlan-id {
    type uint16;
    mandatory true;
    description
```

```
        "Customer VLAN (C-VLAN) identifier.";
    }
  }
}
choice l2-service {
  description
    "The Layer 2 connectivity service can be provided by indicating
     a pointer to an L2VPN or by specifying a Layer 2 tunnel
     service.";
  container l2-tunnel-service {
    description
      "Defines a Layer 2 tunnel termination.
       It is only applicable when a tunnel is required.
       The supported values are 'pseudowire', 'vpls', and 'vxlan'.
       Other values may be defined, if needed.";
    leaf type {
      type identityref {
        base l2-tunnel-type;
      }

      description
        "Selects the tunnel termination option for each AC
         Endpoint.";
    }
    container pseudowire {
      when "derived-from-or-self(../type, 'pseudowire')" {
        description
          "Only applies when the Layer 2 service type is
           'pseudowire'.";
      }
      description
        "Includes pseudowire termination parameters.";
      leaf vcid {
        type uint32;
        description
          "Indicates a pseudowire (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";
      }
    }
    container vpls {
      when "derived-from-or-self(../type, 'vpls')" {
```

```
      description
        "Only applies when the Layer 2 service type is
         '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.";
      }
    }
    container vxlan {
      when "derived-from-or-self(../type, 'vxlan')" {
        description
          "Only applies when the Layer 2 service type is
           'vxlan'.";
      }
      description
        "VXLAN termination parameters.";
      leaf vni-id {
        type uint32;
        mandatory true;
        description
          "VXLAN Network Identifier (VNI).";
      }
      leaf peer-mode {
        type identityref {
          base vpn-common:vxlan-peer-mode;
        }
        default "vpn-common:static-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.";
      }
    }
  }
  case l2vpn {
    leaf l2vpn-id {
      type vpn-common:vpn-id;
```

```
        description
          "Indicates the L2VPN service associated with an Integrated
           Routing and Bridging (IRB) interface.";
      }
    }
  }
  leaf bearer-reference {
    if-feature "vpn-common:bearer-reference";
    type string;
    description
      "This is an internal reference for the service provider
       to identify the bearer associated with this AC.";
  }
}

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

         If there is no value for the address allocation type,
         then IPv4 addressing is not enabled.";
    }
    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
             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.";
          }
        }
      }
    }
  }
}
container ipv6 {
  if-feature "vpn-common:ipv6";
  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;
```

```
      }
      default "static-address";
      description
        "Defines how addresses are allocated.
         If there is no value for the address allocation type,
         then IPv6 addressing is disabled.";
    }
    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 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.";
            }
          }
        }
      }
    }
  }
}
```

```
grouping ip-connection {
  description
    "Defines IP connection parameters.";
  container ipv4 {
    if-feature "vpn-common:ipv4";
    description
      "IPv4-specific parameters.";
    leaf local-address {
      type inet:ipv4-address;
      description
        "The IP address used at the provider's interface.";
    }
    leaf prefix-length {
      type uint8 {
        range "0..32";
      }
      description
        "Subnet prefix length expressed in bits.
         It is applied to both local and customer addresses.";
    }
    leaf address-allocation-type {
      type identityref {
        base address-allocation-type;
      }
      must "not(derived-from-or-self(current(), 'slaac') or "
         + "derived-from-or-self(current(), "
         + "'provider-dhcp-slaac'))" {
        error-message "SLAAC is only applicable to IPv6.";
      }
      default "static-address";
      description
        "Defines how addresses are allocated to the peer site.

         If there is no value for the address allocation type,
         then IPv4 addressing is not enabled.";
    }
    choice allocation-type {
      description
        "Choice of the IPv4 address allocation.";
      case dynamic {
        description
          "When the addresses are allocated by DHCP or other
           dynamic means local to the infrastructure.";
        choice address-assign {
          default "number";
          description
            "A choice for how IPv4 addresses are assigned.";
          case number {
            leaf number-of-dynamic-address {
              type uint16;
              default "1";
              description
```

```
        "Specifies the number of IP addresses to be
         assigned to the customer on this access.";
      }
    }
    case explicit {
      container customer-addresses {
        description
          "Container for customer addresses to be allocated
           using DHCP.";
        list address-pool {
          key "pool-id";
          description
            "Describes IP addresses to be dyncamically allocated

             When only 'start-address' is present, it represents
             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.";
      /*leaf primary-address {
        type leafref {
          path "../address/address-id";
        }
        description
          "Primary IP address of the connection.";
      }*/
      list address {
        key "address-id";
        ordered-by user;
        description
          "Lists the IPv4 addresses that are used. The first address
           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.";
        }
      }
    }
  }
```

```
}
container ipv6 {
  if-feature "vpn-common:ipv6";
  description
    "IPv6-specific parameters.";
  leaf local-address {
    type inet:ipv6-address;
    description
      "IPv6 address of the provider side.";
  }
  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;
    }
    default "static-address";
    description
      "Defines how addresses are allocated.
       If there is no value for the address allocation type,
       then IPv6 addressing is disabled.";
  }
  choice allocation-type {
    description
      "Choice of the IPv6 address allocation.";
    case dynamic {
      description
        "When the addresses are allocated by DHCP or other
         dynamic means local to the infrastructure.";
      choice address-assign {
        default "number";
        description
          "A choice for how IPv6 addresses are assigned.";
        case number {
          leaf number-of-dynamic-address {
            type uint16;
            default "1";
            description
              "Specifies the number of IP addresses to be assigned t
               the customer on this access.";
          }
        }
        case explicit {
          container customer-addresses {
            description
              "Container for customer addresses to be allocated
               using DHCP.";
```

```
      list address-pool {
        key "pool-id";
        description
          "Describes IP addresses to be dyncamically allocated

