Stream: Internet Engineering Task Force (IETF)
RFC: 9835
Category: Standards Track
Published: August 2025
ISSN: 2070-1721
Authors:
M. Boucadair, Ed. *Orange*
R. Roberts *Juniper*
O. Gonzalez de Dios *Telefonica*
S. Barguil Giraldo *Nokia*
B. Wu *Huawei Technologies*

RFC 9835
A Network YANG Data Model for Attachment Circuits

Abstract

This document specifies a network model for attachment circuits. The model can be used for the provisioning of attachment circuits prior to or during service provisioning (e.g., VPN, Network Slice Service). A companion service model is specified in "YANG Data Models for Bearers and 'Attachment Circuits'-as-a-Service (ACaaS)" (RFC9834).

The module augments the base network ('ietf-network') and the Service Attachment Point (SAP) models with the detailed information for the provisioning of attachment circuits in Provider Edges (PEs).

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9835.

Copyright Notice

Table of Contents

1. Introduction

Connectivity services are provided by networks to customers via dedicated terminating points, such as Service Functions [RFC7665], Customer Edges (CEs), peer Autonomous System Border Routers (ASBRs), data center gateways, or Internet Exchange Points.

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 (AC). For example, the same attachment circuit may host multiple services (e.g., Layer 2 VPN (L2VPN), Layer 3 VPN (L3VPN), or Network Slice Service [RFC9543]). 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 then request a standalone attachment circuit to be put in place and refer to that attachment circuit when requesting services to be bound to that AC. [RFC9834] specifies a data model for managing Attachment Circuits as a Service (ACaaS).

Section 6 specifies a network model for attachment circuits ("ietf-ac-ntw"). The model can be used for the provisioning of ACs in a provider network prior to or during service provisioning. For example, [RFC9836] specifies augmentations to the L2VPN Network Model (L2NM) [RFC9291] and the L3VPN Network Model (L3NM) [RFC9182] to bind LxVPNs to ACs that are provisioned using the procedure defined in this document.

This document leverages [RFC9182] and [RFC9291] by adopting an AC provisioning structure that uses data nodes that are defined in those RFCs. Some refinements were introduced to cover not only conventional service provider networks but also specifics of other target deployments (e.g., cloud network).

The AC network model is designed as augmentations of both the 'ietf-network' model [RFC8345] and the Service Attachment Point (SAP) model [RFC9408]. An attachment circuit can be bound to a single or multiple SAPs. Likewise, the model is designed to accommodate deployments where a SAP can be bound to one or multiple ACs (e.g., a parent AC and its child ACs).

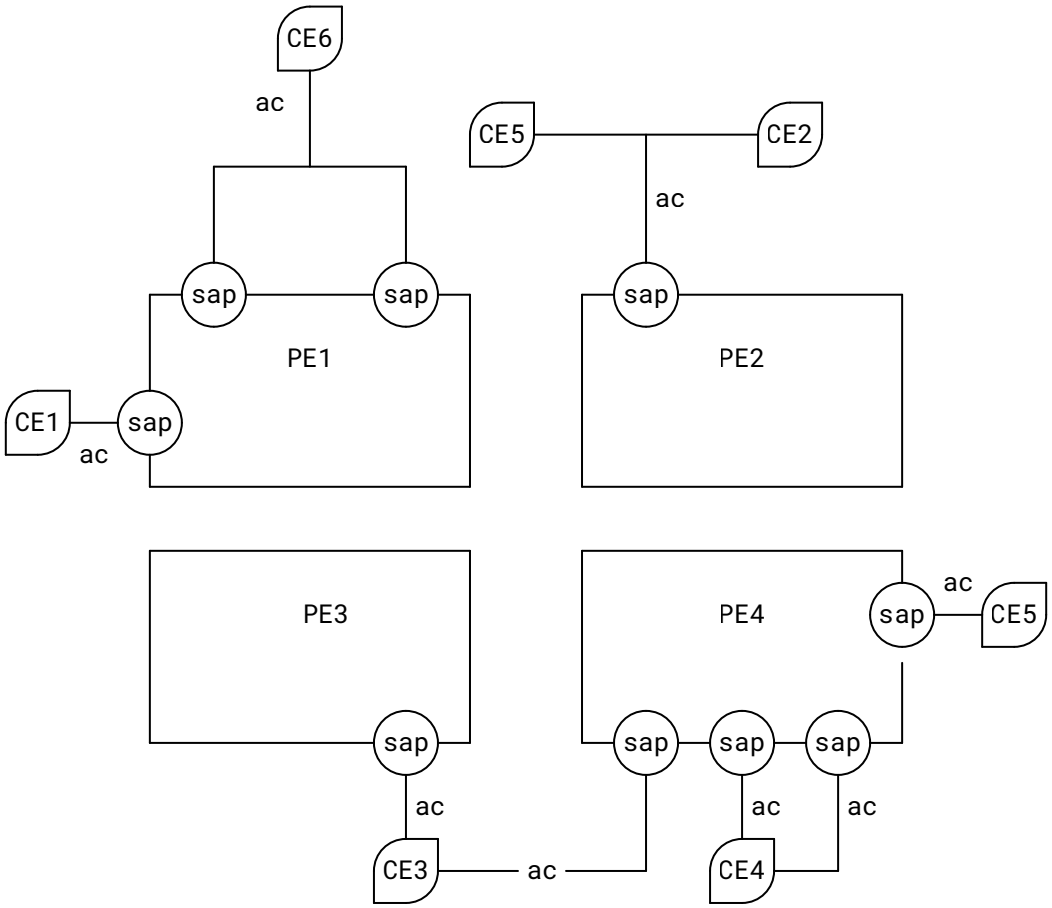

Figure 1: Attachment Circuits Examples

The AC network model uses the AC common model defined in [RFC9833].

The YANG 1.1 [RFC7950] data model in this document conforms to the Network Management Datastore Architecture (NMDA) defined in [RFC8342].

Some examples are provided in Appendix A.

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 reader should be familiar with the terms defined in Section 2 of [RFC9408].

This document uses the term "network model" as defined in Section 2.1 of [RFC8969].

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

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

LxNM refers to both the L2VPN Network Model (L2NM) [RFC9291] and the L3VPN Network Model (L3NM) [RFC9182].

LxVPN refers to both L2VPN and L3VPN.

The following are used in the module prefixes:

ac: Attachment circuit

ntw: Network

sap: Service Attachment Point

svc: Service

In addition, this document uses the following terms:

Bearer: A physical or logical link that connects a customer node (or site) to a provider network.

A bearer can be a wireless or wired link. One or multiple technologies can be used to build a bearer. The bearer type can be specified by a customer.

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

The concept of a bearer can be generalized to refer to the required underlying connection for the provisioning of an attachment circuit.

One or multiple attachment circuits may be hosted over the same bearer (e.g., multiple Virtual Local Area Networks (VLANs) on the same bearer that is provided by a physical link).

Network controller: Denotes a functional entity responsible for the management of the service provider network. One or multiple network controllers can be deployed in a service provider network.

Service orchestrator: Refers to a functional entity that interacts with the customer of a network service.

A service orchestrator is typically responsible for the attachment circuits, the Provider Edge (PE) selection, and requesting the activation of the requested services to a network controller.

A service orchestrator may interact with one or more network controllers.

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

Service provider: An entity that offers network services (e.g., LxVPN or Network Slice Services).

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

Prefix	Module	Reference
ac-common	ietf-ac-common	[RFC9833]
ac-svc	ietf-ac-svc	Section 5.2 of [RFC9834]
dot1q-types	ieee802-dot1q-types	[IEEE802.1Qcp]
if	ietf-interfaces	[RFC8343]
inet	ietf-inet-types	Section 4 of [RFC6991]
key-chain	ietf-key-chain	[RFC8177]
nacm	ietf-netconf-acm	[RFC8341]
nw	ietf-network	[RFC8345]
rt-types	ietf-routing-types	[RFC8294]
rt-pol	ietf-routing-policy	[RFC9067]
sap	ietf-sap-ntw	[RFC9408]
vpn-common	ietf-vpn-common	[RFC9181]

Table 1: Modules and Their Associated Prefixes

3. Relationship to Other AC Data Models

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

- "ietf-ac-common" [RFC9833]
- "ietf-bearer-svc" (Section 6.1 of [RFC9834])
- "ietf-ac-svc" (Section 6.2 of [RFC9834])
- "ietf-ac-ntw" (Section 6)
- "ietf-ac-glue" [RFC9836]

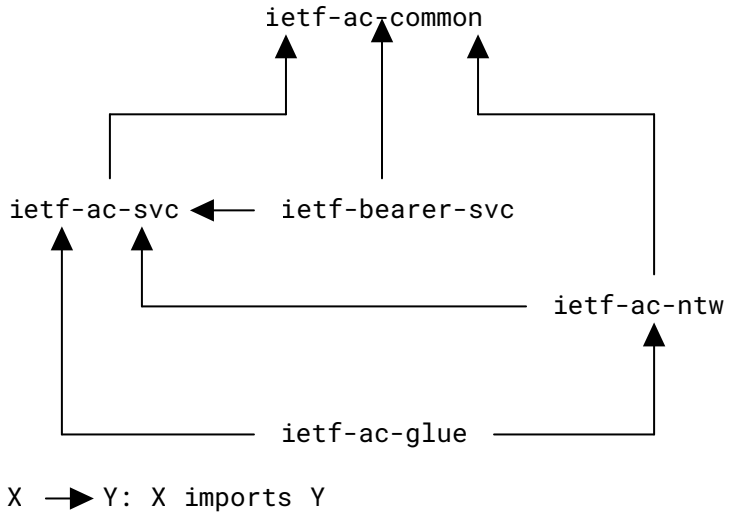

Figure 2: AC Data Models

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

4. Sample Uses of the Attachment Circuit Data Models

4.1. ACs Terminated by One or Multiple Customer Edges (CEs)

Figure 3 depicts a sample target topology that involve ACs:

- ACs are terminated by a SAP at the network side. See Figure 1 for an example of SAPs within a PE.
- A CE can be either a physical device or a logical entity. Such a logical entity is typically a software component (e.g., a virtual Service Function that is hosted within the provider's network or a third-party infrastructure). A CE is seen by the network as a peer SAP [RFC9408].
- CEs may be either dedicated to one single connectivity service or host multiple connectivity services (e.g., CEs with roles of Service Functions [RFC7665]).
- A network provider may bind a single AC to one or multiple peer SAPs (e.g., CE1 and CE2 are tagged as peer SAPs for the same AC). For example, and as discussed in [RFC4364], multiple CEs can be attached to a PE over the same attachment circuit. This scenario is typically implemented when the Layer 2 infrastructure between the CE and the network is a multipoint service.
- A single CE may terminate multiple ACs, which can be associated with the same bearer or distinct bearers (e.g., CE4).
- Customers may request protection schemes in which the ACs associated with their endpoints are terminated by the same PE (e.g., CE3), distinct PEs (e.g., CE4), etc. The network provider uses this request to decide where to terminate the AC in the service provider network and also whether to enable specific capabilities (e.g., Virtual Router Redundancy Protocol (VRRP)).

"ietf-ac-ntw" is a network model that is used to manage the PE side of ACs at a provider network.

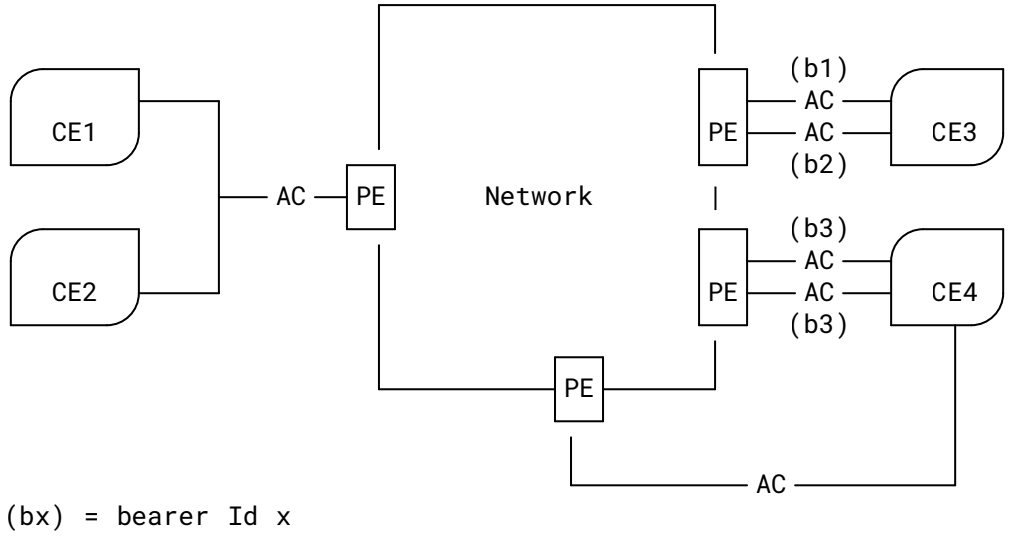

Figure 3: Examples of ACs

4.2. Positioning the AC Network Model in the Overall Service Delivery Process

Figure 4 shows the positioning of the AC network model in the overall service delivery process. The "ietf-ac-ntw" module is a network model that augments the SAP with a comprehensive set of parameters to reflect the attachment circuits that are in place in a network. The model also maintains the mapping with the service references that are used to expose those ACs to customers using the "ietf-ac-svc" module defined in [RFC9834]. Whether the same naming conventions to reference an AC are used in the service and network layers is deployment-specific.

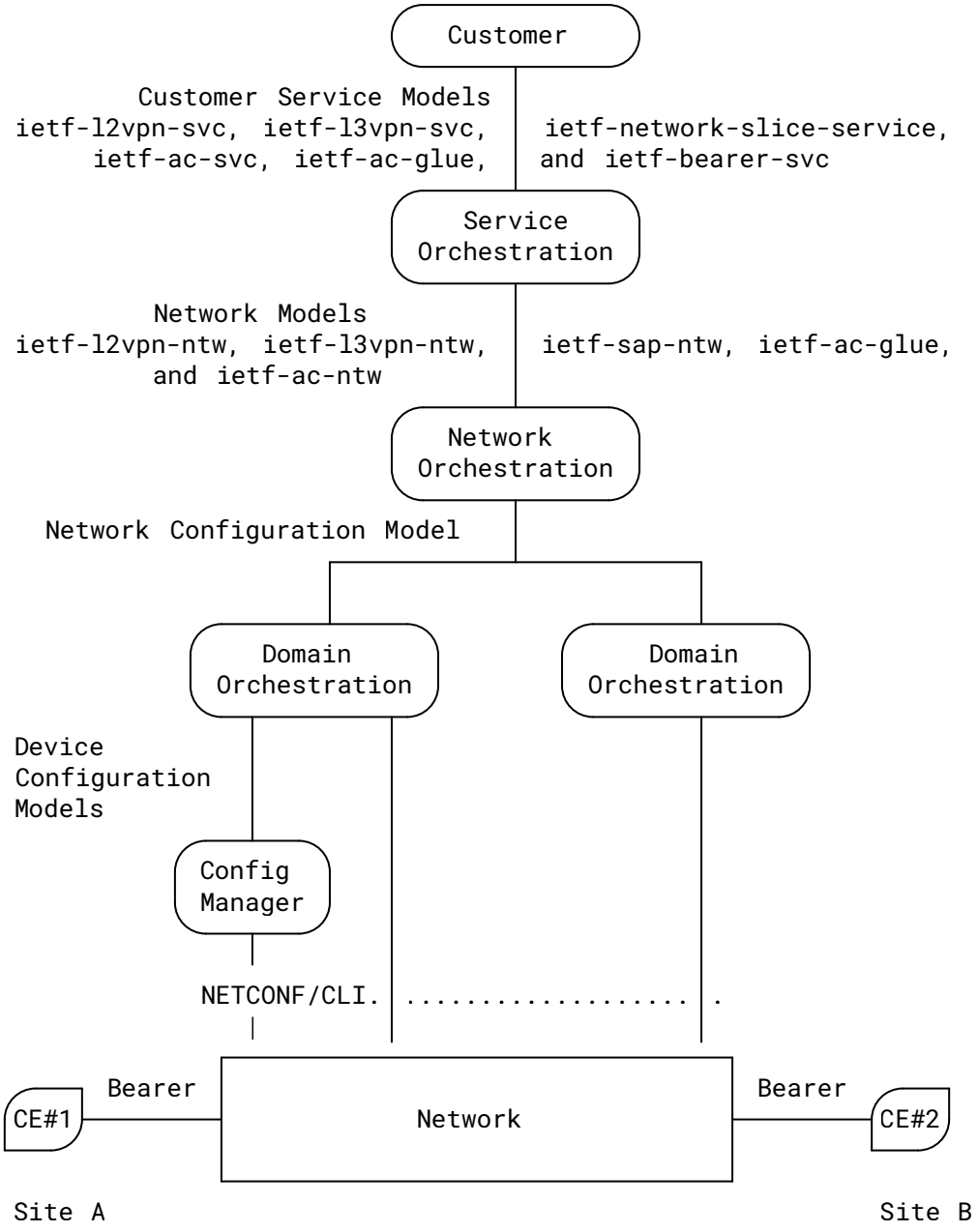

Figure 4: An Example of the Network AC Model Usage

Similar to [RFC9408], the "ietf-ac-ntw" module can be used for both User-to-Network Interface (UNI) and Network-to-Network Interface (NNI). For example, all the ACs shown in Figure 5 have a 'role' set to 'ietf-sap-ntw:nni'. Typically, ASBRs of each network are directly connected to ASBRs of a neighboring network via one or multiple links (bearers). ASBRs of "Network#1" behave as a PE and treat the other adjacent ASBRs as if it were a CE.