           When only 'start-address' is present, it represents
           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.";
        /*leaf primary-address {
          type leafref {
            path "../address/address-id";
          }
          description
            "Primary IP address of the connection.";
        }*/
        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 */

grouping routing-profile {
  description
    "Defines routing protocols.";
  list routing-protocol {
```

```
key "id";
description
  "List of routing protocols used on the AC.";
leaf id {
  type string;
  description
    "Unique identifier for the routing protocol.";
}
leaf type {
  type identityref {
    base vpn-common:routing-protocol-type;
  }
  description
    "Type of routing protocol.";
}
list routing-profiles {
  key "id";
  description
    "Routing profiles.";
  leaf id {
    type routing-profile-reference;
    description
      "Reference to the routing profile to be used.";
  }
  leaf type {
    type identityref {
      base vpn-common:ie-type;
    }
    description
      "Import, export, or both.";
  }
}
container bgp {
  when "derived-from-or-self(../type, 'vpn-common:bgp-routing')" {
    description
      "Only applies when the protocol is BGP.";
  }
  description
    "Configuration specific to BGP.";
  container peer-groups {
    description
      "Configuration for BGP peer-groups";
    list peer-group {
      key "name";
      description
        "List of BGP peer-groups configured on the local system -
         uniquely identified by peer-group name";
      leaf name {
        type string;
        description
          "Name of the BGP peer-group";
      }
      leaf peer-as {
```

```
        type inet:as-number;
        mandatory true;
        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 IPv6e
           will b activated.";
      }
    }
  }
}
container ospf {
  when "derived-from-or-self(../type, "
     + "'vpn-common:ospf-routing')" {
    description
      "Only applies when the protocol is OSPF.";
  }
  description
    "Configuration specific to OSPF.";
  leaf address-family {
    type identityref {
      base vpn-common:address-family;
    }
    description
      "Indicates whether IPv4, IPv6, or both are to be
       activated.";
  }
  leaf area-id {
    type yang:dotted-quad;
    mandatory true;
    description
      "Area ID.";
    reference
      "RFC 4577: OSPF as the Provider/Customer Edge Protocol
                 for BGP/MPLS IP Virtual Private Networks
                 (VPNs), Section 4.2.3
       RFC 6565: OSPFv3 as a Provider Edge to Customer Edge
                 (PE-CE) Routing Protocol, Section 4.2";
  }
  leaf metric {
    type uint16;
    default "1";
    description
      "Metric of the AC.  It is used in the routing state
       calculation and path selection.";
  }
```

```
}
container isis {
  when "derived-from-or-self(../type, "
     + "'vpn-common:isis-routing')" {
    description
      "Only applies when the protocol is IS-IS.";
  }
  description
    "Configuration specific to IS-IS.";
  leaf address-family {
    type identityref {
      base vpn-common:address-family;
    }
    description
      "Indicates whether IPv4, IPv6, or both are to
       be activated.";
  }
  leaf area-address {
    type area-address;
    mandatory true;
    description
      "Area address.";
  }
}
container rip {
  when "derived-from-or-self(../type, "
     + "'vpn-common:rip-routing')" {
    description
      "Only applies when the protocol is RIP.
       For IPv4, the model assumes that RIP
       version 2 is used.";
  }
  description
    "Configuration specific to RIP routing.";
  leaf address-family {
    type identityref {
      base vpn-common:address-family;
    }
    description
      "Indicates whether IPv4, IPv6, or both
       address families are to be activated.";
  }
}
container vrrp {
  when "derived-from-or-self(../type, "
     + "'vpn-common:vrrp-routing')" {
    description
      "Only applies when the protocol is the
       Virtual Router Redundancy Protocol (VRRP).";
  }
  description
    "Configuration specific to VRRP.";
  reference
```

```
        "RFC 5798: Virtual Router Redundancy Protocol (VRRP)
                   Version 3 for IPv4 and IPv6";

      leaf address-family {
        type identityref {
          base vpn-common:address-family;
        }
        description
          "Indicates whether IPv4, IPv6, or both
           address families are to be enabled.";
      }
    }
  }
}

grouping routing {
  description
    "Defines routing protocols.";
  list routing-protocol {
    key "id";
    description
      "List of routing protocols used on the AC.";
    leaf id {
      type string;
      description
        "Unique identifier for the routing protocol.";
    }
    leaf type {
      type identityref {
        base vpn-common:routing-protocol-type;
      }
      description
        "Type of routing protocol.";
    }
    list routing-profiles {
      key "id";
      description
        "Routing profiles.";
      leaf id {
        type routing-profile-reference;
        description
          "Reference to the routing profile to be used.";
      }
      leaf type {
        type identityref {
          base vpn-common:ie-type;
        }
        description
          "Import, export, or both.";
      }
    }
    container static {
      when "derived-from-or-self(../type, "
```

```
   + "'vpn-common:static-routing')" {
  description
    "Only applies when the protocol is a
     static routing protocol.";
}
description
  "Configuration specific to static
   routing.";
container cascaded-lan-prefixes {
  description
    "LAN prefixes from the customer.";
  list ipv4-lan-prefixes {
    if-feature "vpn-common:ipv4";
    key "lan next-hop";
    description
      "List of LAN prefixes for the site.";
    leaf lan {
      type inet:ipv4-prefix;
      description
        "LAN prefixes.";
    }
    leaf lan-tag {
      type string;
      description
        "Internal tag to be used in 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').";
    }
    uses vpn-common:service-status;
  }
  list ipv6-lan-prefixes {
    if-feature "vpn-common:ipv6";
    key "lan next-hop";
    description
      "List of LAN prefixes for the site.";
    leaf lan {
      type inet:ipv6-prefix;
      description
        "LAN prefixes.";
    }
    leaf lan-tag {
      type string;
      description
        "Internal tag to be used in 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').";
      }
      uses vpn-common:service-status;
    }
  }
}
container bgp {
  when "derived-from-or-self(../type, "
     + "'vpn-common:bgp-routing')" {
    description
      "Only applies when the protocol is BGP.";
  }
  description
    "Configuration specific to BGP.";
  container peer-groups {
    description
      "Configuration for BGP peer-groups";
    list peer-group {
      key "name";
      description
        "List of BGP peer-groups configured on the local system -
         uniquely identified by peer-group name";
      leaf name {
        type string;
        description
          "Name of the BGP peer-group.";
      }
      leaf local-address {
        type inet:ip-address;
        config false;
        description
          "The local IP address that will be used to establish
           the BGP session.";
      }
      leaf peer-as {
        type inet:as-number;
        mandatory true;
        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.";
      }
    }
  }
  list neighbor {
    key "remote-address";
    description
      "List of BGP neighbors.";
    leaf remote-address {
      type inet:ip-address;
      description
        "The remote IP address of this entry's BGP peer.";
    }
    leaf local-address {
      type inet:ip-address;
      config false;
      description
        "The local IP address that will be used to establish
         the BGP session.";
    }
    leaf peer-group {
      type leafref {
        path "../../peer-groups/peer-group/name";
      }
      description
        "The peer-group with which this neighbor is
         associated.";
    }
    leaf peer-as {
      type inet:as-number;
      mandatory true;
      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.";
    }
    uses vpn-common:service-status;
  }
}
container ospf {
  when "derived-from-or-self(../type, "
```

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

```
        mandatory true;
        description
          "Area address.";
      }
      uses vpn-common:service-status;
    }
    container rip {
      when "derived-from-or-self(../type, "
         + "'vpn-common:rip-routing')" {
        description
          "Only applies when the protocol is RIP.
           For IPv4, the model assumes that RIP version 2 is used.";
      }
      description
        "Configuration specific to RIP routing.";
      leaf address-family {
        type identityref {
          base vpn-common:address-family;
        }
        description
          "Indicates whether IPv4, IPv6, or both address families
           are to be activated.";
      }
      uses vpn-common:service-status;
    }
    container vrrp {
      when "derived-from-or-self(../type, "
         + "'vpn-common:vrrp-routing')" {
        description
          "Only applies when the protocol is the
           Virtual Router Redundancy Protocol (VRRP).";
      }
      description
        "Configuration specific to VRRP.";
      reference
        "RFC 5798: Virtual Router Redundancy Protocol (VRRP)
                   Version 3 for IPv4 and IPv6";
      leaf address-family {
        type identityref {
          base vpn-common:address-family;
        }
        description
          "Indicates whether IPv4, IPv6, or both
           address families are to be enabled.";
      }
      uses vpn-common:service-status;
    }
  }
}

grouping ac {
  description
    "Grouping for an attachment circuit.";
```

```
leaf name {
  type string;
  description
    "A name of the AC. Data models that need to
     reference an attachment circuits should use
     attachment-circuit-reference.";
}
// Layer 2
container l2-connection {
  description
    "Defines Layer 2 protocols and parameters that
     are required to enable AC connectivity.";
  uses l2-connection;
}
// Layer 3
container ip-connection {
  description
    "Defines IP connection parameters.";
  uses ip-connection;
}
// Routing
container routing-protocols {
  description
    "Defines routing protocols.";
  uses routing;
}
// OAM
container oam {
  description
    "Defines the Operations, Administration, and Maintenance
     (OAM) mechanisms used.

     BFD is set as a fault detection mechanism,
     but other mechanisms can be defined in the future.";
  container bfd {
    if-feature "vpn-common:bfd";
    description
      "Container for BFD.";
    leaf holdtime {
      type uint32;
      units "milliseconds";
      description
        "Expected BFD holdtime.

         The customer may impose some fixed values
         for the holdtime period if the provider allows
         the customer to use this function.

         If the provider doesn't allow the customer to
         use this function, fixed values will not be set.";
      reference
        "RFC 5880: Bidirectional Forwarding
                   Detection (BFD), Section 6.8.18";
```

```
    }
    uses vpn-common:service-status;
  }
}
// Security
container security {
  description
    "Site-specific security parameters.";
  container encryption {
    if-feature "vpn-common:encryption";
    description
      "Container for AC security encryption.";
    leaf enabled {
      type boolean;
      default "false";
      description
        "If set to 'true', traffic encryption on the connection is
         required.  Otherwise, it is disabled.";
    }
    leaf layer {
      when "../enabled = 'true'" {
        description
          "Included only when encryption is enabled.";
      }
      type enumeration {
        enum layer2 {
          description
            "Encryption occurs at Layer 2.";
        }
        enum layer3 {
          description
            "Encryption occurs at Layer 3.
             For example, IPsec may be used when a customer
             requests Layer 3 encryption.";
        }
      }
      description
        "Indicates the layer on which encryption is
         applied.";
    }
  }
  container encryption-profile {
    when "../encryption/enabled = 'true'" {
      description
        "Indicates the layer on which encryption is
         enabled.";
    }
    description
      "Container for the encryption profile.";
    choice profile {
      description
        "Choice for the encryption profile.";
      case customer-profile {
```

```
          leaf customer-key-chain {
            type key-chain:key-chain-ref;
            description
              "Customer-supplied key chain.";
          }
        }
      }
    }
  }
}

grouping ac-profile {
  description
    "Grouping for an attachment circuit.";
  leaf id {
    type string;
    description
      "An identifier of the AC.";
  }
  // Layer 2
  container l2-connection {
    description
      "Defines Layer 2 protocols and parameters that
       are required to enable AC connectivity.";
    uses l2-connection-profile;
  }
  // Layer 3
  container ip-connection {
    description
      "Defines IP connection parameters.";
   uses ip-connection-global-profile;
  }
  // Routing
  container routing-protocols {
    description
      "Defines routing protocols.";
    uses routing-profile;
  }
  // OAM
  container oam {
    description
      "Defines the OAM mechanisms used.