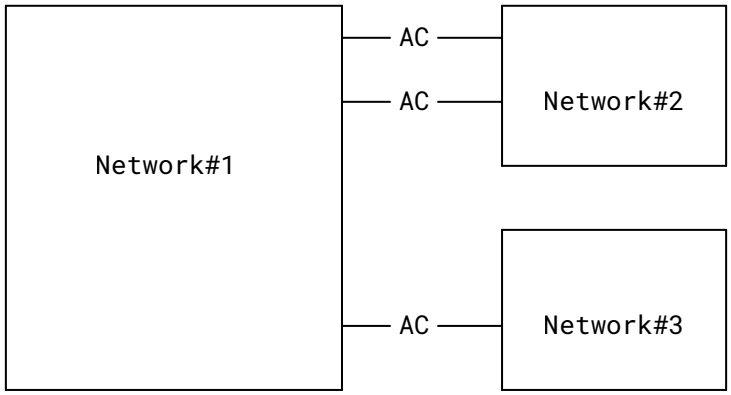

Figure 5: An Example of the Network AC Model Usage Between Provider Networks

5. Description of the Attachment Circuit YANG Module

The full tree diagram of the "ietf-ac-ntw" module is provided in Appendix B. Subtrees are provided in the following subsections for the reader's convenience.

5.1. Overall Structure of the Module

The overall tree structure of the "ietf-ac-ntw" module is shown in Figure 6.

```
augment /nw:networks/nw:network:
  +--rw specific-provisioning-profiles
  |  ...
  +--rw ac-profile* [name]
     ...
augment /nw:networks/nw:network/nw:node:
  +--rw ac* [name]
     +--rw name                string
     +--rw svc-ref?            ac-svc:attachment-circuit-reference
     +--rw profile* [ac-profile-ref]
     |  +--rw ac-profile-ref    leafref
     |  +--rw network-ref?      -> /nw:networks/network/network-id
     +--rw parent-ref
     |  +--rw ac-ref?         leafref
     |  +--rw node-ref?       leafref
     |  +--rw network-ref?    -> /nw:networks/network/network-id
     +--ro child-ref
     |  +--ro ac-ref*         leafref
     |  +--ro node-ref?       leafref
     |  +--ro network-ref?    -> /nw:networks/network/network-id
     +--rw peer-sap-id*          string
     +--rw group* [group-id]
     |  +--rw group-id       string
     |  +--rw precedence?    identityref
     +--rw status
     |  +--rw admin-status
     |  |  +--rw status?         identityref
     |  |  +--ro last-change?    yang:date-and-time
     |  +--ro oper-status
     |     +--ro status?         identityref
     |     +--ro last-change?    yang:date-and-time
     +--rw description?          string
     +--rw l2-connection  {ac-common:layer2-ac}?
     |  ...
     +--rw ip-connection  {ac-common:layer3-ac}?
     |  ...
     +--rw routing-protocols
     |  ...
     +--rw oam
     |  ...
     +--rw security
     |  ...
     +--rw service
        ...
  augment /nw:networks/nw:network/nw:node/sap:service/sap:sap:
    +--rw ac* [ac-ref]
       +--rw ac-ref          leafref
       +--rw node-ref?       leafref
       +--rw network-ref?    -> /nw:networks/network/network-id
```

Figure 6: Overall Tree Structure

A node can host one or more SAPs. Per [RFC9408], a SAP is an abstraction of the network reference point (the PE side of an AC, in the context of this document) where network services can be and/or are delivered to customers. Each SAP terminates one or multiple ACs. In turn, each

AC may be terminated by one or more peer SAPs ('peer-sap'). In order to expose such AC/SAP binding information, the SAP model [RFC9408] is augmented with the required AC-related information.

Unlike the AC service model [RFC9834], an AC is uniquely identified by a name within the scope of a node, not a network. A textual description of the AC may be provided ('description').

Also, in order to ease the correlation between the AC exposed at the service layer and the AC that is actually provisioned in the network operation, a reference to the AC exposed to the customer ('svc-ref') is stored in the "ietf-ac-ntw" module.

ACs that are terminated by a SAP are listed in the 'ac' container under '/nw:networks/nw:network/nw:node/sap:service/sap:sap'. A controller may indicate a filter based on the service type (e.g., Network Slice or L3VPN) to retrieve the list of available SAPs, and thus ACs, for that service.

In order to factorize common data that is provisioned for a group of ACs, a set of profiles (Section 5.3) can be defined at the network level and then called under the node level. The information contained in a profile is thus inherited, unless the corresponding data node is refined at the AC level. In such a case, the value provided at the AC level takes precedence over the global one.

In contexts where the same AC is terminated by multiple peer SAPs (e.g., an AC with multiple CEs) but a subset of them have specific information, the module allows operators to:

- Define a parent AC that may list all these CEs as peer SAPs.
- Create individual ACs that are bound to the parent AC using 'parent-ref'.
- Indicate for each individual AC one or a subset of the CEs as peer SAPs. All these individual ACs will inherit the properties of the parent AC.

Whenever a parent AC is deleted, then all child ACs of that AC **MUST** be deleted. Child ACs are referenced using 'child-ref'.

An AC may belong to one or multiple groups [RFC9181]. For example, the 'group-id' is used to associate redundancy or protection constraints with ACs.

The status of an AC can be tracked using 'status'. Both operational status and administrative status are maintained. A mismatch between the administrative status vs. the operational status can be used as a trigger to detect anomalies.

An AC can be characterized using Layer 2 connectivity (Section 5.4), Layer 3 connectivity (Section 5.5), routing protocols (Section 5.6), Operations, Administration, and Maintenance (OAM) (Section 5.7), security (Section 5.8), and service (Section 5.9) considerations. Features are used to tag conditional portions to accommodate various deployments (support of Layer 2 ACs, Layer 3 ACs, IPv4, IPv6, routing protocols, Bidirectional Forwarding Detection (BFD), etc.).

5.2. References

The AC network module defines a set of groupings depicted in Figure 7 for referencing purposes. These references are used within or outside the AC network module. The use of such groupings is consistent with the design in [RFC8345].

```
grouping attachment-circuit-reference:
  +-- ac-ref?         leafref
  +-- node-ref?       leafref
  +-- network-ref?    -> /nw:networks/network/network-id
grouping attachment-circuit-references:
  +-- ac-ref*         leafref
  +-- node-ref?       leafref
  +-- network-ref?    -> /nw:networks/network/network-id
grouping ac-profile-reference:
  +-- ac-profile-ref?   leafref
  +-- network-ref?      -> /nw:networks/network/network-id
grouping encryption-profile-reference:
  +-- encryption-profile-ref?   leafref
  +-- network-ref?              -> /nw:networks/network/network-id
grouping qos-profile-reference:
  +-- qos-profile-ref?   leafref
  +-- network-ref?       -> /nw:networks/network/network-id
grouping failure-detection-profile-reference:
  +-- failure-detection-profile-ref?   leafref
  +-- network-ref?       -> /nw:networks/network/network-id
grouping forwarding-profile-reference:
  +-- forwarding-profile-ref?   leafref
  +-- network-ref?              -> /nw:networks/network/network-id
grouping routing-profile-reference:
  +-- routing-profile-ref?   leafref
  +-- network-ref?           -> /nw:networks/network/network-id
```

Figure 7: References Groupings

The groupings shown in Figure 7 contain the information necessary to reference:

- an attachment circuit that is terminated by a specific node in a given network,
- an attachment circuit profile of a specific network (Section 5.3), and
- specific provisioning profiles that are bound to a specific network (Section 5.3).

5.3. Provisioning Profiles

The AC and specific provisioning profiles tree structure is shown in Figure 8.

```
augment /nw:networks/nw:network:
  +--rw specific-provisioning-profiles
  |  +--rw valid-provider-identifiers
  |     +--rw encryption-profile-identifier* [id]
  |     |  +--rw id    string
  |     +--rw qos-profile-identifier* [id]
  |     |  +--rw id    string
  |     +--rw failure-detection-profile-identifier* [id]
  |     |  +--rw id    string
  |     +--rw forwarding-profile-identifier* [id]
  |     |  +--rw id    string
  |     +--rw routing-profile-identifier* [id]
  |        +--rw id    string
  +--rw ac-profile* [name]
     +--rw name                  string
     +--rw routing-protocols
     |  +--rw routing-protocol* [id]
     |     +--rw id      string
     |     +--rw type?   identityref
     |     +--rw bgp {vpn-common:rtg-bgp}?
     |     |  +--rw peer-groups
     |     |     +--rw peer-group* [name]
     |     |        +--rw name                        string
     |     |        +--rw description?                string
     |     |        +--rw apply-policy
     |     |        |  +--rw import-policy*             leafref
     |     |        |  +--rw default-import-policy?
     |     |        |  |       default-policy-type
     |     |        |  +--rw export-policy*             leafref
     |     |        |  +--rw default-export-policy?
     |     |        |          default-policy-type
     |     |        +--rw local-as?                   inet:as-number
     |     |        +--rw peer-as                     inet:as-number
     |     |        +--rw address-family?             identityref
     |     |        +--rw role?                       identityref
     |     |        +--rw multihop?                   uint8
     |     |        +--rw as-override?                boolean
     |     |        +--rw allow-own-as?               uint8
     |     |        +--rw prepend-global-as?          boolean
     |     |        +--rw send-default-route?         boolean
     |     |        +--rw site-of-origin?
     |     |        |       rt-types:route-origin
     |     |        +--rw ipv6-site-of-origin?
     |     |        |       rt-types:ipv6-route-origin
     |     |        +--rw redistribute-connected* [address-family]
     |     |        |  +--rw address-family    identityref
     |     |        |  +--rw enabled?          boolean
     |     |        +--rw bgp-max-prefix
     |     |        |  +--rw max-prefix?          uint32
     |     |        |  +--rw warning-threshold?   decimal64
     |     |        |  +--rw violate-action?      enumeration
     |     |        |  +--rw restart-timer?       uint32
     |     |        +--rw bgp-timers
     |     |           +--rw keepalive?   uint16
     |     |           +--rw hold-time?   uint16
     |     +--rw ospf {vpn-common:rtg-ospf}?
     |     |  +--rw address-family?   identityref
```

```
|      |  +--rw area-id           yang:dotted-quad
|      |  +--rw metric?           uint16
|      |  +--rw max-lsa?          uint32
|      |  +--rw passive?          boolean
|      +--rw isis {vpn-common:rtg-isis}?
|      |  +--rw address-family?   identityref
|      |  +--rw area-address      area-address
|      |  +--rw level?            identityref
|      |  +--rw metric?           uint32
|      |  +--rw passive?          boolean
|      +--rw rip {vpn-common:rtg-rip}?
|      |  +--rw address-family?   identityref
|      |  +--rw timers
|      |  |  +--rw update-interval?     uint16
|      |  |  +--rw invalid-interval?    uint16
|      |  |  +--rw holddown-interval?   uint16
|      |  |  +--rw flush-interval?      uint16
|      |  +--rw default-metric?   uint8
|      +--rw vrrp {vpn-common:rtg-vrrp}?
|         +--rw address-family?   identityref
|         +--rw ping-reply?       boolean
+--rw oam
   +--rw bfd {vpn-common:bfd}?
      +--rw session-type?               identityref
      +--rw desired-min-tx-interval?    uint32
      +--rw required-min-rx-interval?   uint32
      +--rw local-multiplier?           uint8
      +--rw holdtime?                   uint32
```

Figure 8: Profiles Tree Structure

Similar to [RFC9182] and [RFC9291], the exact definition of the specific provisioning profiles is local to each service provider. The model only includes an identifier for these profiles in order to ease identifying and binding local policies when building an AC. As shown in Figure 8, the following identifiers can be included:

'encryption-profile-identifier': An encryption profile refers to a set of policies related to the encryption schemes and setup that can be applied on the AC. See also Section 5.8.

'qos-profile-identifier': A Quality of Service (QoS) profile refers to a set of policies such as classification, marking, and actions (e.g., [RFC3644]). See also Section 5.9.

'failure-detection-profile-identifier': A failure detection profile refers to a set of failure detection policies such as Bidirectional Forwarding Detection (BFD) policies [RFC5880] that can be invoked when building an AC. Such a profile can be, for example, referenced in static routes (Section 5.6.1) or under the OAM level (Section 5.7). The use of this profile is similar to the detailed examples depicted in Appendices A.11.3 and A.12 of [RFC9834].

'forwarding-profile-identifier': A forwarding profile refers to the policies that apply to the forwarding of packets conveyed over an AC. Such policies may consist of, for example, applying Access Control Lists (ACLs) as in Section 5.9.

'routing-profile-identifier': A routing profile refers to a set of routing policies that will be invoked (e.g., BGP policies) for an AC. Refer to Section 5.6.

The 'ac-profile' defines parameters that can be factorized among a set of ACs. Each profile is identified by a 'name' that is unique in a network. Some of the data nodes can be adjusted at the node level. These adjusted values take precedence over the values in the profile.

5.4. L2 Connection

The 'l2-connection' container is used to manage the Layer 2 properties of an AC (mainly, the PE side of an AC). The Layer 2 connection tree structure is shown in Figure 9.

```
augment /nw:networks/nw:network/nw:node:
  +--rw ac* [name]
     +--rw name                 string
     + ...
     +--rw l2-connection  {ac-common:layer2-ac}?
     |  +--rw encapsulation
     |  |  +--rw encap-type?        identityref
     |  |  +--rw dot1q
     |  |  |  +--rw tag-type?         identityref
     |  |  |  +--rw cvlan-id?         uint16
     |  |  |  +--rw tag-operations
     |  |  |     +--rw (op-choice)?
     |  |  |     |  +--:(pop)
     |  |  |     |  |  +--rw pop?         empty
     |  |  |     |  +--:(push)
     |  |  |     |  |  +--rw push?        empty
     |  |  |     |  +--:(translate)
     |  |  |     |     +--rw translate?   empty
     |  |  |     +--rw tag-1?             dot1q-types:vlanid
     |  |  |     +--rw tag-1-type?
     |  |  |     |       dot1q-types:dot1q-tag-type
     |  |  |     +--rw tag-2?             dot1q-types:vlanid
     |  |  |     +--rw tag-2-type?
     |  |  |             dot1q-types:dot1q-tag-type
     |  |  +--rw priority-tagged
     |  |  |  +--rw tag-type?   identityref
     |  |  +--rw qinq
     |  |     +--rw tag-type?         identityref
     |  |     +--rw svlan-id?         uint16
     |  |     +--rw cvlan-id?         uint16
     |  |     +--rw tag-operations
     |  |        +--rw (op-choice)?
     |  |        |  +--:(pop)
     |  |        |  |  +--rw pop?         uint8
     |  |        |  +--:(push)
     |  |        |  |  +--rw push?        empty
     |  |        |  +--:(translate)
     |  |        |     +--rw translate?   uint8
     |  |        +--rw tag-1?             dot1q-types:vlanid
     |  |        +--rw tag-1-type?
     |  |        |       dot1q-types:dot1q-tag-type
     |  |        +--rw tag-2?             dot1q-types:vlanid
     |  |        +--rw tag-2-type?
     |  |                dot1q-types:dot1q-tag-type
     |  +--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 l2-termination-point?      string
|  +--rw local-bridge-reference?    string
|  +--rw bearer-reference?          string
|  |       {ac-common:server-assigned-reference}?
|  +--rw lag-interface {vpn-common:lag-interface}?
|     +--rw lag-interface-id?   string
|     +--rw member-link-list
|        +--rw member-link* [name]
|           +--rw name    string
+--rw ip-connection  {ac-common:layer3-ac}?
|  ...
+--rw routing-protocols
|  ...
+--rw oam
|  ...
+--rw security
|  ...
+--rw service
   ...
```

Figure 9: Layer 2 Connection Tree Structure

The 'encapsulation' container specifies the Layer 2 encapsulation to use (if any) and allows the configuration of the relevant tags. Also, the model supports tag manipulation operations (e.g., tag rewrite).

The 'l2-tunnel-service' container is used to specify the required parameters to set a Layer 2 tunneling service (e.g., a Virtual Private LAN Service (VPLS), a Virtual eXtensible Local Area Network (VXLAN), or a pseudowire (Section 6.1 of [RFC8077])). 'l2vpn-id' is used to identify a L2VPN service that is associated with an Integrated Routing and Bridging (IRB) interface.

Specific Layer 2 sub-interfaces may be required to be configured in some implementations/ deployments. Such a Layer-2-specific interface can be included in 'l2-termination-point'.

To accommodate implementations that require internal bridging, a local bridge reference can be specified in 'local-bridge-reference'. Such a reference may be a local bridge domain.

A reference to the bearer used by this AC is maintained using 'bearer-reference'.

5.5. IP Connection

This 'ip-connection' container is used to group Layer 3 connectivity information, particularly the IP addressing information, of an AC.