       BFD is set as a fault detection mechanism, but
       other mechanisms can be defined in the future.";
    container bfd {
      if-feature "vpn-common:bfd";
      description
        "Container for BFD.";
      leaf holdtime {
        type uint32;
        units "milliseconds";
        description
```

```
        "Expected BFD holdtime.

         The customer may impose some fixed values
         for the holdtime period if the provider allows
         the customer to use this function.

         If the provider doesn't allow the customer to use
         this function, fixed values will not be set.";
      reference
        "RFC 5880: Bidirectional Forwarding
                   Detection (BFD), Section 6.8.18";
    }
  }
}
// Security
container security {
  description
    "Site-specific security parameters.";
  container encryption {
    if-feature "vpn-common:encryption";
    description
      "Container for AC security encryption.";
    leaf enabled {
      type boolean;
      default "false";
      description
        "If set to 'true', traffic encryption on the connection
         is required.  Otherwise, it is disabled.";
    }
    leaf layer {
      when "../enabled = 'true'" {
        description
          "Included only when encryption is enabled.";
      }
      type enumeration {
        enum layer2 {
          description
            "Encryption occurs at Layer 2.";
        }
        enum layer3 {
          description
            "Encryption occurs at Layer 3.
             For example, IPsec may be used when
             a customer requests Layer 3 encryption.";
        }
      }
      description
        "Indicates the layer on which encryption is applied.";
    }
  }
  container encryption-profile {
    when "../encryption/enabled = 'true'" {
      description
```

```
          "Indicates the layer on which encryption is enabled.";
      }
      description
        "Container for the encryption profile.";
      choice profile {
        description
          "Choice for the encryption profile.";
        case customer-profile {
        leaf customer-key-chain {
            type key-chain:key-chain-ref;
            description
              "Customer-supplied key chain.";
          }
        }
      }
    }
  }
}

container specific-provisioning-profiles {
  description
    "Contains a set of valid profiles to reference for an AC.";
  uses vpn-common:vpn-profile-cfg;
}
container service-provisioning-profiles {
  description
    "Contains a set of valid profiles to reference for an AC.";
  list service-profile-identifier {
    key "id";
    description
      "List of generic service profile identifiers.";
    leaf id {
      type string;
      description
        "Identification of the service profile to be used.
         the profile only has significance within the service
         provider's administrative domain.";
    }
  }
}
container attachment-circuits {
  description
    "Main container for the attachment circuits.";
  list ac-global-profile {
    key "id";
    description
      "Maintains a list of AC profiles.";
    uses ac-profile;
  }
  list ac-node-group {
    key "id";
    description
      "Maintains a list of per-node AC profiles.";
```

```
  leaf id {
    type string;
    description
      "An identifier of the AC group.";
  }
  uses ac;
}
list ac {
  key "name";
  description
    "Global provisioning of attachment circuits.";
  leaf customer-name {
    type string;
    description
      "Indicates the name of the customer that requested this
       AC.";
  }
  leaf description {
    type string;
    description
      "Associates a description with an AC.";
  }
  leaf requested-ac-start {
    type yang:date-and-time;
    description
      "Indicates the requested date and time when the AC is
       expected to be active.";
  }
  leaf requested-ac-stop {
    type yang:date-and-time;
    description
      "Indicates the requested date and time when the AC is
       expected to be disabled.";
  }
  leaf actual-ac-start {
    type yang:date-and-time;
    config false;
    description
      "Indciates the actual date and time when the AC
       actually was enabled.";
  }
  leaf actual-ac-stop {
    type yang:date-and-time;
    config false;
    description
      "Indciates the actual date and time when the AC
       actually was disabled.";
  }
  leaf-list peer-sap-id {
    type string;
    description
      "One or more peer SAPs can be indicated.";
  }
```

```
      leaf-list ac-global-profile {
        /*type leafref {
          path "/attachment-circuits/ac-global-profile/id";
        }*/
        type ac-global-profile-reference;
        description
          "A reference to an AC profile.";
      }
      leaf-list ac-node-profile {
        /*type leafref {
          path "/attachment-circuits/ac-node-group/id";
        }*/
        type ac-node-group-reference;
        description
          "A reference to a per-node AC profile.";
      }
      uses ac;
    }
  }
}

<CODE ENDS>
```

6. Security Considerations

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

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

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) and delete operations to these data nodes without proper protection or authentication can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/ vulnerability in the "ietf-ac-svc" module:

*TBC

*TBC

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability in the "ietf-ac-svc" module:

*TBC

*TBC

7. IANA Considerations

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

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

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

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

8. References

8.1. Normative References

[I-D.ietf-opsawg-sap] Boucadair, M., de Dios, O. G., Barguil, S., Wu, Q., and V. Lopez, "A YANG Network Model for Service Attachment Points (SAPs)", Work in Progress, Internet-Draft, draft-ietf-opsawg-sap-13, 9 January 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-sap-13>.

[ISO10589] ISO, "Information technology - Telecommunications and information exchange between systems - Intermediate System to Intermediate System intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO8473)", 2002, <https://www.iso.org/standard/30932.html>.

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

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

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

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

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

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

[RFC4577] Rosen, E., Psenak, P., and P. Pillay-Esnault, "OSPF as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4577, DOI 10.17487/

RFC4577, June 2006, <https://www.rfc-editor.org/rfc/rfc4577>.

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

[RFC5798] Nadas, S., Ed., "Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6", RFC 5798, DOI 10.17487/RFC5798, March 2010, <https://www.rfc-editor.org/rfc/rfc5798>.

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

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

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

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

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

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

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

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

[RFC8177] Lindem, A., Ed., Qu, Y., Yeung, D., Chen, I., and J. Zhang, "YANG Data Model for Key Chains", RFC 8177, DOI

10.17487/RFC8177, June 2017, <https://www.rfc-editor.org/rfc/rfc8177>.

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

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/rfc/rfc8342>.

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

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

8.2. Informative References

[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/rfc/rfc3644>.

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

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

Appendix A. Full Tree