The Layer 3 connection tree structure is shown in Figure 10.

```
augment /nw:networks/nw:network/nw:node:
  +--rw ac* [name]
     +--rw name                 string
     + ...
     +--rw l2-connection  {ac-common:layer2-ac}?
     |  ...
     +--rw ip-connection  {ac-common:layer3-ac}?
     |  +--rw l3-termination-point?   string
     |  +--rw ipv4 {vpn-common:ipv4}?
     |  |  +--rw local-address?
     |  |  |       inet:ipv4-address
     |  |  +--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
     |  |     |  |  +--:(service-type)
     |  |     |  |     +--rw (service-type)?
     |  |     |  |        +--:(relay)
     |  |     |  |           +--rw server-ip-address*
     |  |     |  |                   inet:ipv4-address
     |  |     |  +--rw (dhcp-relay)?
     |  |     |     +--:(customer-dhcp-servers)
     |  |     |        +--rw customer-dhcp-servers
     |  |     |           +--rw server-ip-address*
     |  |     |                   inet:ipv4-address
     |  |     +--:(static-addresses)
     |  |        +--rw address* [address-id]
     |  |           +--rw address-id                      string
     |  |           +--rw customer-address?
     |  |           |       inet:ipv4-address
     |  |           +--rw failure-detection-profile-ref?   leafref
     |  |           +--rw network-ref?
     |  |                   -> /nw:networks/network/network-id
     |  +--rw ipv6 {vpn-common:ipv6}?
     |     +--rw local-address?
     |     |       inet:ipv6-address
     |     +--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
|        |  |  +--:(service-type)
|        |  |     +--rw (service-type)?
|        |  |        +--:(relay)
|        |  |           +--rw server-ip-address*
|        |  |                   inet:ipv6-address
|        |  +--rw (dhcp-relay)?
|        |     +--:(customer-dhcp-servers)
|        |        +--rw customer-dhcp-servers
|        |           +--rw server-ip-address*
|        |                   inet:ipv6-address
|        +--:(static-addresses)
|           +--rw address* [address-id]
|              +--rw address-id                       string
|              +--rw customer-address?
|              |       inet:ipv6-address
|              +--rw failure-detection-profile-ref?   leafref
|              +--rw network-ref?
|                      -> /nw:networks/network/network-id
+--rw routing-protocols
|  ...
+--rw oam
|  ...
+--rw security
|  ...
+--rw service
   ...
```

Figure 10: IP Connection Tree Structure

A distinct Layer 3 interface other than the interface indicated under the 'l2-connection' container may be needed to terminate the Layer 3 connectivity. The identifier of such an interface is included in 'l3-termination-point'. For example, this data node can be used to carry the identifier of a bridge domain interface.

This container can include IPv4, IPv6, or both if dual-stack is enabled. For both IPv4 and IPv6, the IP connection supports three IP address assignment modes for customer addresses: provider DHCP, DHCP relay, and static addressing. Note that for the IPv6 case, Stateless Address Autoconfiguration (SLAAC) [RFC4862] can be used.

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

For IPv6, if 'address-allocation-type' is set to 'slaac', the Prefix Information option of Router Advertisements that will be issued for SLAAC purposes will carry the IPv6 prefix that is determined by 'local-address' and 'prefix-length'. For example, if 'local-address' is set to '2001:db8:0:1::1' and 'prefix-length' is set to '64', the IPv6 prefix that will be used is '2001:db8:0:1::/64'.

In some deployment contexts (e.g., network merging), multiple IP subnets may be used in a transition period. For such deployments, multiple ACs (typically, two) with overlapping information may be maintained during a transition period. The correlation between these ACs may rely upon the same 'svc-ref'.

5.6. Routing

The overall routing subtree structure is shown in Figure 11.

```
module: ietf-ac-ntw
  augment /nw:networks/nw:network:
    +--rw ac-profile* [name]
       +--rw name                 string
       +--rw routing-protocols
       |  +--rw routing-protocol* [id]
       |     +--rw id      string
       |     +--rw type?   identityref
       |     +--rw bgp  {vpn-common:rtg-bgp}?
       |     |  ...
       |     +--rw ospf  {vpn-common:rtg-ospf}?
       |     |  ...
       |     +--rw isis  {vpn-common:rtg-isis}?
       |     |  ...
       |     +--rw rip  {vpn-common:rtg-rip}?
       |     |  ...
       |     +--rw vrrp  {vpn-common:rtg-vrrp}?
       |        ...
       +--rw oam
          ...
  augment /nw:networks/nw:network/nw:node:
    +--rw ac* [name]
       +--rw name                 string
       ...
       +--rw l2-connection  {ac-common:layer2-ac}?
       |  ...
       +--rw ip-connection  {ac-common:layer3-ac}?
       |  ...
       +--rw routing-protocols
       |  +--rw routing-protocol* [id]
       |     +--rw id                  string
       |     +--rw type?               identityref
       |     +--rw routing-profile* [routing-profile-ref]
       |     |  +--rw routing-profile-ref    leafref
       |     |  +--rw network-ref?
       |     |  |       -> /nw:networks/network/network-id
       |     |  +--rw type?                  identityref
       |     +--rw static
       |     |  ...
       |     +--rw bgp  {vpn-common:rtg-bgp}?
       |     |  ...
       |     +--rw ospf  {vpn-common:rtg-ospf}?
       |     |  ...
       |     +--rw isis  {vpn-common:rtg-isis}?
       |     |  ...
       |     +--rw rip  {vpn-common:rtg-rip}?
       |     |  ...
       |     +--rw vrrp  {vpn-common:rtg-vrrp}?
       |        ...
       +--rw oam
       |  ...
       +--rw security
       |  ...
```

```
        +--rw service
           ...
```

Figure 11: Routing Tree Structure

Multiple routing instances ('routing-protocol') can be defined, each uniquely identified by an 'id'. Specifically, each instance is uniquely identified to accommodate scenarios where multiple instances of the same routing protocol have to be configured on the same AC.

The type of a routing instance is indicated in 'type'. The values of this attribute are those defined in [RFC9181] (the 'routing-protocol-type' identity). Specific data nodes are then provided as a function of the 'type'. See more details in the following subsections.

One or multiple routing profiles ('routing-profile') can be provided for a given routing instance.

5.6.1. Static Routing

The static routing subtree structure is shown in Figure 12.

```
module: ietf-ac-ntw
          ...
  augment /nw:networks/nw:network/nw:node:
    +--rw ac* [name]
       +--rw name                 string
       ...
       +--rw l2-connection  {ac-common:layer2-ac}?
       |  ...
       +--rw ip-connection  {ac-common:layer3-ac}?
       |  ...
       +--rw routing-protocols
       |  +--rw routing-protocol* [id]
       |     +--rw id                 string
       |     +--rw type?              identityref
       |     +--rw routing-profile* [routing-profile-ref]
       |     |  +--rw routing-profile-ref    leafref
       |     |  +--rw network-ref?
       |     |  |       -> /nw:networks/network/network-id
       |     |  +--rw type?                  identityref
       |     +--rw static
       |     |  +--rw cascaded-lan-prefixes
       |     |     +--rw ipv4-lan-prefix* [lan next-hop]
       |     |     |       {vpn-common:ipv4}?
       |     |     |  +--rw lan           inet:ipv4-prefix
       |     |     |  +--rw lan-tag?      string
       |     |     |  +--rw next-hop      union
       |     |     |  +--rw metric?       uint32
       |     |     |  +--rw bfd {vpn-common:bfd}?
       |     |     |  |  +--rw enabled?
       |     |     |  |  |       boolean
       |     |     |  |  +--rw failure-detection-profile-ref?
       |     |     |  |  |       leafref
       |     |     |  |  +--rw network-ref?
       |     |     |  |          -> /nw:networks/network/network-id
       |     |     |  +--rw preference?   uint32
       |     |     |  +--rw status
       |     |     |     +--rw admin-status
       |     |     |     |  +--rw status?        identityref
       |     |     |     |  +--ro last-change?   yang:date-and-time
       |     |     |     +--ro oper-status
       |     |     |        +--ro status?        identityref
       |     |     |        +--ro last-change?   yang:date-and-time
       |     |     +--rw ipv6-lan-prefix* [lan next-hop]
       |     |             {vpn-common:ipv6}?
       |     |        +--rw lan           inet:ipv6-prefix
       |     |        +--rw lan-tag?      string
       |     |        +--rw next-hop      union
       |     |        +--rw metric?       uint32
       |     |        +--rw bfd {vpn-common:bfd}?
       |     |        |  +--rw enabled?
       |     |        |  |       boolean
       |     |        |  +--rw failure-detection-profile-ref?
       |     |        |  |       leafref
       |     |        |  +--rw network-ref?
       |     |        |          -> /nw:networks/network/network-id
       |     |        +--rw preference?   uint32
       |     |        +--rw status
```

```
|     |           +--rw admin-status
|     |           |  +--rw status?        identityref
|     |           |  +--ro last-change?   yang:date-and-time
|     |           +--ro oper-status
|     |              +--ro status?        identityref
|     |              +--ro last-change?   yang:date-and-time
|     +--rw bgp  {vpn-common:rtg-bgp}?
|     |  ...
|     +--rw ospf  {vpn-common:rtg-ospf}?
|     |  ...
|     +--rw isis  {vpn-common:rtg-isis}?
|     |  ...
|     +--rw rip  {vpn-common:rtg-rip}?
|     |  ...
|     +--rw vrrp  {vpn-common:rtg-vrrp}?
|        ...
+--rw oam
|  ...
+--rw security
|  ...
+--rw service
   ...
```

Figure 12: Static Routing Tree Structure

The following data nodes can be defined for a given IP prefix:

'lan-tag': Indicates a local tag (e.g., 'myfavorite-lan') that is used to enforce local policies.

'next-hop': Indicates the next hop to be used for the static route.

It can be identified by an IP address, a predefined next-hop type (e.g., 'discard' or 'local-link'), etc.

'bfd': Indicates whether BFD is enabled or disabled for this static route entry. A BFD profile may also be provided.

'metric': Indicates the metric associated with the static route entry. This metric is used when the route is exported into an IGP.

'preference': Indicates the preference associated with the static route entry.

This preference is used to select a preferred route among routes to the same destination prefix.

'status': Used to convey the status of a static route entry. This data node can also be used to control the (de)activation of individual static route entries.

5.6.2. BGP

The BGP routing subtree structure is shown in Figure 13.

```
module: ietf-ac-ntw
  augment /nw:networks/nw:network:
    +--rw ac-profile* [name]
       +--rw name                 string
       +--rw routing-protocols
       |  +--rw routing-protocol* [id]
       |     +--rw id      string
       |     +--rw type?   identityref
       |     +--rw bgp  {vpn-common:rtg-bgp}?
       |     |  +--rw peer-groups
       |     |     +--rw peer-group* [name]
       |     |        +--rw name                       string
       |     |        +--rw description?               string
       |     |        +--rw apply-policy
       |     |        |  +--rw import-policy*            leafref
       |     |        |  +--rw default-import-policy?
       |     |        |  |       default-policy-type
       |     |        |  +--rw export-policy*            leafref
       |     |        |  +--rw default-export-policy?
       |     |        |          default-policy-type
       |     |        +--rw local-as?                  inet:as-number
       |     |        +--rw peer-as                    inet:as-number
       |     |        +--rw address-family?            identityref
       |     |        +--rw role?                      identityref
       |     |        +--rw multihop?                  uint8
       |     |        +--rw as-override?               boolean
       |     |        +--rw allow-own-as?              uint8
       |     |        +--rw prepend-global-as?         boolean
       |     |        +--rw send-default-route?        boolean
       |     |        +--rw site-of-origin?
       |     |        |       rt-types:route-origin
       |     |        +--rw ipv6-site-of-origin?
       |     |        |       rt-types:ipv6-route-origin
       |     |        +--rw redistribute-connected* [address-family]
       |     |        |  +--rw address-family    identityref
       |     |        |  +--rw enabled?          boolean
       |     |        +--rw bgp-max-prefix
       |     |        |  +--rw max-prefix?          uint32
       |     |        |  +--rw warning-threshold?   decimal64
       |     |        |  +--rw violate-action?      enumeration
       |     |        |  +--rw restart-timer?       uint32
       |     |        +--rw bgp-timers
       |     |           +--rw keepalive?   uint16
       |     |           +--rw hold-time?   uint16
       |     +--rw ospf  {vpn-common:rtg-ospf}?
       |     |  ...
       |     +--rw isis  {vpn-common:rtg-isis}?
       |     |  ...
       |     +--rw rip  {vpn-common:rtg-rip}?
       |     |  ...
       |     +--rw vrrp  {vpn-common:rtg-vrrp}?
       |        ...
       +--rw oam
          ...
  augment /nw:networks/nw:network/nw:node:
    +--rw ac* [name]
       +--rw name                 string
```

```
        ...
        +--rw l2-connection  {ac-common:layer2-ac}?
        |  ...
        +--rw ip-connection  {ac-common:layer3-ac}?
        |  ...
        +--rw routing-protocols
        |  +--rw routing-protocol* [id]
        |     +--rw id                  string
        |     +--rw type?               identityref
        |     +--rw routing-profile* [routing-profile-ref]
        |     |  +--rw routing-profile-ref    leafref
        |     |  +--rw network-ref?
        |     |  |       -> /nw:networks/network/network-id
        |     |  +--rw type?                  identityref
        |     +--rw static
        |     |  ...
        |     +--rw bgp {vpn-common:rtg-bgp}?
        |     |  +--rw peer-groups
        |     |  |  +--rw peer-group* [name]
        |     |  |     +--rw name                       string
        |     |  |     +--rw local-address?             union
        |     |  |     +--rw description?               string
        |     |  |     +--rw apply-policy
        |     |  |     |  +--rw import-policy*            leafref
        |     |  |     |  +--rw default-import-policy?
        |     |  |     |  |       default-policy-type
        |     |  |     |  +--rw export-policy*            leafref
        |     |  |     |  +--rw default-export-policy?
        |     |  |     |          default-policy-type
        |     |  |     +--rw local-as?                  inet:as-number
        |     |  |     +--rw peer-as                    inet:as-number
        |     |  |     +--rw address-family?            identityref
        |     |  |     +--rw role?                      identityref
        |     |  |     +--rw multihop?                  uint8
        |     |  |     +--rw as-override?               boolean
        |     |  |     +--rw allow-own-as?              uint8
        |     |  |     +--rw prepend-global-as?         boolean
        |     |  |     +--rw send-default-route?        boolean
        |     |  |     +--rw site-of-origin?
        |     |  |     |       rt-types:route-origin
        |     |  |     +--rw ipv6-site-of-origin?
        |     |  |     |       rt-types:ipv6-route-origin
        |     |  |     +--rw redistribute-connected* [address-family]
        |     |  |     |  +--rw address-family    identityref
        |     |  |     |  +--rw enabled?          boolean
        |     |  |     +--rw bgp-max-prefix
        |     |  |     |  +--rw max-prefix?          uint32
        |     |  |     |  +--rw warning-threshold?   decimal64
        |     |  |     |  +--rw violate-action?      enumeration
        |     |  |     |  +--rw restart-timer?       uint32
        |     |  |     +--rw bgp-timers
        |     |  |     |  +--rw keepalive?   uint16
        |     |  |     |  +--rw hold-time?   uint16
        |     |  |     +--rw authentication
        |     |  |        +--rw enabled?           boolean
        |     |  |        +--rw keying-material
        |     |  |           +--rw (option)?
        |     |  |              +--:(ao)
```

```
        |      |  |              |  +--rw enable-ao?          boolean
        |      |  |              |  +--rw ao-keychain?
        |      |  |              |          key-chain:key-chain-ref
        |      |  |              +--:(md5)
        |      |  |              |  +--rw md5-keychain?
        |      |  |              |          key-chain:key-chain-ref
        |      |  |              +--:(explicit)
        |      |  |                 +--rw key-id?             uint32
        |      |  |                 +--rw key?                string
        |      |  |                 +--rw crypto-algorithm?
        |      |  |                         identityref
        |      |  +--rw neighbor* [remote-address]
        |      |     +--rw remote-address            inet:ip-address
        |      |     +--rw local-address?            union
        |      |     +--rw peer-group?
        |      |     |       -> ../../peer-groups/peer-group/name
        |      |     +--rw description?              string
        |      |     +--rw apply-policy
        |      |     |  +--rw import-policy*           leafref
        |      |     |  +--rw default-import-policy?
        |      |     |  |       default-policy-type
        |      |     |  +--rw export-policy*           leafref
        |      |     |  +--rw default-export-policy?
        |      |     |          default-policy-type
        |      |     +--rw local-as?                 inet:as-number
        |      |     +--rw peer-as                   inet:as-number
        |      |     +--rw address-family?           identityref
        |      |     +--rw role?                     identityref
        |      |     +--rw multihop?                 uint8
        |      |     +--rw as-override?              boolean
        |      |     +--rw allow-own-as?             uint8
        |      |     +--rw prepend-global-as?        boolean
        |      |     +--rw send-default-route?       boolean
        |      |     +--rw site-of-origin?
        |      |     |       rt-types:route-origin
        |      |     +--rw ipv6-site-of-origin?
        |      |     |       rt-types:ipv6-route-origin
        |      |     +--rw redistribute-connected* [address-family]
        |      |     |  +--rw address-family    identityref
        |      |     |  +--rw enabled?          boolean
        |      |     +--rw bgp-max-prefix
        |      |     |  +--rw max-prefix?          uint32
        |      |     |  +--rw warning-threshold?   decimal64
        |      |     |  +--rw violate-action?      enumeration
        |      |     |  +--rw restart-timer?       uint32
        |      |     +--rw bgp-timers
        |      |     |  +--rw keepalive?   uint16
        |      |     |  +--rw hold-time?   uint16
        |      |     +--rw bfd {vpn-common:bfd}?
        |      |     |  +--rw enabled?                          boolean
        |      |     |  +--rw failure-detection-profile-ref?   leafref
        |      |     |  +--rw network-ref?
        |      |     |          -> /nw:networks/network/network-id
        |      |     +--rw authentication
        |      |     |  +--rw enabled?           boolean
        |      |     |  +--rw keying-material
        |      |     |     +--rw (option)?
        |      |     |        +--:(ao)
```

```
|     |     |        |  +--rw enable-ao?          boolean
|     |     |        |  +--rw ao-keychain?
|     |     |        |          key-chain:key-chain-ref
|     |     |        +--:(md5)
|     |     |        |  +--rw md5-keychain?
|     |     |        |          key-chain:key-chain-ref
|     |     |        +--:(explicit)
|     |     |           +--rw key-id?            uint32
|     |     |           +--rw key?               string
|     |     |           +--rw crypto-algorithm?  identityref
|     |     +--rw status
|     |        +--rw admin-status
|     |        |  +--rw status?        identityref
|     |        |  +--ro last-change?   yang:date-and-time
|     |        +--ro oper-status
|     |           +--ro status?        identityref
|     |           +--ro last-change?   yang:date-and-time
|     +--rw ospf  {vpn-common:rtg-ospf}?
|     |  ...
|     +--rw isis  {vpn-common:rtg-isis}?
|     |  ...
|     +--rw rip  {vpn-common:rtg-rip}?
|     |  ...
|     +--rw vrrp  {vpn-common:rtg-vrrp}?
|        ...
+--rw oam
|  ...
+--rw security
|  ...
+--rw service
   ...
```

Figure 13: BGP Routing Tree Structure

The following data nodes are supported for each 'peer-group':

'name': Defines a name for the peer group.

'local-address': Specifies an address or a reference to an interface to use when establishing the BGP transport session.

'description': Includes a description of the peer group.

'apply-policy': Lists a set of import/export policies [RFC9067] to apply for this group.

'local-as': Indicates a local AS Number (ASN).

'peer-as': Indicates the peer's ASN.

'address-family': Indicates the address family of the peer. It can be set to 'ipv4', 'ipv6', or 'dual-stack'.

This address family might be used together with the service type that uses an AC (e.g., 'vpn-type' [RFC9182]) to derive the appropriate Address Family Identifiers (AFIs) / Subsequent Address Family Identifiers (SAFIs) that will be part of the derived device configurations (e.g., unicast IPv4 MPLS L3VPN (AFI,SAFI = 1,128) as defined in Section 4.3.4 of [RFC4364]).

'role': Specifies the BGP role in a session. Role values are taken from the list defined in Section 4 of [RFC9234].

'multihop': Indicates the number of allowed IP hops to reach a BGP peer.

'as-override': If set, this parameter indicates whether ASN override is enabled, i.e., replacing the ASN of the customer specified in the AS_PATH BGP attribute with the ASN identified in the 'local- as' attribute.

'allow-own-as': Used in some topologies (e.g., hub-and-spoke) to allow the provider's ASN to be included in the AS_PATH BGP attribute received from a peer. Loops are prevented by setting 'allow-own-as' to a maximum number of the provider's ASN occurrences. By default, this parameter is set to '0' (that is, reject any AS_PATH attribute that includes the provider's ASN).

'prepend-global-as': When distinct ASNs are configured at the node and AC levels, this parameter controls whether the ASN provided at the node level is prepended to the AS_PATH attribute.

'send-default-route': Controls whether default routes can be advertised to the peer.

'site-of-origin': Meant to uniquely identify the set of routes learned from a site via a particular AC. It is used to prevent routing loops (Section 7 of [RFC4364]). The Site of Origin attribute is encoded as a Route Origin Extended Community.

'ipv6-site-of-origin': Carries an IPv6 Address Specific BGP Extended Community that is used to indicate the Site of Origin [RFC5701]. It is used to prevent routing loops.

'redistribute-connected': Controls whether the AC is advertised to other PEs.

'bgp-max-prefix': Controls the behavior when a prefix maximum is reached.

'max-prefix': Indicates the maximum number of BGP prefixes allowed in a session for this group. If the limit is reached, the action indicated in 'violate-action' will be followed.

'warning-threshold': A warning notification is triggered when this limit is reached.

'violate-action': Indicates which action to execute when the maximum number of BGP prefixes is reached. Examples of such actions include sending a warning message, discarding extra paths from the peer, or restarting the session.

'restart-timer': Indicates, in seconds, the time interval after which the BGP session will be reestablished.

'bgp-timers': Two timers can be captured in this container: (1) 'hold-time', which is the time interval that will be used for the Hold Timer (Section 4.2 of [RFC4271]) when establishing a BGP session and (2) 'keepalive', which is the time interval for the KeepaliveTimer between a PE and a BGP peer (Section 4.4 of [RFC4271]).

Both timers are expressed in seconds.

'bfd': Indicates whether BFD is enabled or disabled for this neighbor. A BFD profile to apply may also be provided.

'authentication': The module adheres to the recommendations in Section 13.2 of [RFC4364], as it allows enabling the TCP Authentication Option (TCP-AO) [RFC5925] and accommodates the installed base that makes use of MD5.

This version of the model assumes that parameters specific to the TCP-AO are preconfigured as part of the key chain that is referenced in the model. No assumption is made about how such a key chain is preconfigured. However, the structure of the key chain should cover data nodes beyond those in [RFC8177], mainly SendID and RecvID (Section 3.1 of [RFC5925]).

For each neighbor, the following data nodes are supported in addition to similar parameters that are provided for a peer group:

'remote-address': Specifies the remote IP address of a BGP neighbor.

'peer-group': A name of a peer group.

Parameters that are provided at the 'neighbor' level take precedence over the ones provided in the peer group.

'status': Indicates the status of the BGP session.

5.6.3. OSPF

The OSPF routing subtree structure is shown in Figure 14.

```
module: ietf-ac-ntw
  augment /nw:networks/nw:network:
    +--rw ac-profile* [name]
       +--rw name                 string
       +--rw routing-protocols
       |  +--rw routing-protocol* [id]
       |     +--rw id      string
       |     +--rw type?   identityref
       |     +--rw bgp  {vpn-common:rtg-bgp}?
       |     |  ...
       |     +--rw ospf  {vpn-common:rtg-ospf}?
       |     |  +--rw address-family?   identityref
       |     |  +--rw area-id           yang:dotted-quad
       |     |  +--rw metric?           uint16
       |     |  +--rw max-lsa?          uint32
       |     +--rw isis  {vpn-common:rtg-isis}?
       |     |  ...
       |     +--rw rip  {vpn-common:rtg-rip}?
       |     |  ...
       |     +--rw vrrp  {vpn-common:rtg-vrrp}?
       |        ...
       +--rw oam
          ...
  augment /nw:networks/nw:network/nw:node:
    +--rw ac* [name]
       +--rw name                 string
       ...
       +--rw l2-connection  {ac-common:layer2-ac}?
       |  ...
       +--rw ip-connection  {ac-common:layer3-ac}?
       |  ...
       +--rw routing-protocols
       |  +--rw routing-protocol* [id]
       |     +--rw id                  string
       |     +--rw type?               identityref
       |     +--rw routing-profile* [routing-profile-ref]
       |     |  +--rw routing-profile-ref    leafref
       |     |  +--rw network-ref?
       |     |  |       -> /nw:networks/network/network-id
       |     |  +--rw type?                  identityref
       |     +--rw static
       |     |  ...
       |     +--rw bgp  {vpn-common:rtg-bgp}?
       |     |  ...
       |     +--rw ospf {vpn-common:rtg-ospf}?
       |     |  +--rw address-family?   identityref
       |     |  +--rw area-id           yang:dotted-quad
       |     |  +--rw metric?           uint16
       |     |  +--rw sham-links {vpn-common:rtg-ospf-sham-link}?
       |     |  |  +--rw sham-link* [target-site]
       |     |  |     +--rw target-site    string
       |     |  |     +--rw metric?        uint16
       |     |  +--rw max-lsa?          uint32
       |     |  +--rw passive?          boolean
       |     |  +--rw authentication
       |     |  |  +--rw enabled?           boolean
       |     |  |  +--rw keying-material
```

```
|     |  |     +--rw (option)?
|     |  |        +--:(auth-key-chain)
|     |  |        |  +--rw key-chain?
|     |  |        |          key-chain:key-chain-ref
|     |  |        +--:(auth-key-explicit)
|     |  |           +--rw key-id?             uint32
|     |  |           +--rw key?                string
|     |  |           +--rw crypto-algorithm?   identityref
|     |  +--rw status
|     |     +--rw admin-status
|     |     |  +--rw status?        identityref
|     |     |  +--ro last-change?   yang:date-and-time
|     |     +--ro oper-status
|     |        +--ro status?        identityref
|     |        +--ro last-change?   yang:date-and-time
|     +--rw isis  {vpn-common:rtg-isis}?
|     |  ...
|     +--rw rip  {vpn-common:rtg-rip}?
|     |  ...
|     +--rw vrrp  {vpn-common:rtg-vrrp}?
|        ...
+--rw oam
|  ...
+--rw security
|  ...
+--rw service
   ...
```

Figure 14: OSPF Routing Tree Structure

The following OSPF data nodes are supported:

'address-family': Indicates whether IPv4, IPv6, or both address families are to be activated.

When the IPv4 or dual-stack address family is requested, it is up to the implementation (e.g., network orchestrator) to decide whether OSPFv2 [RFC4577] or OSPFv3 [RFC6565] is used to announce IPv4 routes.

'area-id': Indicates the OSPF Area ID.

'metric': Associates a metric with OSPF routes.

'sham-links': Used to create OSPF sham links between two ACs sharing the same area and having a backdoor link (Section 4.2.7 of [RFC4577] and Section 5 of [RFC6565]).

'max-lsa': Sets the maximum number of Link State Advertisements (LSAs) that the OSPF instance will accept.

'passive': Controls whether an OSPF interface is passive or active.

'authentication': Controls the authentication schemes to be enabled for the OSPF instance. The module supports authentication options that are common to both OSPF versions: the Authentication Trailer for OSPFv2 [RFC5709] [RFC7474] and OSPFv3 [RFC7166]; as such, the model does not support [RFC4552].

'status': Indicates the status of the OSPF routing instance.

5.6.4. IS-IS

The IS-IS routing subtree structure is shown in Figure 15.

```
module: ietf-ac-ntw
  augment /nw:networks/nw:network:
    +--rw ac-profile* [name]
       +--rw name                 string
       +--rw routing-protocols
       |  +--rw routing-protocol* [id]
       |     +--rw id      string
       |     +--rw type?   identityref
       |     +--rw bgp  {vpn-common:rtg-bgp}?
       |     |  ...
       |     +--rw ospf  {vpn-common:rtg-ospf}?
       |     |  ...
       |     +--rw isis  {vpn-common:rtg-isis}?
       |     |  +--rw address-family?   identityref
       |     |  +--rw area-address      area-address
       |     |  +--rw level?            identityref
       |     |  +--rw metric?           uint32
       |     |  +--rw passive?          boolean
       |     +--rw rip  {vpn-common:rtg-rip}?
       |     |  ...
       |     +--rw vrrp  {vpn-common:rtg-vrrp}?
       |        ...
       +--rw oam
          ...
  augment /nw:networks/nw:network/nw:node:
    +--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-profile* [routing-profile-ref]
       |     |  +--rw routing-profile-ref    leafref
       |     |  +--rw network-ref?
       |     |  |       -> /nw:networks/network/network-id
       |     |  +--rw type?                  identityref
       |     +--rw static
       |     |  ...
       |     +--rw bgp  {vpn-common:rtg-bgp}?
       |     |  ...
       |     +--rw ospf  {vpn-common:rtg-ospf}?
       |     |  ...
       |     +--rw isis {vpn-common:rtg-isis}?
       |     |  +--rw address-family?   identityref
       |     |  +--rw area-address      area-address
       |     |  +--rw level?            identityref
       |     |  +--rw metric?           uint32
       |     |  +--rw passive?          boolean
       |     |  +--rw authentication
       |     |  |  +--rw enabled?           boolean
       |     |  |  +--rw keying-material
       |     |  |     +--rw (option)?
```

```
|     |  |        +--:(auth-key-chain)
|     |  |        |  +--rw key-chain?
|     |  |        |          key-chain:key-chain-ref
|     |  |        +--:(auth-key-explicit)
|     |  |           +--rw key-id?             uint32
|     |  |           +--rw key?                string
|     |  |           +--rw crypto-algorithm?   identityref
|     |  +--rw status
|     |     +--rw admin-status
|     |     |  +--rw status?        identityref
|     |     |  +--ro last-change?   yang:date-and-time
|     |     +--ro oper-status
|     |        +--ro status?        identityref
|     |        +--ro last-change?   yang:date-and-time
|     +--rw rip  {vpn-common:rtg-rip}?
|     |  ...
|     +--rw vrrp  {vpn-common:rtg-vrrp}?
|        ...
+--rw oam
|  ...
+--rw security
|  ...
+--rw service
   ...
```

Figure 15: IS-IS Routing Tree Structure

The following IS-IS data nodes are supported:

'address-family': Indicates whether IPv4, IPv6, or both address families are to be activated.

'area-address': Indicates the IS-IS area address.

'level': Indicates the IS-IS level: Level 1, Level 2, or both.

'metric': Associates a metric with IS-IS routes.

'passive': Controls whether an IS-IS interface is passive or active.

'authentication': Controls the authentication schemes to be enabled for the IS-IS instance. Both the specification of a key chain [RFC8177] and the direct specification of key and authentication algorithms are supported.

'status': Indicates the status of the IS-IS routing instance.

5.6.5. RIP

The RIP routing subtree structure is shown in Figure 16.

```
module: ietf-ac-ntw
  augment /nw:networks/nw:network:
    +--rw ac-profile* [name]
       +--rw name                 string
       +--rw routing-protocols
       |  +--rw routing-protocol* [id]
       |     +--rw id      string
       |     +--rw type?   identityref
       |     +--rw bgp  {vpn-common:rtg-bgp}?
       |     |  ...
       |     +--rw ospf  {vpn-common:rtg-ospf}?
       |     |  ...
       |     +--rw isis  {vpn-common:rtg-isis}?
       |     |  ...
       |     +--rw rip  {vpn-common:rtg-rip}?
       |     |  +--rw address-family?   identityref
       |     |  +--rw timers
       |     |  |  +--rw update-interval?     uint16
       |     |  |  +--rw invalid-interval?    uint16
       |     |  |  +--rw holddown-interval?   uint16
       |     |  |  +--rw flush-interval?      uint16
       |     |  +--rw default-metric?   uint8
       |     +--rw vrrp  {vpn-common:rtg-vrrp}?
       |        ...
       +--rw oam
          ...
  augment /nw:networks/nw:network/nw:node:
    +--rw ac* [name]
       +--rw name                 string
       ...
       +--rw l2-connection  {ac-common:layer2-ac}?
       |  ...
       +--rw ip-connection  {ac-common:layer3-ac}?
       |  ...
       +--rw routing-protocols
       |  +--rw routing-protocol* [id]
       |     +--rw id                  string
       |     +--rw type?               identityref
       |     +--rw routing-profile* [routing-profile-ref]
       |     |  +--rw routing-profile-ref    leafref
       |     |  +--rw network-ref?
       |     |  |       -> /nw:networks/network/network-id
       |     |  +--rw type?                  identityref
       |     +--rw static
       |     |  ...
       |     +--rw bgp  {vpn-common:rtg-bgp}?
       |     |  ...
       |     +--rw ospf  {vpn-common:rtg-ospf}?
       |     |  ...
       |     +--rw isis  {vpn-common:rtg-isis}?
       |     |  ...
       |     +--rw rip  {vpn-common:rtg-rip}?
       |     |  +--rw address-family?   identityref
       |     |  +--rw timers
       |     |  |  +--rw update-interval?     uint16
       |     |  |  +--rw invalid-interval?    uint16
       |     |  |  +--rw holddown-interval?   uint16
```

```
        |     |  |  +--rw flush-interval?      uint16
        |     |  +--rw default-metric?   uint8
        |     |  +--rw authentication
        |     |  |  +--rw enabled?            boolean
        |     |  |  +--rw keying-material
        |     |  |     +--rw (option)?
        |     |  |        +--:(auth-key-chain)
        |     |  |        |  +--rw key-chain?
        |     |  |        |          key-chain:key-chain-ref
        |     |  |        +--:(auth-key-explicit)
        |     |  |           +--rw key?                string
        |     |  |           +--rw crypto-algorithm?   identityref
        |     |  +--rw status
        |     |     +--rw admin-status
        |     |     |  +--rw status?        identityref
        |     |     |  +--ro last-change?   yang:date-and-time
        |     |     +--ro oper-status
        |     |        +--ro status?        identityref
        |     |        +--ro last-change?   yang:date-and-time
        |     +--rw vrrp
        |        ...
        +--rw oam
        |  ...
        +--rw security
        |  ...
        +--rw service
           ...
```

Figure 16: RIP Routing Tree Structure

The following RIP data nodes are supported:

'address-family': Indicates whether IPv4, IPv6, or both address families are to be activated. This parameter is used to determine whether RIPv2 [RFC2453], RIP Next Generation (RIPng) [RFC2080], or both are to be enabled.