```
module: ietf-ac-svc
  +--rw specific-provisioning-profiles
  |  +--rw valid-provider-identifiers
  |     +--rw external-connectivity-identifier* [id] {external-connectiv
  |     |  +--rw id    string
  |     +--rw encryption-profile-identifier* [id]
  |     |  +--rw id    string
  |     +--rw qos-profile-identifier* [id]
  |     |  +--rw id    string
  |     +--rw bfd-profile-identifier* [id]
  |     |  +--rw id    string
  |     +--rw forwarding-profile-identifier* [id]
  |     |  +--rw id    string
  |     +--rw routing-profile-identifier* [id]
  |        +--rw id    string
  +--rw service-provisioning-profiles
  |  +--rw service-profile-identifier* [id]
  |     +--rw id    string
  +--rw attachment-circuits
     +--rw ac-global-profile* [id]
     |  +--rw id                   string
     |  +--rw l2-connection
     |  |  +--rw encapsulation
     |  |     +--rw type?    identityref
     |  |     +--rw dot1q
     |  |     |  +--rw tag-type?   identityref
     |  |     |  +--rw cvlan-id?   uint16
     |  |     +--rw qinq
     |  |        +--rw tag-type?   identityref
     |  |        +--rw svlan-id    uint16
     |  |        +--rw cvlan-id    uint16
     |  +--rw ip-connection
     |  |  +--rw ipv4 {vpn-common:ipv4}?
     |  |  |  +--rw prefix-length?                      uint8
     |  |  |  +--rw address-allocation-type?            identityref
     |  |  |  +--rw (allocation-type)?
     |  |  |     +--:(dynamic)
     |  |  |        +--rw (provider-dhcp)?
     |  |  |        |  +--:(dhcp-service-type)
     |  |  |        |     +--rw dhcp-service-type?      enumeration
     |  |  |        +--rw (dhcp-relay)?
     |  |  |           +--:(customer-dhcp-servers)
     |  |  |              +--rw customer-dhcp-servers
     |  |  |                 +--rw server-ip-address*   inet:ipv4-addres
     |  |  +--rw ipv6 {vpn-common:ipv6}?
     |  |     +--rw prefix-length?                      uint8
     |  |     +--rw address-allocation-type?            identityref
     |  |     +--rw (allocation-type)?
     |  |        +--:(dynamic)
     |  |           +--rw (provider-dhcp)?
     |  |           |  +--:(dhcp-service-type)
     |  |           |     +--rw dhcp-service-type?      enumeration
     |  |           +--rw (dhcp-relay)?
```

```
|  |              +--:(customer-dhcp-servers)
|  |                 +--rw customer-dhcp-servers
|  |                    +--rw server-ip-address*   inet:ipv6-addres
|  +--rw routing-protocols
|  |  +--rw routing-protocol* [id]
|  |     +--rw id                  string
|  |     +--rw type?               identityref
|  |     +--rw routing-profiles* [id]
|  |     |  +--rw id      routing-profile-reference
|  |     |  +--rw type?   identityref
|  |     +--rw bgp
|  |     |  +--rw peer-groups
|  |     |     +--rw peer-group* [name]
|  |     |        +--rw name              string
|  |     |        +--rw peer-as           inet:as-number
|  |     |        +--rw address-family?   identityref
|  |     +--rw ospf
|  |     |  +--rw address-family?   identityref
|  |     |  +--rw area-id           yang:dotted-quad
|  |     |  +--rw metric?           uint16
|  |     +--rw isis
|  |     |  +--rw address-family?   identityref
|  |     |  +--rw area-address      area-address
|  |     +--rw rip
|  |     |  +--rw address-family?   identityref
|  |     +--rw vrrp
|  |        +--rw address-family?   identityref
|  +--rw oam
|  |  +--rw bfd {vpn-common:bfd}?
|  |     +--rw holdtime?   uint32
|  +--rw security
|     +--rw encryption {vpn-common:encryption}?
|     |  +--rw enabled?   boolean
|     |  +--rw layer?     enumeration
|     +--rw encryption-profile
|        +--rw (profile)?
|           +--:(customer-profile)
|              +--rw customer-key-chain?   key-chain:key-chain-ref
+--rw ac-node-group* [id]
|  +--rw id                   string
|  +--rw name?                string
|  +--rw l2-connection
|  |  +--rw encapsulation
|  |  |  +--rw type?              identityref
|  |  |  +--rw dot1q
|  |  |  |  +--rw tag-type?   identityref
|  |  |  |  +--rw cvlan-id?   uint16
|  |  |  +--rw priority-tagged
|  |  |  |  +--rw tag-type?   identityref
|  |  |  +--rw qinq
|  |  |     +--rw tag-type?   identityref
|  |  |     +--rw svlan-id    uint16
|  |  |     +--rw cvlan-id    uint16
```