'timers': Indicates the following timers (expressed in seconds):

'update-interval': The interval at which RIP updates are sent.

'invalid-interval': The interval before a RIP route is declared invalid.

'holddown-interval': The interval before better RIP routes are released.

'flush-interval': The interval before a route is removed from the routing table.

'default-metric': Sets the default RIP metric.

'authentication': Controls the authentication schemes to be enabled for the RIP instance.

'status': Indicates the status of the RIP routing instance.

5.6.6. VRRP

The VRRP subtree structure is shown in Figure 17.

```
module: ietf-ac-ntw
  augment /nw:networks/nw:network:
    +--rw ac-profile* [name]
       +--rw name                 string
       +--rw routing-protocols
       |  +--rw routing-protocol* [id]
       |     +--rw id      string
       |     +--rw type?   identityref
       |     +--rw bgp  {vpn-common:rtg-bgp}?
       |     |  ...
       |     +--rw ospf  {vpn-common:rtg-ospf}?
       |     |  ...
       |     +--rw isis  {vpn-common:rtg-isis}?
       |     |  ...
       |     +--rw rip  {vpn-common:rtg-rip}?
       |     |  ...
       |     +--rw vrrp  {vpn-common:rtg-vrrp}?
       |        +--rw address-family?   identityref
       |        +--rw ping-reply?       boolean
       +--rw oam
          ...
  augment /nw:networks/nw:network/nw:node:
    +--rw ac* [name]
       +--rw name                 string
       ...
       +--rw l2-connection  {ac-common:layer2-ac}?
       |  ...
       +--rw ip-connection  {ac-common:layer3-ac}?
       |  ...
       +--rw routing-protocols
       |  +--rw routing-protocol* [id]
       |     +--rw id                  string
       |     +--rw type?               identityref
       |     +--rw routing-profile* [routing-profile-ref]
       |     |  +--rw routing-profile-ref    leafref
       |     |  +--rw network-ref?
       |     |  |       -> /nw:networks/network/network-id
       |     |  +--rw type?                  identityref
       |     +--rw static
       |     |  ...
       |     +--rw bgp  {vpn-common:rtg-bgp}?
       |     |  ...
       |     +--rw ospf  {vpn-common:rtg-ospf}?
       |     |  ...
       |     +--rw isis  {vpn-common:rtg-isis}?
       |     |  ...
       |     +--rw rip  {vpn-common:rtg-rip}?
       |     |  ...
       |     +--rw vrrp  {vpn-common:rtg-vrrp}?
       |        +--rw address-family?       identityref
       |        +--rw vrrp-group?           uint8
       |        +--rw backup-peer?          inet:ip-address
       |        +--rw virtual-ip-address*   inet:ip-address
       |        +--rw priority?             uint8
       |        +--rw ping-reply?           boolean
       |        +--rw status
       |           +--rw admin-status
```

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

Figure 17: VRRP Tree Structure

The following VRRP data nodes are supported:

'address-family': Indicates whether IPv4, IPv6, or both address families are to be activated. Note that VRRP version 3 [RFC9568] supports both IPv4 and IPv6.

'vrrp-group': Used to identify the VRRP group.

'backup-peer': Carries the IP address of the peer.

'virtual-ip-address': Includes virtual IP addresses for a single VRRP group.

'priority': Assigns the VRRP election priority for the backup virtual router.

'ping-reply': Controls whether the VRRP speaker should reply to ping requests.

'status': Indicates the status of the VRRP instance.

Note that no authentication data node is included for VRRP, as there isn't any type of VRRP authentication at this time (see Section 9 of [RFC9568]).

5.7. OAM

The OAM subtree structure is shown in Figure 18.

```
augment /nw:networks/nw:network:
  +--rw ac-profile* [name]
     +--rw name                 string
     +--rw routing-protocols
     |  ...
     +--rw oam
        +--rw bfd {vpn-common:bfd}?
           +--rw session-type?               identityref
           +--rw desired-min-tx-interval?    uint32
           +--rw required-min-rx-interval?   uint32
           +--rw local-multiplier?           uint8
           +--rw holdtime?                   uint32
augment /nw:networks/nw:network/nw:node:
  +--rw ac* [name]
     +--rw name                 string
     + ...
     +--rw l2-connection  {ac-common:layer2-ac}?
     |  ...
     +--rw ip-connection  {ac-common:layer3-ac}?
     |  ...
     +--rw routing-protocols
     |  ...
     +--rw oam
     |  +--rw bfd {vpn-common:bfd}?
     |     +--rw session* [dest-addr]
     |        +--rw dest-addr                   inet:ip-address
     |        +--rw source-address?             union
     |        +--rw failure-detection-profile-ref?   leafref
     |        +--rw network-ref?
     |        |       -> /nw:networks/network/network-id
     |        +--rw session-type?               identityref
     |        +--rw desired-min-tx-interval?    uint32
     |        +--rw required-min-rx-interval?   uint32
     |        +--rw local-multiplier?           uint8
     |        +--rw holdtime?                   uint32
     |        +--rw authentication!
     |        |  +--rw key-chain?    key-chain:key-chain-ref
     |        |  +--rw meticulous?   boolean
     |        +--rw status
     |           +--rw admin-status
     |           |  +--rw status?        identityref
     |           |  +--ro last-change?   yang:date-and-time
     |           +--ro oper-status
     |              +--ro status?        identityref
     |              +--ro last-change?   yang:date-and-time
     +--rw security
     |  ...
     +--rw service
        ...
```

Figure 18: OAM Tree Structure

The following OAM data nodes can be specified for each BFD session:

'dest-addr':

Specifies the BFD peer address. This data node is mapped to 'remote-address' of the BFD container in [RFC9834]. 'dest-address' is used here to ease the mapping with the underlying device model defined in [RFC9127].

'source-address': Specifies the local IP address or interface to use for the session. This data node is mapped to 'local-address' of the BFD container in [RFC9834]. 'source-address' is used here to ease the mapping with the underlying device model defined in [RFC9127].

'failure-detection-profile-ref': Refers to a BFD profile in Section 5.3.

'network-ref': Includes a network reference to uniquely identify a BFD profile.

'session-type': Indicates which BFD flavor is used to set up the session (e.g., classic BFD [RFC5880], Seamless BFD [RFC7880]). By default, it is assumed that the BFD session will follow the behavior specified in [RFC5880].

'desired-min-tx-interval': The minimum interval, in microseconds, to use when transmitting BFD Control packets, less any jitter applied.

'required-min-rx-interval': The minimum interval, in microseconds, between received BFD Control packets, less any jitter applied by the sender.

'local-multiplier': The negotiated transmit interval, multiplied by this value, provides the detection time for the peer.

'holdtime': Used to indicate the expected BFD holddown time, in milliseconds.

'authentication': Includes the required information to enable the BFD authentication modes discussed in Section 6.7 of [RFC5880]. In particular, 'meticulous' controls the activation of meticulous mode as discussed in Sections 6.7.3 and 6.7.4 of [RFC5880].

'status': Indicates the status of BFD.

5.8. Security

The security subtree structure is shown in Figure 19. The 'security' container specifies the encryption to be applied to traffic for a given AC. The model can be used to directly control the encryption to be applied (e.g., Layer 2 or Layer 3 encryption) or invoke a local encryption profile.

```
augment /nw:networks/nw:network/nw:node:
  +--rw ac* [name]
     +--rw name                 string
     + ...
     +--rw l2-connection  {ac-common:layer2-ac}?
     |  ...
     +--rw ip-connection  {ac-common:layer3-ac}?
     |  ...
     +--rw routing-protocols
     |  ...
     +--rw oam
     |  ...
     +--rw security
     |  +--rw encryption {vpn-common:encryption}?
     |  |  +--rw enabled?   boolean
     |  |  +--rw layer?     enumeration
     |  +--rw encryption-profile
     |     +--rw (profile)?
     |        +--:(provider-profile)
     |        |  +--rw encryption-profile-ref?   leafref
     |        |  +--rw network-ref?
     |        |          -> /nw:networks/network/network-id
     |        +--:(customer-profile)
     |           +--rw customer-key-chain?  key-chain:key-chain-ref
     +--rw service
        ...
```

Figure 19: Security Tree Structure

5.9. Service

The service subtree structure is shown in Figure 20.

```
augment /nw:networks/nw:network/nw:node:
  +--rw ac* [name]
     +--rw name                 string
     + ...
     +--rw l2-connection  {ac-common:layer2-ac}?
     |  ...
     +--rw ip-connection  {ac-common:layer3-ac}?
     |  ...
     +--rw routing-protocols
     |  ...
     +--rw oam
     |  ...
     +--rw security
     |  ...
     +--rw service
        +--rw mtu?                       uint32
        +--rw svc-pe-to-ce-bandwidth {vpn-common:inbound-bw}?
        |  +--rw bandwidth* [bw-type]
        |     +--rw bw-type      identityref
        |     +--rw (type)?
        |        +--:(per-cos)
        |        |  +--rw cos* [cos-id]
        |        |     +--rw cos-id    uint8
        |        |     +--rw cir?      uint64
        |        |     +--rw cbs?      uint64
        |        |     +--rw eir?      uint64
        |        |     +--rw ebs?      uint64
        |        |     +--rw pir?      uint64
        |        |     +--rw pbs?      uint64
        |        +--:(other)
        |           +--rw cir?   uint64
        |           +--rw cbs?   uint64
        |           +--rw eir?   uint64
        |           +--rw ebs?   uint64
        |           +--rw pir?   uint64
        |           +--rw pbs?   uint64
        +--rw svc-ce-to-pe-bandwidth {vpn-common:outbound-bw}?
        |  +--rw bandwidth* [bw-type]
        |     +--rw bw-type      identityref
        |     +--rw (type)?
        |        +--:(per-cos)
        |        |  +--rw cos* [cos-id]
        |        |     +--rw cos-id    uint8
        |        |     +--rw cir?      uint64
        |        |     +--rw cbs?      uint64
        |        |     +--rw eir?      uint64
        |        |     +--rw ebs?      uint64
        |        |     +--rw pir?      uint64
        |        |     +--rw pbs?      uint64
        |        +--:(other)
        |           +--rw cir?   uint64
        |           +--rw cbs?   uint64
        |           +--rw eir?   uint64
        |           +--rw ebs?   uint64
        |           +--rw pir?   uint64
        |           +--rw pbs?   uint64
        +--rw qos {vpn-common:qos}?
```

```
           |  +--rw qos-profiles
           |     +--rw qos-profile* [qos-profile-ref]
           |        +--rw qos-profile-ref    leafref
           |        +--rw network-ref?
           |        |       -> /nw:networks/network/network-id
           |        +--rw direction?         identityref
           +--rw access-control-list
              +--rw acl-profiles
                 +--rw acl-profile* [forwarding-profile-ref]
                    +--rw forwarding-profile-ref    leafref
                    +--rw network-ref?
                            -> /nw:networks/network/network-id
```

Figure 20: Service Tree Structure

The service data nodes are defined as follows:

'mtu': Specifies the Layer 2 MTU, in bytes, for the AC.

'svc-pe-to-ce-bandwidth' and 'svc-ce-to-pe-bandwidth': Specify the service bandwidth for the AC.

'svc-pe-to-ce-bandwidth': Indicates the inbound bandwidth of the connection (i.e., download bandwidth from the service provider to the site).

'svc-ce-to-pe-bandwidth': Indicates the outbound bandwidth of the connection (i.e., upload bandwidth from the site to the service provider).

'svc-pe-to-ce-bandwidth' and 'svc-ce-to-pe-bandwidth' can be represented using the Committed Information Rate (CIR), the Committed Burst Size (CBS), the Excess Information Rate (EIR), the Excess Burst Size (EBS), the Peak Information Rate (PIR), and the Peak Burst Size (PBS). CIR, EIR, and PIR are expressed in bps, while CBS, EBS, and PBS are expressed in bytes.

The following types, defined in [RFC9181], can be used to indicate the bandwidth type:

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

'bw-per-port': The bandwidth is per port.

'bw-per-site': The bandwidth is to all peer SAPs that belong to the same site.

'bw-per-service': The bandwidth is per service instance that is bound to an AC.

'qos': Specifies a list of QoS profiles to apply for this AC.

'access-control-list': Specifies a list of ACL profiles to apply for this AC.