```
|  |  +--rw (l2-service)?
|  |  |  +--:(l2-tunnel-service)
|  |  |  |  +--rw l2-tunnel-service
|  |  |  |     +--rw type?         identityref
|  |  |  |     +--rw pseudowire
|  |  |  |     |  +--rw vcid?      uint32
|  |  |  |     |  +--rw far-end?   union
|  |  |  |     +--rw vpls
|  |  |  |     |  +--rw vcid?      uint32
|  |  |  |     |  +--rw far-end*   union
|  |  |  |     +--rw vxlan
|  |  |  |        +--rw vni-id             uint32
|  |  |  |        +--rw peer-mode?         identityref
|  |  |  |        +--rw peer-ip-address*   inet:ip-address
|  |  |  +--:(l2vpn)
|  |  |     +--rw l2vpn-id?           vpn-common:vpn-id
|  |  +--rw bearer-reference?         string {vpn-common:bearer-re
|  +--rw ip-connection
|  |  +--rw ipv4 {vpn-common:ipv4}?
|  |  |  +--rw local-address?                          inet:ipv4-a
|  |  |  +--rw prefix-length?                          uint8
|  |  |  +--rw address-allocation-type?                identityref
|  |  |  +--rw (allocation-type)?
|  |  |     +--:(dynamic)
|  |  |     |  +--rw (address-assign)?
|  |  |     |  |  +--:(number)
|  |  |     |  |  |  +--rw number-of-dynamic-address?   uint16
|  |  |     |  |  +--:(explicit)
|  |  |     |  |     +--rw customer-addresses
|  |  |     |  |        +--rw address-pool* [pool-id]
|  |  |     |  |           +--rw pool-id          string
|  |  |     |  |           +--rw start-address    inet:ipv4-address
|  |  |     |  |           +--rw end-address?     inet:ipv4-address
|  |  |     |  +--rw (provider-dhcp)?
|  |  |     |  |  +--:(dhcp-service-type)
|  |  |     |  |     +--rw dhcp-service-type?           enumeration
|  |  |     |  +--rw (dhcp-relay)?
|  |  |     |     +--:(customer-dhcp-servers)
|  |  |     |        +--rw customer-dhcp-servers
|  |  |     |           +--rw server-ip-address*   inet:ipv4-addres
|  |  |     +--:(static-addresses)
|  |  |        +--rw address* [address-id]
|  |  |           +--rw address-id          string
|  |  |           +--rw customer-address?   inet:ipv4-address
|  |  +--rw ipv6 {vpn-common:ipv6}?
|  |     +--rw local-address?                          inet:ipv6-a
|  |     +--rw prefix-length?                          uint8
|  |     +--rw address-allocation-type?                identityref
|  |     +--rw (allocation-type)?
|  |        +--:(dynamic)
|  |        |  +--rw (address-assign)?
|  |        |  |  +--:(number)
|  |        |  |  |  +--rw number-of-dynamic-address?   uint16
```

```
|  |        |  |  +--:(explicit)
|  |        |  |     +--rw customer-addresses
|  |        |  |        +--rw address-pool* [pool-id]
|  |        |  |           +--rw pool-id          string
|  |        |  |           +--rw start-address    inet:ipv6-address
|  |        |  |           +--rw end-address?     inet:ipv6-address
|  |        |  +--rw (provider-dhcp)?
|  |        |  |  +--:(dhcp-service-type)
|  |        |  |     +--rw dhcp-service-type?           enumeration
|  |        |  +--rw (dhcp-relay)?
|  |        |     +--:(customer-dhcp-servers)
|  |        |        +--rw customer-dhcp-servers
|  |        |           +--rw server-ip-address*   inet:ipv6-addres
|  |        +--:(static-addresses)
|  |           +--rw address* [address-id]
|  |              +--rw address-id          string
|  |              +--rw customer-address?   inet:ipv6-address
|  +--rw routing-protocols
|  |  +--rw routing-protocol* [id]
|  |     +--rw id                 string
|  |     +--rw type?              identityref
|  |     +--rw routing-profiles* [id]
|  |     |  +--rw id      routing-profile-reference
|  |     |  +--rw type?   identityref
|  |     +--rw static
|  |     |  +--rw cascaded-lan-prefixes
|  |     |     +--rw ipv4-lan-prefixes* [lan next-hop] {vpn-common:
|  |     |     |  +--rw lan         inet:ipv4-prefix
|  |     |     |  +--rw lan-tag?    string
|  |     |     |  +--rw next-hop    union
|  |     |     |  +--rw status
|  |     |     |     +--rw admin-status
|  |     |     |     |  +--rw status?        identityref
|  |     |     |     |  +--rw last-change?   yang:date-and-time
|  |     |     |     +--ro oper-status
|  |     |     |        +--ro status?        identityref
|  |     |     |        +--ro last-change?   yang:date-and-time
|  |     |     +--rw ipv6-lan-prefixes* [lan next-hop] {vpn-common:
|  |     |        +--rw lan         inet:ipv6-prefix
|  |     |        +--rw lan-tag?    string
|  |     |        +--rw next-hop    union
|  |     |        +--rw status
|  |     |           +--rw admin-status
|  |     |           |  +--rw status?        identityref
|  |     |           |  +--rw last-change?   yang:date-and-time
|  |     |           +--ro oper-status
|  |     |              +--ro status?        identityref
|  |     |              +--ro last-change?   yang:date-and-time
|  |     +--rw bgp
|  |     |  +--rw peer-groups
|  |     |  |  +--rw peer-group* [name]
|  |     |  |     +--rw name              string
|  |     |  |     +--ro local-address?    inet:ip-address
```

```
|  |     |  |     +--rw peer-as           inet:as-number
|  |     |  |     +--rw address-family?   identityref
|  |     |  +--rw neighbor* [remote-address]
|  |     |     +--rw remote-address    inet:ip-address
|  |     |     +--ro local-address?    inet:ip-address
|  |     |     +--rw peer-group?       -> ../../peer-groups/peer-gr
|  |     |     +--rw peer-as           inet:as-number
|  |     |     +--rw address-family?   identityref
|  |     |     +--rw status
|  |     |        +--rw admin-status
|  |     |        |  +--rw status?        identityref
|  |     |        |  +--rw last-change?   yang:date-and-time
|  |     |        +--ro oper-status
|  |     |           +--ro status?        identityref
|  |     |           +--ro last-change?   yang:date-and-time
|  |     +--rw ospf
|  |     |  +--rw address-family?   identityref
|  |     |  +--rw area-id           yang:dotted-quad
|  |     |  +--rw metric?           uint16
|  |     |  +--rw status
|  |     |     +--rw admin-status
|  |     |     |  +--rw status?        identityref
|  |     |     |  +--rw last-change?   yang:date-and-time
|  |     |     +--ro oper-status
|  |     |        +--ro status?        identityref
|  |     |        +--ro last-change?   yang:date-and-time
|  |     +--rw isis
|  |     |  +--rw address-family?   identityref
|  |     |  +--rw area-address      area-address
|  |     |  +--rw status
|  |     |     +--rw admin-status
|  |     |     |  +--rw status?        identityref
|  |     |     |  +--rw last-change?   yang:date-and-time
|  |     |     +--ro oper-status
|  |     |        +--ro status?        identityref
|  |     |        +--ro last-change?   yang:date-and-time
|  |     +--rw rip
|  |     |  +--rw address-family?   identityref
|  |     |  +--rw status
|  |     |     +--rw admin-status
|  |     |     |  +--rw status?        identityref
|  |     |     |  +--rw last-change?   yang:date-and-time
|  |     |     +--ro oper-status
|  |     |        +--ro status?        identityref
|  |     |        +--ro last-change?   yang:date-and-time
|  |     +--rw vrrp
|  |        +--rw address-family?   identityref
|  |        +--rw status
|  |           +--rw admin-status
|  |           |  +--rw status?        identityref
|  |           |  +--rw last-change?   yang:date-and-time
|  |           +--ro oper-status
|  |              +--ro status?        identityref
```

```
|  |              +--ro last-change?   yang:date-and-time
|  +--rw oam
|  |  +--rw bfd {vpn-common:bfd}?
|  |     +--rw holdtime?   uint32
|  |     +--rw status
|  |        +--rw admin-status
|  |        |  +--rw status?        identityref
|  |        |  +--rw last-change?   yang:date-and-time
|  |        +--ro oper-status
|  |           +--ro status?        identityref
|  |           +--ro last-change?   yang:date-and-time
|  +--rw security
|     +--rw encryption {vpn-common:encryption}?
|     |  +--rw enabled?   boolean
|     |  +--rw layer?     enumeration
|     +--rw encryption-profile
|        +--rw (profile)?
|           +--:(customer-profile)
|              +--rw customer-key-chain?   key-chain:key-chain-ref
+--rw ac* [name]
   +--rw customer-name?        string
   +--rw description?          string
   +--rw requested-ac-start?   yang:date-and-time
   +--rw requested-ac-stop?    yang:date-and-time
   +--ro actual-ac-start?      yang:date-and-time
   +--ro actual-ac-stop?       yang:date-and-time
   +--rw peer-sap-id*          string
   +--rw ac-global-profile*    ac-global-profile-reference
   +--rw ac-node-profile*      ac-node-group-reference
   +--rw name                  string
   +--rw l2-connection
   |  +--rw encapsulation
   |  |  +--rw type?              identityref
   |  |  +--rw dot1q
   |  |  |  +--rw tag-type?   identityref
   |  |  |  +--rw cvlan-id?   uint16
   |  |  +--rw priority-tagged
   |  |  |  +--rw tag-type?   identityref
   |  |  +--rw qinq
   |  |     +--rw tag-type?   identityref
   |  |     +--rw svlan-id    uint16
   |  |     +--rw cvlan-id    uint16
   |  +--rw (l2-service)?
   |  |  +--:(l2-tunnel-service)
   |  |  |  +--rw l2-tunnel-service
   |  |  |     +--rw type?         identityref
   |  |  |     +--rw pseudowire
   |  |  |     |  +--rw vcid?      uint32
   |  |  |     |  +--rw far-end?   union
   |  |  |     +--rw vpls
   |  |  |     |  +--rw vcid?      uint32
   |  |  |     |  +--rw far-end*   union
   |  |  |     +--rw vxlan
```

```
|  |  |       +--rw vni-id             uint32
|  |  |       +--rw peer-mode?         identityref
|  |  |       +--rw peer-ip-address*   inet:ip-address
|  |  +--:(l2vpn)
|  |     +--rw l2vpn-id?            vpn-common:vpn-id
|  +--rw bearer-reference?          string {vpn-common:bearer-re
+--rw ip-connection
|  +--rw ipv4 {vpn-common:ipv4}?
|  |  +--rw local-address?                          inet:ipv4-a
|  |  +--rw prefix-length?                          uint8
|  |  +--rw address-allocation-type?                identityref
|  |  +--rw (allocation-type)?
|  |     +--:(dynamic)
|  |     |  +--rw (address-assign)?
|  |     |  |  +--:(number)
|  |     |  |  |  +--rw number-of-dynamic-address?   uint16
|  |     |  |  +--:(explicit)
|  |     |  |     +--rw customer-addresses
|  |     |  |        +--rw address-pool* [pool-id]
|  |     |  |           +--rw pool-id          string
|  |     |  |           +--rw start-address    inet:ipv4-address
|  |     |  |           +--rw end-address?     inet:ipv4-address
|  |     |  +--rw (provider-dhcp)?
|  |     |  |  +--:(dhcp-service-type)
|  |     |  |     +--rw dhcp-service-type?           enumeration
|  |     |  +--rw (dhcp-relay)?
|  |     |     +--:(customer-dhcp-servers)
|  |     |        +--rw customer-dhcp-servers
|  |     |           +--rw server-ip-address*   inet:ipv4-addres
|  |     +--:(static-addresses)
|  |        +--rw address* [address-id]
|  |           +--rw address-id          string
|  |           +--rw customer-address?   inet:ipv4-address
|  +--rw ipv6 {vpn-common:ipv6}?
|     +--rw local-address?                          inet:ipv6-a
|     +--rw prefix-length?                          uint8
|     +--rw address-allocation-type?                identityref
|     +--rw (allocation-type)?
|        +--:(dynamic)
|        |  +--rw (address-assign)?
|        |  |  +--:(number)
|        |  |  |  +--rw number-of-dynamic-address?   uint16
|        |  |  +--:(explicit)
|        |  |     +--rw customer-addresses
|        |  |        +--rw address-pool* [pool-id]
|        |  |           +--rw pool-id          string
|        |  |           +--rw start-address    inet:ipv6-address
|        |  |           +--rw end-address?     inet:ipv6-address
|        |  +--rw (provider-dhcp)?
|        |  |  +--:(dhcp-service-type)
|        |  |     +--rw dhcp-service-type?           enumeration
|        |  +--rw (dhcp-relay)?
|        |     +--:(customer-dhcp-servers)
```