6. YANG Module

This module uses types defined in [RFC6991], [RFC8177], [RFC8294], [RFC8343], [RFC9067], [RFC9181], [RFC9833], and [IEEE802.1Qcp].

```
<CODE BEGINS> file "ietf-ac-ntw@2025-08-11.yang"

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

  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-key-chain {
    prefix key-chain;
    reference
      "RFC 8177: YANG Data Model for Key Chains";
  }
  import ietf-routing-types {
    prefix rt-types;
    reference
      "RFC 8294: Common YANG Data Types for the Routing Area";
  }
  import ietf-routing-policy {
    prefix rt-pol;
    reference
      "RFC 9067: A YANG Data Model for Routing Policy";
  }
  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343: A YANG Data Model for Interface Management";
  }
  import ieee802-dot1q-types {
    prefix dot1q-types;
    reference
      "IEEE Std 802.1Qcp: Bridges and Bridged Networks--
                          Amendment 30: YANG Data Model";
  }
  import ietf-network {
    prefix nw;
    reference
      "RFC 8345: A YANG Data Model for Network Topologies,
                 Section 6.1";
```

```
  }
  import ietf-sap-ntw {
    prefix sap;
    reference
      "RFC 9408: A YANG Network Data Model for Service Attachment
                 Points (SAPs)";
  }
  import ietf-ac-common {
    prefix ac-common;
    reference
      "RFC 9833: A Common YANG Data Model for Attachment Circuits";
  }
  import ietf-ac-svc {
    prefix ac-svc;
    reference
      "RFC 9834: YANG Data Models for Bearers and 'Attachment
                  Circuits'-as-a-Service (ACaaS)";
  }

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

     Editor:   Mohamed Boucadair
               <mailto:mohamed.boucadair@orange.com>
     Author:   Richard Roberts
               <mailto:rroberts@juniper.net>
     Author:   Oscar Gonzalez de Dios
               <mailto:oscar.gonzalezdedios@telefonica.com>
     Author:   Samier Barguil
               <mailto:ssamier.barguil_giraldo@nokia.com>
     Author:   Bo Wu
               <mailto:lana.wubo@huawei.com>";
  description
    "This YANG module defines a YANG network model for the management
     of attachment circuits (ACs).

     Copyright (c) 2025 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

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

     This version of this YANG module is part of RFC 9835; see the
     RFC itself for full legal notices.";

  revision 2025-08-11 {
    description
      "Initial revision.";
    reference
      "RFC 9835: A YANG Network Data Model for Attachment Circuits";
  }
```

```
  // References

  /* A set of groupings to ease referencing cross-modules */

  grouping attachment-circuit-reference {
    description
      "This grouping can be used to reference an attachment circuit
       in a specific node.";
    leaf ac-ref {
      type leafref {
        path "/nw:networks/nw:network[nw:network-id=current()/../"
           + "network-ref]/nw:node[nw:node-id=current()/../"
           + "node-ref]/ac-ntw:ac/ac-ntw:name";
        require-instance false;
      }
      description
        "An absolute reference to an attachment circuit.";
    }
    uses nw:node-ref;
  }

  grouping attachment-circuit-references {
    description
      "This grouping can be used to reference a list of attachment
       circuits in a specific node.";
    leaf-list ac-ref {
      type leafref {
        path "/nw:networks/nw:network[nw:network-id=current()/../"
           + "network-ref]/nw:node[nw:node-id=current()/../"
           + "node-ref]/ac-ntw:ac/ac-ntw:name";
        require-instance false;
      }
      description
        "An absolute reference to an attachment circuit.";
    }
    uses nw:node-ref;
  }

  grouping ac-profile-reference {
    description
      "This grouping can be used to reference an attachment circuit
       profile.";
    leaf ac-profile-ref {
      type leafref {
        path "/nw:networks/nw:network[nw:network-id=current()/../"
           + "network-ref]/ac-ntw:ac-profile/ac-ntw:name";
        require-instance false;
      }
      description
        "An absolute reference to an attachment circuit.";
    }
    uses nw:network-ref;
  }

  grouping encryption-profile-reference {
    description
      "This grouping can be used to reference an encryption
```

```
        profile.";
    leaf encryption-profile-ref {
      type leafref {
        path "/nw:networks/nw:network[nw:network-id=current()/../"
           + "network-ref]"
           + "/ac-ntw:specific-provisioning-profiles"
           + "/ac-ntw:valid-provider-identifiers"
           + "/ac-ntw:encryption-profile-identifier/ac-ntw:id";
        require-instance false;
      }
      description
        "An absolute reference to an encryption profile.";
    }
    uses nw:network-ref;
  }

  grouping qos-profile-reference {
    description
      "This grouping can be used to reference a QoS profile.";
    leaf qos-profile-ref {
      type leafref {
        path "/nw:networks/nw:network[nw:network-id=current()/../"
           + "network-ref]"
           + "/ac-ntw:specific-provisioning-profiles"
           + "/ac-ntw:valid-provider-identifiers"
           + "/ac-ntw:qos-profile-identifier/ac-ntw:id";
        require-instance false;
      }
      description
        "An absolute reference to a QoS profile.";
    }
    uses nw:network-ref;
  }

  grouping failure-detection-profile-reference {
    description
      "This grouping can be used to reference a failure detection
       profile.";
    leaf failure-detection-profile-ref {
      type leafref {
        path "/nw:networks/nw:network[nw:network-id=current()/../"
          + "network-ref]"
          + "/ac-ntw:specific-provisioning-profiles"
          + "/ac-ntw:valid-provider-identifiers"
          + "/ac-ntw:failure-detection-profile-identifier/ac-ntw:id";
        require-instance false;
      }
      description
        "An absolute reference to a failure detection profile.";
    }
    uses nw:network-ref;
  }

  grouping forwarding-profile-reference {
    description
      "This grouping can be used to reference a forwarding profile.";
    leaf forwarding-profile-ref {
      type leafref {
```

```
        path "/nw:networks/nw:network[nw:network-id=current()/../"
           + "network-ref]"
           + "/ac-ntw:specific-provisioning-profiles"
           + "/ac-ntw:valid-provider-identifiers"
           + "/ac-ntw:forwarding-profile-identifier/ac-ntw:id";
        require-instance false;
      }
      description
        "An absolute reference to a forwarding profile.";
    }
    uses nw:network-ref;
  }

  grouping routing-profile-reference {
    description
      "This grouping can be used to reference a routing profile.";
    leaf routing-profile-ref {
      type leafref {
        path "/nw:networks/nw:network[nw:network-id=current()/../"
           + "network-ref]"
           + "/ac-ntw:specific-provisioning-profiles"
           + "/ac-ntw:valid-provider-identifiers"
           + "/ac-ntw:routing-profile-identifier/ac-ntw:id";
        require-instance false;
      }
      description
        "An absolute reference to a routing profile.";
    }
    uses nw:network-ref;
  }

  // Layer 2 connection

  grouping l2-connection {
    description
      "Defines Layer 2 protocols and parameters that are required to
       enable AC connectivity on the network side.";
    container encapsulation {
      description
        "Container for Layer 2 encapsulation.";
      leaf encap-type {
        type identityref {
          base vpn-common:encapsulation-type;
        }
        description
          "Tagged interface type.";
      }
      container dot1q {
        when "derived-from-or-self(../encap-type, "
           + "'vpn-common:dot1q')" {
          description
            "Only applies when the type of the tagged interface is
             'dot1q'.";
        }
        description
          "Tagged interface.";
        uses ac-common:dot1q;
        container tag-operations {
```

```
          description
            "Sets the tag manipulation policy for this AC.  It
             defines a set of tag manipulations that allow for the
             insertion, removal, or rewriting of 802.1Q VLAN tags.
             These operations are indicated for the CE-PE direction.
             By default, tag operations are symmetric.  As such, the
             reverse tag operation is assumed on the PE-CE
             direction.";
          choice op-choice {
            description
              "Selects the tag rewriting policy for an AC.";
            leaf pop {
              type empty;
              description
                "Pop the outer tag.";
            }
            leaf push {
              type empty;
              description
                "Pushes one or two tags defined by the tag-1 and
                 tag-2 leaves.  It is assumed that, absent any
                 policy, the default value of 0 will be used for
                 the Priority Code Point (PCP) setting.";
            }
            leaf translate {
              type empty;
              description
                "Translates the outer tag to one or two tags.  PCP
                 bits are preserved.";
            }
          }
          leaf tag-1 {
            when 'not(../pop)';
            type dot1q-types:vlanid;
            description
              "A first tag to be used for push or translate
               operations.  This tag will be used as the outermost
               tag as a result of the tag operation.";
          }
          leaf tag-1-type {
            type dot1q-types:dot1q-tag-type;
            default "dot1q-types:s-vlan";
            description
              "Specifies a specific 802.1Q tag type of tag-1.";
          }
          leaf tag-2 {
            when '(../translate)';
            type dot1q-types:vlanid;
            description
              "A second tag to be used for translation.";
          }
          leaf tag-2-type {
            type dot1q-types:dot1q-tag-type;
            default "dot1q-types:c-vlan";
            description
              "Specifies a specific 802.1Q tag type of tag-2.";
          }
        }
```

```
      }
      container priority-tagged {
        when "derived-from-or-self(../encap-type, "
           + "'vpn-common:priority-tagged')" {
          description
            "Only applies when the type of the tagged interface is
             'priority-tagged'.";
        }
        description
          "Priority tagged container.";
        uses ac-common:priority-tagged;
      }
      container qinq {
        when "derived-from-or-self(../encap-type, "
           + "'vpn-common:qinq')" {
          description
            "Only applies when the type of the tagged interface is
             'QinQ'.";
        }
        description
          "Includes QinQ parameters.";
        uses ac-common:qinq;
        container tag-operations {
          description
            "Sets the tag manipulation policy for this AC.  It
             defines a set of tag manipulations that allow for the
             insertion, removal, or rewriting of 802.1Q VLAN tags.
             These operations are indicated for the CE-PE direction.
             By default, tag operations are symmetric.  As such, the
             reverse tag operation is assumed on the PE-CE
             direction.";
          choice op-choice {
            description
              "Selects the tag rewriting policy for an AC.";
            leaf pop {
              type uint8 {
                range "1|2";
              }
              description
                "Pops one or two tags as a function of the indicated
                 pop value.";
            }
            leaf push {
              type empty;
              description
                "Pushes one or two tags defined by the tag-1 and
                 tag-2 leaves.  It is assumed that, absent any
                 policy, the default value of 0 will be used for
                 PCP setting.";
            }
            leaf translate {
              type uint8 {
                range "1|2";
              }
              description
                "Translates one or two outer tags.  PCP bits are
                 preserved.  The following operations are supported:
```

```
                - translate 1 with tag-1 leaf is provided: only the
                  outermost tag is translated to the value in tag-1.

                - translate 2 with both tag-1 and tag-2 leaves are
                  provided: both outer and inner tags are translated
                  to the values in tag-1 and tag-2, respectively.

                - translate 2 with tag-1 leaf is provided: the
                  outer tag is popped while the inner tag is
                  translated to the value in tag-1.";
            }
          }
          leaf tag-1 {
            when 'not(../pop)';
            type dot1q-types:vlanid;
            description
              "A first tag to be used for push or translate
               operations.  This tag will be used as the outermost
               tag as a result of the tag operation.";
          }
          leaf tag-1-type {
            type dot1q-types:dot1q-tag-type;
            default "dot1q-types:s-vlan";
            description
              "Specifies a specific 802.1Q tag type of tag-1.";
          }
          leaf tag-2 {
            when 'not(../pop)';
            type dot1q-types:vlanid;
            description
              "A second tag to be used for push or translate
               operations.";
          }
          leaf tag-2-type {
            type dot1q-types:dot1q-tag-type;
            default "dot1q-types:c-vlan";
            description
              "Specifies a specific 802.1Q tag type of tag-2.";
          }
        }
      }
    }
    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.";
        uses ac-common:l2-tunnel-service;
      }
      case l2vpn {
        leaf l2vpn-id {
          type vpn-common:vpn-id;
          description
            "Indicates the L2VPN service associated with an
             Integrated Routing and Bridging (IRB) interface.";
```

```
        }
      }
    }
  }

  grouping l2-connection-if-ref {
    description
      "Specifies Layer 2 connection parameters with interface
       references.";
    uses l2-connection;
    leaf l2-termination-point {
      type string;
      description
        "Specifies a reference to a local Layer 2 termination point,
         such as a Layer 2 sub-interface.";
    }
    leaf local-bridge-reference {
      type string;
      description
        "Specifies a local bridge reference to accommodate, e.g.,
         implementations that require internal bridging.
         A reference may be a local bridge domain.";
    }
    leaf bearer-reference {
      if-feature "ac-common:server-assigned-reference";
      type string;
      description
        "This is an internal reference for the service provider to
         identify the bearer associated with this AC.";
    }
    container lag-interface {
      if-feature "vpn-common:lag-interface";
      description
        "Container for configuration of Link Aggregation Group (LAG)
         interface attributes.";
      leaf lag-interface-id {
        type string;
        description
          "LAG interface identifier.";
      }
      container member-link-list {
        description
          "Container for the member link list.";
        list member-link {
          key "name";
          description
            "Member link.";
          leaf name {
            type string;
            description
              "Member link name.";
          }
        }
      }
    }
  }

  // IPv4 connection
```

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

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

                   When both 'start-address' and 'end-address' are
                   specified, it implies a range inclusive of both
                   addresses.";
                leaf pool-id {
                  type string;
                  description
                    "A pool identifier for the address range from
                     'start-address' to 'end-address'.";
                }
                leaf start-address {
                  type inet: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 access.";
        }
        choice service-type {
          description
            "Choice based on the DHCP service type.";
          case relay {
            description
              "Container for a list of the provider's DHCP servers
               (i.e., 'dhcp-service-type' is set to 'relay').";
            leaf-list server-ip-address {
              type inet:ipv4-address;
              description
                "IPv4 addresses of the provider's DHCP server, for
                 use by the local DHCP relay.";
            }
          }
        }
      }
      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 static IPv4 addresses that are used.";
        list address {
          key "address-id";
          ordered-by user;
          description
            "Lists the IPv4 addresses that are used.  The first
             address of the list is the primary address of the
             connection.";
          leaf address-id {
            type string;
            description
              "An identifier of the static IPv4 address.";
          }
          leaf customer-address {
            type inet:ipv4-address;
            description
              "An IPv4 address of the customer side.";
          }
          uses failure-detection-profile-reference;
        }
      }
    }
  }

  grouping ipv6-connection {
    description
      "IPv6-specific connection parameters.";
    leaf local-address {
      type inet:ipv6-address;
      description
        "IPv6 address of the provider side.";
    }
    uses ac-common:ipv6-allocation-type;
    choice allocation-type {
      description
        "Choice of the IPv6 address allocation.";
      case dynamic {
        description
          "When the addresses are allocated by DHCP or other
           dynamic means local to the infrastructure.";
        choice address-assign {
          description
            "A choice for how IPv6 addresses are assigned.";
          case number {
            leaf number-of-dynamic-address {
              type uint16;
              description
                "Specifies the number of IP addresses to be
                 assigned to the customer on this access.";
            }
          }
          case explicit {
            container customer-addresses {
              description
                "Container for customer addresses to be allocated
```

```
                using DHCP.";
            list address-pool {
              key "pool-id";
              description
                "Describes IPv6 addresses to be dynamically
                 allocated.

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

                 When both 'start-address' and 'end-address' are
                 specified, it implies a range inclusive of both
                 addresses.";
              leaf pool-id {
                type string;
                description
                  "A pool identifier for the address range from
                   'start-address' to 'end-address'.";
              }
              leaf start-address {
                type inet:ipv6-address;
                mandatory true;
                description
                  "Indicates the first address in the pool.";
              }
              leaf end-address {
                type inet:ipv6-address;
                description
                  "Indicates the last address in the pool.";
              }
            }
          }
        }
      }
      choice provider-dhcp {
        description
          "Parameters related to DHCP-allocated addresses.
           IP addresses are allocated by DHCP, which is provided
           by the operator.";
        leaf dhcp-service-type {
          type enumeration {
            enum server {
              description
                "Local DHCP server.";
            }
            enum relay {
              description
                "Local DHCP relay.  DHCP requests are relayed to
                 a provider's server.";
            }
          }
          description
            "Indicates the type of DHCP service to be enabled on
             this access.";
        }
        choice service-type {
          description
            "Choice based on the DHCP service type.";
```

```
            case relay {
              description
                "Container for a list of the provider's DHCP servers
                 (i.e., 'dhcp-service-type' is set to 'relay').";
              leaf-list server-ip-address {
                type inet:ipv6-address;
                description
                  "IPv6 addresses of the provider's DHCP server, for
                   use by the local DHCP relay.";
              }
            }
          }
        }
        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 servers.";
            }
          }
        }
      }
      case static-addresses {
        description
          "Lists the static IPv6 addresses that are used.";
        list address {
          key "address-id";
          ordered-by user;
          description
            "Lists the IPv6 addresses that are used.  The first
             address of the list is the primary 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.";
          }
          uses failure-detection-profile-reference;
        }
      }
    }
  }

  grouping ip-connection {
    description
      "Defines IP connection parameters.";
    leaf l3-termination-point {
      type string;
```

```
      description
        "Specifies a reference to a local Layer 3 termination point,
         such as a bridge domain interface.";
    }
    container ipv4 {
      if-feature "vpn-common:ipv4";
      description
        "IPv4-specific connection parameters.";
      uses ipv4-connection;
    }
    container ipv6 {
      if-feature "vpn-common:ipv6";
      description
        "IPv6-specific connection parameters.";
      uses ipv6-connection;
    }
  }

  /* Routing */
  //BGP base parameters

  grouping bgp-base {
    description
      "Configuration specific to BGP.";
    leaf description {
      type string;
      description
        "Includes a description of the BGP session.  This description
         is meant to be used for diagnostic purposes.  The semantics
         of the description are local to an implementation.";
    }
    uses rt-pol:apply-policy-group;
    leaf local-as {
      type inet:as-number;
      description
        "Indicates a local AS Number (ASN), if an ASN distinct from
         the ASN configured at the AC level is needed.";
    }
    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.";
    }
    leaf role {
      type identityref {
        base ac-common:bgp-role;
      }
```

```
        description
          "Specifies the BGP role (provider, customer, peer, etc.).";
      }
      leaf multihop {
        type uint8;
        description
          "Describes the number of IP hops allowed between a given BGP
           neighbor and the PE.";
      }
      leaf as-override {
        type boolean;
        description
          "Defines whether ASN override is enabled, i.e., replacing the
           ASN of the customer specified in the AS_PATH attribute with
           the local ASN.";
      }
      leaf allow-own-as {
        type uint8;
        description
          "If set, specifies the maximum number of occurrences of the
           provider's ASN that are permitted within the AS_PATH
           before it is rejected.";
      }
      leaf prepend-global-as {
        type boolean;
        description
          "In some situations, the ASN that is provided at the node
           level may be distinct from the ASN configured at the AC.
           When such ASNs are provided, they are both prepended to the
           BGP route updates for this AC.  To disable that behavior,
           'prepend-global-as' must be set to 'false'.  In such a
           case, the ASN that is provided at the node level is not
           prepended to the BGP route updates for this access.";
      }
      leaf send-default-route {
        type boolean;
        description
          "Defines whether default routes can be advertised to a peer.
           If set to 'true', the default routes are advertised to
           a peer.";
      }
      leaf site-of-origin {
        when "derived-from-or-self(../address-family, "
           + "'vpn-common:ipv4' or 'vpn-common:dual-stack')" {
          description
            "Only applies if IPv4 is activated.";
        }
        type rt-types:route-origin;
        description
          "The Site of Origin attribute is encoded as a Route Origin
           Extended Community.  It is meant to uniquely identify the
           set of routes learned from a site via a particular AC and
           is used to prevent routing loops.";
        reference
          "RFC 4364: BGP/MPLS IP Virtual Private Networks (VPNs),
                     Section 7";
      }
      leaf ipv6-site-of-origin {
```

```
        when "derived-from-or-self(../address-family, "
           + "'vpn-common:ipv6' or 'vpn-common:dual-stack')" {
          description
            "Only applies if IPv6 is activated.";
        }
        type rt-types:ipv6-route-origin;
        description
          "The IPv6 Site of Origin attribute is encoded as an IPv6
           Route Origin Extended Community.  It is meant to uniquely
           identify the set of routes learned from a site.";
        reference
          "RFC 5701: IPv6 Address Specific BGP Extended Community
                     Attribute";
      }
      list redistribute-connected {
        key "address-family";
        description
          "Indicates, per address family, the policy to follow for
           connected routes.";
        leaf address-family {
          type identityref {
            base vpn-common:address-family;
          }
          description
            "Indicates the address family.";
        }
        leaf enabled {
          type boolean;
          description
            "Enables, when set to 'true', the redistribution of
             connected routes.";
        }
      }
      container bgp-max-prefix {
        description
          "Controls the behavior when a prefix maximum is reached.";
        leaf max-prefix {
          type uint32;
          description
            "Indicates the maximum number of BGP prefixes allowed in
             the BGP session.

             It allows control of how many prefixes can be received
             from a neighbor.

             If the limit is exceeded, the action indicated in
             'violate-action' will be followed.";
          reference
            "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
                       Section 8.2.2";
        }
        leaf warning-threshold {
          type decimal64 {
            fraction-digits 5;
            range "0..100";
          }
          units "percent";
          description
```

```
          "When this value is reached, a warning notification will be
           triggered.";
      }
      leaf violate-action {
        type enumeration {
          enum warning {
            description
              "Only a warning message is sent to the peer when the
               limit is exceeded.";
          }
          enum discard-extra-paths {
            description
              "Discards extra paths when the limit is exceeded.";
          }
          enum restart {
            description
              "The BGP session restarts after the indicated time
               interval.";
          }
        }
        description
          "If the BGP neighbor 'max-prefix' limit is reached, the
           action indicated in 'violate-action' will be followed.";
      }
      leaf restart-timer {
        type uint32;
        units "seconds";
        description
          "Time interval after which the BGP session will be
           reestablished.";
      }
    }
    container bgp-timers {
      description
        "Includes two BGP timers.";
      leaf keepalive {
        type uint16 {
          range "0..21845";
        }
        units "seconds";
        description
          "This timer indicates the KEEPALIVE messages' frequency
           between a PE and a BGP peer.

           If set to '0', it indicates that KEEPALIVE messages are
           disabled.

           It is suggested that the maximum time between KEEPALIVE
           messages be one-third of the Hold Time interval.";
        reference
          "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
                     Section 4.4";
      }
      leaf hold-time {
        type uint16 {
          range "0 | 3..65535";
        }
        units "seconds";
```

```
        description
          "Indicates the maximum number of seconds that may elapse
           between the receipt of successive KEEPALIVE and/or UPDATE
           messages from the peer.

           The Hold Time must be either zero or at least three
           seconds.";
        reference
          "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
                     Section 4.2";
      }
    }
  }

  grouping bgp-base-peer-group {
    description
      "Grouping for a basic BGP peer group.";
    leaf name {
      type string;
      description
        "Name of the BGP peer group.";
    }
    uses bgp-base;
  }

  grouping bgp-base-peer-group-list {
    description
      "Grouping for a list of basic BGP peer groups.";
    list peer-group {
      key "name";
      description
        "List of BGP peer groups uniquely identified by a name.";
      uses bgp-base-peer-group;
    }
  }

  grouping bgp-peer-group {
    description
      "Grouping for BGP peer group.";
    leaf name {
      type string;
      description
        "Name of the BGP peer group";
    }
    leaf local-address {
      type union {
        type inet:ip-address;
        type if:interface-ref;
      }
      description
        "Sets the local IP address to use for the BGP transport
         session.  This may be expressed as either an IP
         address or a reference to an interface.";
    }
    uses bgp-base;
    uses ac-common:bgp-authentication;
  }
```

```
  grouping bgp-peer-group-list {
    description
      "Grouping for a list of BGP peer groups.";
    list peer-group {
      key "name";
      description
        "List of BGP peer groups uniquely identified by a name.";
      uses bgp-peer-group;
    }
  }

  // RIP base parameters

  grouping rip-base {
    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 timers {
      description
        "Indicates the RIP timers.";
      reference
        "RFC 2080: RIPng for IPv6
         RFC 2453: RIP Version 2";
      leaf update-interval {
        type uint16 {
          range "1..32767";
        }
        units "seconds";
        description
          "Indicates the RIP update time, i.e., the amount of time
           for which RIP updates are sent.";
      }
      leaf invalid-interval {
        type uint16 {
          range "1..32767";
        }
        units "seconds";
        description
          "The interval before a route is declared invalid after no
           updates are received.  This value is at least three times
           the value for the 'update-interval' argument.";
      }
      leaf holddown-interval {
        type uint16 {
          range "1..32767";
        }
        units "seconds";
        description
          "Specifies the interval before better routes are
           released.";
      }
```

```
      leaf flush-interval {
        type uint16 {
          range "1..32767";
        }
        units "seconds";
        description
          "Indicates the RIP flush timer, i.e., the amount of time
           that must elapse before a route is removed from the
           routing table.";
      }
    }
    leaf default-metric {
      type uint8 {
        range "0..16";
      }
      description
        "Sets the default metric.";
    }
  }

  // Routing profile

  grouping routing-profile {
    description
      "Defines profiles for 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.";
      }
      container bgp {
        when "derived-from-or-self(../type, "
           + "'vpn-common:bgp-routing')" {
          description
            "Only applies when the protocol is BGP.";
        }
        if-feature "vpn-common:rtg-bgp";
        description
          "Configuration specific to BGP.";
        container peer-groups {
          description
            "Lists a set of BGP peer groups.";
          uses bgp-base-peer-group-list;
        }
      }
      container ospf {
        when "derived-from-or-self(../type, "
```

```
           + "'vpn-common:ospf-routing')" {
          description
            "Only applies when the protocol is OSPF.";
        }
        if-feature "vpn-common:rtg-ospf";
        description
          "Configuration specific to OSPF.";
        uses ac-common:ospf-basic;
        leaf max-lsa {
          type uint32 {
            range "1..4294967294";
          }
          description
            "Maximum number of allowed Link State Advertisements
             (LSAs) that the OSPF instance will accept.";
        }
        leaf passive {
          type boolean;
          description
            "When set to 'true', enables a passive interface.  It is
             active when set to 'false'.  A passive interface's
             prefix will be advertised, but no neighbor adjacencies
             will be formed on the interface.";
        }
      }
      container isis {
        when "derived-from-or-self(../type, "
           + "'vpn-common:isis-routing')" {
          description
            "Only applies when the protocol is IS-IS.";
        }
        if-feature "vpn-common:rtg-isis";
        description
          "Configuration specific to IS-IS.";
        uses ac-common:isis-basic;
        leaf level {
          type identityref {
            base vpn-common:isis-level;
          }
          description
            "Can be 'level-1', 'level-2', or 'level-1-2'.";
          reference
            "RFC 9181: A Common YANG Data Model for Layer 2
                       and Layer 3 VPNs";
        }
        leaf metric {
          type uint32 {
            range "0 .. 16777215";
          }
          description
            "Metric of the AC.  It is used in the routing state
             calculation and path selection.";
        }
        leaf passive {
          type boolean;
          description
            "When set to 'false', the interface is active.  In such
             mode, the interface sends or receives IS-IS protocol
```

```
            control packets.

            When set to 'true', the interface is passive.  That
            is, it suppresses the sending of IS-IS updates through
            the specified interface.";
        }
      }
      container rip {
        when "derived-from-or-self(../type, "
           + "'vpn-common:rip-routing')" {
          description
            "Only applies when the protocol is RIP.";
        }
        if-feature "vpn-common:rtg-rip";
        description
          "Configuration specific to RIP routing.";
        uses rip-base;
      }
      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).";
        }
        if-feature "vpn-common:rtg-vrrp";
        description
          "Configuration specific to VRRP.";
        reference
          "RFC 9568: 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.";
        }
        leaf ping-reply {
          type boolean;
          description
            "Controls whether the VRRP speaker should reply to ping
             requests.  Such behavior is enabled, if set to 'true'.";
        }
      }
    }
  }

  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-profile {
          key "routing-profile-ref";
          description
            "Routing profiles.";
          uses routing-profile-reference;
          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 static routing.";
          }
          description
            "Configuration specific to static routing.";
          container cascaded-lan-prefixes {
            description
              "LAN prefixes from the customer.";
            list ipv4-lan-prefix {
              if-feature "vpn-common:ipv4";
              key "lan next-hop";
              description
                "List of LAN prefixes for the site.";
              uses ac-common:ipv4-static-rtg-entry;
              uses bfd-routing;
              leaf preference {
                type uint32;
                description
                  "Indicates the preference associated with the static
                   route.";
              }
              uses ac-common:service-status;
            }
            list ipv6-lan-prefix {
              if-feature "vpn-common:ipv6";
              key "lan next-hop";
              description
                "List of LAN prefixes for the site.";
              uses ac-common:ipv6-static-rtg-entry;
              uses bfd-routing;
              leaf preference {
                type uint32;
```

```
            description
              "Indicates the preference associated with the static
               route.";
          }
          uses ac-common:service-status;
        }
      }
    }
    container bgp {
      when "derived-from-or-self(../type, "
         + "'vpn-common:bgp-routing')" {
        description
          "Only applies when the protocol is BGP.";
      }
      if-feature "vpn-common:rtg-bgp";
      description
        "Configuration specific to BGP.";
      container peer-groups {
        description
          "Configuration for BGP peer groups";
        uses bgp-peer-group-list;
      }
      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 union {
            type inet:ip-address;
            type if:interface-ref;
          }
          description
            "Sets the local IP address to use for the BGP transport
             session.  This may be expressed as either an IP
             address or a reference to an interface.";
        }
        leaf peer-group {
          type leafref {
            path "../../peer-groups/peer-group/name";
          }
          description
            "The peer group with which this neighbor is
             associated.";
        }
        uses bgp-base;
        uses bfd-routing;
        uses ac-common:bgp-authentication;
        uses ac-common:service-status;
      }
    }
    container ospf {
      when "derived-from-or-self(../type, "
         + "'vpn-common:ospf-routing')" {
```

```
          description
            "Only applies when the protocol is OSPF.";
        }
        if-feature "vpn-common:rtg-ospf";
        description
          "Configuration specific to OSPF.";
        uses ac-common:ospf-basic;
        container sham-links {
          if-feature "vpn-common:rtg-ospf-sham-link";
          description
            "List of sham links.";
          reference
            "RFC 4577: OSPF as the Provider/Customer Edge Protocol
                       for BGP/MPLS IP Virtual Private Networks
                       (VPNs), Section 4.2.7
             RFC 6565: OSPFv3 as a Provider Edge to Customer Edge
                       (PE-CE) Routing Protocol, Section 5";
          list sham-link {
            key "target-site";
            description
              "Creates a sham link with another site.";
            leaf target-site {
              type string;
              description
                "Target site for the sham link connection.  The site
                 is referred to by its identifier.";
            }
            leaf metric {
              type uint16;
              description
                "Metric of the sham link.  It is used in the routing
                 state calculation and path selection.";
              reference
                "RFC 4577: OSPF as the Provider/Customer Edge
                           Protocol for BGP/MPLS IP Virtual Private
                           Networks (VPNs), Section 4.2.7.3
                 RFC 6565: OSPFv3 as a Provider Edge to Customer Edge
                           (PE-CE) Routing Protocol, Section 5.2";
            }
          }
        }
        leaf max-lsa {
          type uint32 {
            range "1..4294967294";
          }
          description
            "Maximum number of allowed Link State Advertisements
             (LSAs) that the OSPF instance will accept.";
        }
        leaf passive {
          type boolean;
          description
            "When set to 'true', enables a passive interface.  It is
             active when set to 'false'.  A passive interface's
             prefix will be advertised, but no neighbor adjacencies
             will be formed on the interface.";
        }
        uses ac-common:ospf-authentication;
```

```
        uses ac-common:service-status;
      }
      container isis {
        when "derived-from-or-self(../type, "
           + "'vpn-common:isis-routing')" {
          description
            "Only applies when the protocol is IS-IS.";
        }
        if-feature "vpn-common:rtg-isis";
        description
          "Configuration specific to IS-IS.";
        uses ac-common:isis-basic;
        leaf level {
          type identityref {
            base vpn-common:isis-level;
          }
          description
            "Can be 'level-1', 'level-2', or 'level-1-2'.";
          reference
            "RFC 9181: A Common YANG Data Model for Layer 2 and
                       Layer 3 VPNs";
        }
        leaf metric {
          type uint32 {
            range "0 .. 16777215";
          }
          description
            "Metric of the AC.  It is used in the routing state
             calculation and path selection.";
        }
        leaf passive {
          type boolean;
          description
            "When set to 'false', the interface is active.  In such
             mode, the interface sends or receives IS-IS protocol
             control packets.

             When set to 'true', the interface is passive.  That
             is, it suppresses the sending of IS-IS updates through
             the specified interface.";
        }
        uses ac-common:isis-authentication;
        uses ac-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.";
        }
        if-feature "vpn-common:rtg-rip";
        description
          "Configuration specific to RIP routing.";
        uses rip-base;
        uses ac-common:rip-authentication;
        uses ac-common:service-status;
```

```
      }
      container vrrp {
        when "derived-from-or-self(../type, "
           + "'vpn-common:vrrp-routing')" {
          description
            "Only applies when the protocol is VRRP.";
        }
        if-feature "vpn-common:rtg-vrrp";
        description
          "Configuration specific to VRRP.";
        reference
          "RFC 9568: 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.";
        }
        leaf vrrp-group {
          type uint8 {
            range "1..255";
          }
          description
            "Includes the VRRP group identifier.";
        }
        leaf backup-peer {
          type inet:ip-address;
          description
            "Indicates the IP address of the peer.";
        }
        leaf-list virtual-ip-address {
          type inet:ip-address;
          description
            "Virtual IP addresses for a single VRRP group.";
          reference
            "RFC 9568: Virtual Router Redundancy Protocol (VRRP)
                       Version 3 for IPv4 and IPv6, Sections 1.2
                       and 1.3";
        }
        leaf priority {
          type uint8 {
            range "1..254";
          }
          description
            "Sets the local priority of the VRRP speaker.";
        }
        leaf ping-reply {
          type boolean;
          description
            "Controls whether the VRRP speaker should reply to ping
             requests.";
        }
        uses ac-common:service-status;
      }
    }
```

```
  }

  // OAM

  grouping bfd {
    description
      "Grouping for BFD.";
    leaf session-type {
      type identityref {
        base vpn-common:bfd-session-type;
      }
      description
        "Specifies the BFD session type.";
    }
    leaf desired-min-tx-interval {
      type uint32;
      units "microseconds";
      description
        "The minimum interval between transmissions of BFD Control
         packets, as desired by the operator.";
      reference
        "RFC 5880: Bidirectional Forwarding Detection (BFD),
                   Section 6.8.7";
    }
    leaf required-min-rx-interval {
      type uint32;
      units "microseconds";
      description
        "The minimum interval between received BFD Control packets
         that the PE should support.";
      reference
        "RFC 5880: Bidirectional Forwarding Detection (BFD),
                   Section 6.8.7";
    }
    leaf local-multiplier {
      type uint8 {
        range "1..255";
      }
      description
        "Specifies the detection multiplier that is transmitted to a
         BFD peer.

         The detection interval for the receiving BFD peer is
         calculated by multiplying the value of the negotiated
         transmission interval by the received detection multiplier
         value.";
      reference
        "RFC 5880: Bidirectional Forwarding Detection (BFD),
                   Section 6.8.7";
    }
    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.";
      reference
        "RFC 5880: Bidirectional Forwarding Detection (BFD),
                   Section 6.8.18";
    }
  }

  grouping bfd-routing {
    description
      "Defines a basic BFD grouping for routing configuration.";
    container bfd {
      if-feature "vpn-common:bfd";
      description
        "BFD control for this neighbor.";
      leaf enabled {
        type boolean;
        description
          "Enables BFD if set to 'true'. BFD is disabled if set to
           'false'.";
      }
      uses failure-detection-profile-reference;
    }
  }

  grouping oam {
    description
      "Defines the Operations, Administration, and Maintenance
       (OAM) mechanisms used.";
    container bfd {
      if-feature "vpn-common:bfd";
      description
        "Container for BFD.";
      list session {
        key "dest-addr";
        description
          "List of IP sessions.";
        leaf dest-addr {
          type inet:ip-address;
          description
            "IP address of the peer.";
        }
        leaf source-address {
          type union {
            type inet:ip-address;
            type if:interface-ref;
          }
          description
            "Sets the local IP address to use for the BFD session.
             This may be expressed as either an IP address or
             a reference to an interface.";
        }
        uses failure-detection-profile-reference;
        uses bfd;
        container authentication {
          presence "Enables BFD authentication";
          description
            "Parameters for BFD authentication.";
          leaf key-chain {
```

```
            type key-chain:key-chain-ref;
            description
              "Name of the key chain.";
          }
          leaf meticulous {
            type boolean;
            description
              "Enables meticulous mode, if set to 'true'.";
            reference
              "RFC 5880: Bidirectional Forwarding Detection (BFD),
                         Section 6.7";
          }
        }
        uses ac-common:service-status;
      }
    }
  }

  // Security

  grouping security {
    description
      "Security parameters for an AC.";
    container encryption {
      if-feature "vpn-common:encryption";
      description
        "Container for AC encryption.";
      leaf enabled {
        type boolean;
        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 provider-profile {
          uses encryption-profile-reference;
        }
        case customer-profile {
          leaf customer-key-chain {
            type key-chain:key-chain-ref;
            description
              "Customer-supplied key chain.";
          }
        }
      }
    }
  }

  // AC profile

  grouping ac-profile {
    description
      "Grouping for attachment circuit profiles.";
    container routing-protocols {
      description
        "Defines routing protocols.";
      uses routing-profile;
    }
    container oam {
      description
        "Defines the OAM mechanisms used for the AC profile.";
      container bfd {
        if-feature "vpn-common:bfd";
        description
          "Container for BFD.";
        uses bfd;
      }
    }
  }

  // Parent and Child ACs

  grouping ac-hierarchy {
    description
      "Container for parent and child AC references.";
    container parent-ref {
      description
        "Specifies the parent AC that is inherited by an AC.
         Parent ACs are used, e.g., in contexts where multiple
         CEs are terminating the same AC, but some specific
         information is required for each peer SAP.";
      uses ac-ntw:attachment-circuit-reference;
    }
    container child-ref {
      config false;
      description
```

```
        "Specifies a child AC that relies upon a parent AC.";
      uses ac-ntw:attachment-circuit-references;
    }
  }

  // AC network provisioning

  grouping ac {
    description
      "Grouping for attachment circuits.";
    leaf description {
      type string;
      description
        "Associates a description with an AC.";
    }
    container l2-connection {
      if-feature "ac-common:layer2-ac";
      description
        "Defines Layer 2 protocols and parameters that are required
         to enable AC connectivity.";
      uses l2-connection-if-ref;
    }
    container ip-connection {
      if-feature "ac-common:layer3-ac";
      description
        "Defines IP connection parameters.";
      uses ip-connection;
    }
    container routing-protocols {
      description
        "Defines routing protocols.";
      uses routing;
    }
    container oam {
      description
        "Defines the OAM mechanisms used for the AC.";
      uses oam;
    }
    container security {
      description
        "AC-specific security parameters.";
      uses security;
    }
    container service {
      description
        "AC-specific bandwidth parameters.";
      leaf mtu {
        type uint32;
        units "bytes";
        description
          "Layer 2 MTU.";
      }
      uses ac-svc:bandwidth;
      container qos {
        if-feature "vpn-common:qos";
        description
          "QoS configuration.";
        container qos-profiles {
```