```
|        |        +--rw customer-dhcp-servers
|        |           +--rw server-ip-address*   inet:ipv6-addres
|        +--:(static-addresses)
|           +--rw address* [address-id]
|              +--rw address-id          string
|              +--rw customer-address?   inet:ipv6-address
+--rw routing-protocols
|  +--rw routing-protocol* [id]
|     +--rw id                  string
|     +--rw type?               identityref
|     +--rw routing-profiles* [id]
|     |  +--rw id      routing-profile-reference
|     |  +--rw type?   identityref
|     +--rw static
|     |  +--rw cascaded-lan-prefixes
|     |     +--rw ipv4-lan-prefixes* [lan next-hop] {vpn-common:
|     |     |  +--rw lan         inet:ipv4-prefix
|     |     |  +--rw lan-tag?    string
|     |     |  +--rw next-hop    union
|     |     |  +--rw status
|     |     |     +--rw admin-status
|     |     |     |  +--rw status?        identityref
|     |     |     |  +--rw last-change?   yang:date-and-time
|     |     |     +--ro oper-status
|     |     |        +--ro status?        identityref
|     |     |        +--ro last-change?   yang:date-and-time
|     |     +--rw ipv6-lan-prefixes* [lan next-hop] {vpn-common:
|     |        +--rw lan         inet:ipv6-prefix
|     |        +--rw lan-tag?    string
|     |        +--rw next-hop    union
|     |        +--rw status
|     |           +--rw admin-status
|     |           |  +--rw status?        identityref
|     |           |  +--rw last-change?   yang:date-and-time
|     |           +--ro oper-status
|     |              +--ro status?        identityref
|     |              +--ro last-change?   yang:date-and-time
|     +--rw bgp
|     |  +--rw peer-groups
|     |  |  +--rw peer-group* [name]
|     |  |     +--rw name              string
|     |  |     +--ro local-address?    inet:ip-address
|     |  |     +--rw peer-as           inet:as-number
|     |  |     +--rw address-family?   identityref
|     |  +--rw neighbor* [remote-address]
|     |     +--rw remote-address    inet:ip-address
|     |     +--ro local-address?    inet:ip-address
|     |     +--rw peer-group?       -> ../../peer-groups/peer-gr
|     |     +--rw peer-as           inet:as-number
|     |     +--rw address-family?   identityref
|     |     +--rw status
|     |        +--rw admin-status
|     |        |  +--rw status?        identityref
```

```
|     |        |  +--rw last-change?   yang:date-and-time
|     |        +--ro oper-status
|     |           +--ro status?        identityref
|     |           +--ro last-change?   yang:date-and-time
|     +--rw ospf
|     |  +--rw address-family?   identityref
|     |  +--rw area-id           yang:dotted-quad
|     |  +--rw metric?           uint16
|     |  +--rw status
|     |     +--rw admin-status
|     |     |  +--rw status?        identityref
|     |     |  +--rw last-change?   yang:date-and-time
|     |     +--ro oper-status
|     |        +--ro status?        identityref
|     |        +--ro last-change?   yang:date-and-time
|     +--rw isis
|     |  +--rw address-family?   identityref
|     |  +--rw area-address      area-address
|     |  +--rw status
|     |     +--rw admin-status
|     |     |  +--rw status?        identityref
|     |     |  +--rw last-change?   yang:date-and-time
|     |     +--ro oper-status
|     |        +--ro status?        identityref
|     |        +--ro last-change?   yang:date-and-time
|     +--rw rip
|     |  +--rw address-family?   identityref
|     |  +--rw status
|     |     +--rw admin-status
|     |     |  +--rw status?        identityref
|     |     |  +--rw last-change?   yang:date-and-time
|     |     +--ro oper-status
|     |        +--ro status?        identityref
|     |        +--ro last-change?   yang:date-and-time
|     +--rw vrrp
|        +--rw address-family?   identityref
|        +--rw status
|           +--rw admin-status
|           |  +--rw status?        identityref
|           |  +--rw last-change?   yang:date-and-time
|           +--ro oper-status
|              +--ro status?        identityref
|              +--ro last-change?   yang:date-and-time
+--rw oam
|  +--rw bfd {vpn-common:bfd}?
|     +--rw holdtime?   uint32
|     +--rw status
|        +--rw admin-status
|        |  +--rw status?        identityref
|        |  +--rw last-change?   yang:date-and-time
|        +--ro oper-status
|           +--ro status?        identityref
|           +--ro last-change?   yang:date-and-time
```

```
+--rw security
   +--rw encryption {vpn-common:encryption}?
   |  +--rw enabled?   boolean
   |  +--rw layer?     enumeration
   +--rw encryption-profile
      +--rw (profile)?
         +--:(customer-profile)
            +--rw customer-key-chain?   key-chain:key-chain-ref
```

Figure 19: AC Service Tree Structure

Acknowledgments

TODO acknowledge.

Contributors

TODO contribute.

Authors' Addresses

Mohamed Boucadair (editor)
Orange

Email: mohamed.boucadair@orange.com

Richard Roberts
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