```
          description
            "QoS profile configuration.";
          list qos-profile {
            key "qos-profile-ref";
            description
              "Points to a QoS profile.";
            uses qos-profile-reference;
            leaf direction {
              type identityref {
                base vpn-common:qos-profile-direction;
              }
              description
                "The direction to which the QoS profile is applied.";
            }
          }
        }
      }
      container access-control-list {
        description
          "Container for the Access Control List (ACL).";
        container acl-profiles {
          description
            "ACL profile configuration.";
          list acl-profile {
            key "forwarding-profile-ref";
            description
              "Points to an ACL profile.";
            uses forwarding-profile-reference;
          }
        }
      }
    }
  }

  augment "/nw:networks/nw:network" {
    description
      "Add a list of profiles.";
    container specific-provisioning-profiles {
      description
        "Contains a set of valid profiles to reference in the AC
         activation.";
      uses ac-common:ac-profile-cfg;
    }
    list ac-profile {
      key "name";
      description
        "Specifies a list of AC profiles.";
      leaf name {
        type string;
        description
          "Name of the AC.";
      }
      uses ac-ntw:ac-profile;
    }
  }

  augment "/nw:networks/nw:network/nw:node" {
    when '../nw:network-types/sap:sap-network' {
```

```
      description
        "Augmentation parameters apply only for SAP networks.";
    }
    description
      "Augments nodes with AC provisioning details.";
    list ac {
      key "name";
      description
        "List of ACs.";
      leaf name {
        type string;
        description
          "A name that identifies the AC locally.";
      }
      leaf svc-ref {
        type ac-svc:attachment-circuit-reference;
        description
          "A reference to the AC as exposed at the service level.";
      }
      list profile {
        key "ac-profile-ref";
        description
          "List of AC profiles.";
        uses ac-profile-reference;
      }
      uses ac-hierarchy;
      leaf-list peer-sap-id {
        type string;
        description
          "One or more peer SAPs can be indicated.";
      }
      uses ac-common:redundancy-group;
      uses ac-common:service-status;
      uses ac-ntw:ac;
    }
  }

  augment "/nw:networks/nw:network/nw:node"
        + "/sap:service/sap:sap" {
    when '../../../nw:network-types/sap:sap-network' {
      description
        "Augmentation parameters apply only for SAP networks.";
    }
    description
      "Augments SAPs with AC provisioning details.";
    list ac {
      key "ac-ref";
      description
        "Specifies the ACs that are terminated by the SAP.";
      uses ac-ntw:attachment-circuit-reference;
    }
  }
}

<CODE ENDS>
```

7. Security Considerations

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

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

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

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). All writable data nodes are likely to be reasonably 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. The following subtrees and data nodes have particular sensitivities/vulnerabilities:

'specific-provisioning-profiles': This container includes a set of sensitive data that influences how an AC is delivered. For example, an attacker who has access to these data nodes may be able to manipulate routing policies, QoS policies, or encryption properties. These data nodes are defined with "nacm:default-deny- write" tagging [RFC9833].

'ac': An attacker who is able to access network nodes can undertake various attacks, such as modify the attributes of an AC (e.g., QoS, bandwidth, routing protocols, keying material), leading to malfunctioning of services that are delivered over that AC and therefore to Service Level Agreement (SLA) violations. In addition, an attacker could attempt to add a new AC. By also using NACM to prevent unauthorized access, such activity can be detected by adequately monitoring and tracking network configuration changes.

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. Specifically, the following subtrees and data nodes have particular sensitivities/vulnerabilities:

'ac': Unauthorized access to this subtree can disclose the identity of a customer 'peer-sap-id'.

'l2-connection' and 'ip-connection': An attacker can retrieve privacy-related information, which can be used to track a customer. Disclosing such information may be considered a violation of the customer-provider trust relationship.

'keying-material' and 'customer-key-chain': An attacker can retrieve the cryptographic keys protecting an AC (routing, in particular). These keys could be used to inject spoofed routing advertisements.

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

Section 5.8 specifies the encryption to be applied to traffic for a given AC.

8. IANA Considerations

IANA has registered the following URI in the "ns" subregistry within the "IETF XML Registry" [RFC3688]:

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

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

Name: ietf-ac-ntw
Maintained by IANA? N
Namespace: urn:ietf:params:xml:ns:yang:ietf-ac-ntw
Prefix: ac-ntw
Reference: RFC 9835

9. References

9.1. Normative References

[IEEE802.1Qcp] IEEE, "IEEE Standard for Local and metropolitan area networks--Bridges and Bridged Networks--Amendment 30: YANG Data Model", IEEE Std 802.1Qcp-2018, DOI 10.1109/IEEESTD.2018.8467507, September 2018, <https://doi.org/10.1109/IEEESTD.2018.8467507>.

[RFC2080] Malkin, G. and R. Minnear, "RIPng for IPv6", RFC 2080, DOI 10.17487/RFC2080, January 1997, <https://www.rfc-editor.org/info/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/info/rfc2119>.

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

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

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

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

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

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

[RFC5701] Rekhter, Y., "IPv6 Address Specific BGP Extended Community Attribute", RFC 5701, DOI 10.17487/RFC5701, November 2009, <https://www.rfc-editor.org/info/rfc5701>.

[RFC5709] Bhatia, M., Manral, V., Fanto, M., White, R., Barnes, M., Li, T., and R. Atkinson, "OSPFv2 HMAC-SHA Cryptographic Authentication", RFC 5709, DOI 10.17487/RFC5709, October 2009, <https://www.rfc-editor.org/info/rfc5709>.

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

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

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

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

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

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

[RFC7166] Bhatia, M., Manral, V., and A. Lindem, "Supporting Authentication Trailer for OSPFv3", RFC 7166, DOI 10.17487/RFC7166, March 2014, <https://www.rfc-editor.org/info/rfc7166>.

[RFC7474] Bhatia, M., Hartman, S., Zhang, D., and A. Lindem, Ed., "Security Extension for OSPFv2 When Using Manual Key Management", RFC 7474, DOI 10.17487/RFC7474, April 2015, <https://www.rfc-editor.org/info/rfc7474>.

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

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

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

[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/info/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/info/rfc8177>.

[RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, DOI 10.17487/RFC8294, December 2017, <https://www.rfc-editor.org/info/rfc8294>.

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

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

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

[RFC8345] Clemm, A., Medved, J., Varga, R., Bahadur, N., Ananthakrishnan, H., and X. Liu, "A YANG Data Model for Network Topologies", RFC 8345, DOI 10.17487/RFC8345, March 2018, <https://www.rfc-editor.org/info/rfc8345>.

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

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

[RFC9067] Qu, Y., Tantsura, J., Lindem, A., and X. Liu, "A YANG Data Model for Routing Policy", RFC 9067, DOI 10.17487/RFC9067, October 2021, <https://www.rfc-editor.org/info/rfc9067>.

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

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

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

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

[RFC9568] Lindem, A. and A. Dogra, "Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6", RFC 9568, DOI 10.17487/RFC9568, April 2024, <https://www.rfc-editor.org/info/rfc9568>.

[RFC9833] Boucadair, M., Ed., Roberts, R., Ed., Gonzalez de Dios, O., Barguil Giraldo, S., and B. Wu, "A Common YANG Data Model for Attachment Circuits", RFC 9833, DOI 10.17487/RFC9833, August 2025, <https://www.rfc-editor.org/info/rfc9833>.

[RFC9834] Boucadair, M., Ed., Roberts, R., Ed., Gonzalez de Dios, O., Barguil, S., and B. Wu, "YANG Data Models for Bearers and 'Attachment Circuits'-as-a-Service (ACaaS)", RFC 9834, DOI 10.17487/RFC9834, August 2025, <https://www.rfc-editor.org/info/rfc9834>.

9.2. 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/info/rfc3644>.

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

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

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

[RFC7880] Pignataro, C., Ward, D., Akiya, N., Bhatia, M., and S. Pallagatti, "Seamless Bidirectional Forwarding Detection (S-BFD)", RFC 7880, DOI 10.17487/RFC7880, July 2016, <https://www.rfc-editor.org/info/rfc7880>.

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

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

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

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

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

[RFC9127] Rahman, R., Ed., Zheng, L., Ed., Jethanandani, M., Ed., Pallagatti, S., and G. Mirsky, "YANG Data Model for Bidirectional Forwarding Detection (BFD)", RFC 9127, DOI 10.17487/RFC9127, October 2021, <https://www.rfc-editor.org/info/rfc9127>.

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

[RFC9543] Farrel, A., Ed., Drake, J., Ed., Rokui, R., Homma, S., Makhijani, K., Contreras, L., and J. Tantsura, "A Framework for Network Slices in Networks Built from IETF Technologies", RFC 9543, DOI 10.17487/RFC9543, March 2024, <https://www.rfc-editor.org/info/rfc9543>.

[RFC9836] Boucadair, M., Ed., Roberts, R., Barguil, S., and O. Gonzalez de Dios, "A YANG Data Model for Augmenting VPN Service and Network Models with Attachment Circuits", RFC 9836, DOI 10.17487/RFC9836, August 2025, <https://www.rfc-editor.org/info/rfc9836>.

[YANG-GUIDELINES] Bierman, A., Boucadair, M., and Q. Wu, "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", Work in Progress, Internet-Draft, draft-ietf-netmod-rfc8407bis-28, 5 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-netmod-rfc8407bis-28>.

Appendix A. Examples

A.1. VPLS

Let us consider the example depicted in Figure 21 with two customer terminating points (CE1 and CE2). Let us also assume that the bearers to attach these CEs to the provider network are already in place. References to identify these bearers are shown in the figure.

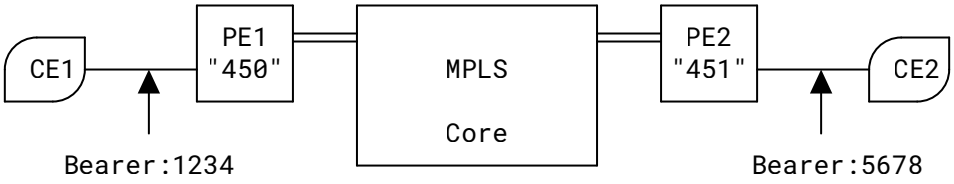

Figure 21: Topology Example

The AC service model [RFC9834] can be used by the provider to manage and expose the ACs over existing bearers as shown in Figure 22.

```
{
  "ietf-ac-svc:attachment-circuits": {
    "ac-group-profile": [
      {
        "name": "an-ac-profile",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "tag-type": "ietf-vpn-common:c-vlan",
              "cvlan-id": 550
            }
          }
        },
        "service": {
          "mtu": 1550,
          "svc-pe-to-ce-bandwidth": {
            "bandwidth": [
              {
                "bw-type": "ietf-vpn-common:bw-per-port",
                "cir": "20480000"
              }
            ]
          },
          "svc-ce-to-pe-bandwidth": {
            "bandwidth": [
              {
                "bw-type": "ietf-vpn-common:bw-per-port",
                "cir": "20480000"
              }
            ]
          },
          "qos": {
            "qos-profiles": {
              "qos-profile": [
                {
                  "profile": "QoS_Profile_A",
                  "direction": "ietf-vpn-common:both"
                }
              ]
            }
          }
        }
      }
    ],
    "ac": [
      {
        "name": "ac-1",
        "description": "First attachment",
        "ac-group-profile": [
          "an-ac-profile"
        ],
        "l2-connection": {
          "bearer-reference": "1234"
        }
      },
      {
```

```
        "name": "ac-2",
        "description": "Second attachment",
        "ac-group-profile": [
          "an-ac-profile"
        ],
        "l2-connection": {
          "bearer-reference": "5678"
        }
      }
    ]
  }
}
```

Figure 22: ACs Created Using ACaaS

The provisioned AC at PE1 can be retrieved using the AC network model as depicted in Figure 23. A similar query can be used for the AC at PE2.

```
{
   "ietf-ac-ntw:ac":[
      {
         "name":"ac-11",
         "svc-ref":"ac-1",
         "peer-sap-id":[
            "ce-1"
         ],
         "status":{
            "admin-status":{
               "status":"ietf-vpn-common:admin-up"
            },
            "oper-status":{
               "status":"ietf-vpn-common:op-up"
            }
         },
         "l2-connection":{
            "encapsulation":{
               "encap-type":"ietf-vpn-common:dot1q",
               "dot1q":{
                  "tag-type":"ietf-vpn-common:c-vlan",
                  "cvlan-id":550
               }
            },
            "bearer-reference":"1234"
         },
         "service":{
            "mtu":1550,
            "svc-pe-to-ce-bandwidth":{
               "bandwidth":[
                  {
                     "bw-type": "ietf-vpn-common:bw-per-port",
                     "cir":"20480000"
                  }
               ]
            },
            "svc-ce-to-pe-bandwidth":{
               "bandwidth":[
                  {
                     "bw-type": "ietf-vpn-common:bw-per-port",
                     "cir":"20480000"
                  }
               ]
            },
            "qos":{
               "qos-profiles":{
                  "qos-profile":[
                     {
                        "qos-profile-ref":"QoS_Profile_A",
                        "network-ref":"example:an-id",
                        "direction":"ietf-vpn-common:both"
                     }
                  ]
               }
            }
         }
      }
```

```
    ]
}
```

Figure 23: Example of AC Network Response (Message Body)

Also, the AC network model can be used to retrieve the list of SAPs to which the ACs are bound as shown in Figure 23.

```
{
   "ietf-sap-ntw:service":[
      {
         "service-type":"ietf-vpn-common:vpls",
         "sap":[
            {
               "sap-id":"sap#1",
               "peer-sap-id":[
                  "ce-1"
               ],
               "description":"A parent SAP",
               "attachment-interface":"GE0/6/1",
               "interface-type":"ietf-sap-ntw:phy",
               "role":"ietf-sap-ntw:uni",
               "allows-child-saps":true,
               "sap-status":{
                  "status":"ietf-vpn-common:op-up"
               }
            },
            {
               "sap-id":"sap#11",
               "description":"A child SAP",
               "parent-termination-point":"GE0/6/4",
               "attachment-interface":"GE0/6/4.2",
               "interface-type":"ietf-sap-ntw:logical",
               "encapsulation-type":"ietf-vpn-common:vlan-type",
               "sap-status":{
                  "status":"ietf-vpn-common:op-up"
               },
               "ietf-ac-ntw:ac":[
                  {
                     "ac-ref":"ac-1",
                     "node-ref":"example:pe2",
                     "network-ref":"example:an-id"
                  }
               ]
            }
         ]
      }
   ]
}
```

Figure 24: Example of AC Network Response to Retrieve the SAP (Message Body)

A.2. Parent AC

In reference to the topology depicted in Figure 1, PE2 has a SAP that terminates an AC with two peer SAPs (CE2 and CE5). In order to control data that is specific to each of these peer SAPs over the same AC, child ACs can be instantiated as depicted in Figure 25.

```
{
   "ietf-ac-ntw:ac":[
      {
         "name":"ac-1",
         "peer-sap-id":[
            "CE2",
            "CE5"
         ],
         "status":{
            "admin-status":{
               "status":"ietf-vpn-common:admin-up"
            },
            "oper-status":{
               "status":"ietf-vpn-common:op-up"
            }
         },
         "l2-connection":{
            "encapsulation":{
               "encap-type":"ietf-vpn-common:dot1q",
               "dot1q":{
                  "tag-type":"ietf-vpn-common:c-vlan",
                  "cvlan-id":550
               }
            },
            "bearer-reference":"1234"
         }
      },
      {
         "name":"ac-1-to-ce2",
         "parent-ref":{
            "ac-ref":"ac-1",
            "node-ref":"example:pe2",
            "network-ref":"example:an-id"
         },
         "peer-sap-id":[
            "CE2"
         ]
      },
      {
         "name":"ac-1-to-ce5",
         "parent-ref":{
            "ac-ref":"ac-1",
            "node-ref":"example:pe2",
            "network-ref":"example:an-id"
         },
         "peer-sap-id":[
            "CE5"
         ]
      }
   ]
}
```

Figure 25: Example of Child ACs

Figure 26 shows how to bind the parent AC to a SAP.

```
{
   "ietf-sap-ntw:service":[
      {
         "service-type":"ietf-vpn-common:l3vpn",
         "sap":[
            {
               "sap-id":"sap#14587",
               "description":"A SAP",
               "parent-termination-point":"GE0/6/4",
               "attachment-interface":"GE0/6/4.2",
               "interface-type":"ietf-sap-ntw:logical",
               "encapsulation-type":"ietf-vpn-common:vlan-type",
               "sap-status":{
                  "status":"ietf-vpn-common:op-up"
               },
               "ietf-ac-ntw:ac":[
                  {
                     "ac-ref":"ac-1",
                     "node-ref":"example:pe2",
                     "network-ref":"example:an-id"
                  }
               ]
            }
         ]
      }
   ]
}
```

Figure 26: Example of Binding Parent ACs to SAPs

Appendix B. Full Tree

```
module: ietf-ac-ntw

  augment /nw:networks/nw:network:
    +--rw specific-provisioning-profiles
    |  +--rw valid-provider-identifiers
    |     +--rw encryption-profile-identifier* [id]
    |     |  +--rw id    string
    |     +--rw qos-profile-identifier* [id]
    |     |  +--rw id    string
    |     +--rw failure-detection-profile-identifier* [id]
    |     |  +--rw id    string
    |     +--rw forwarding-profile-identifier* [id]
    |     |  +--rw id    string
    |     +--rw routing-profile-identifier* [id]
    |        +--rw id    string
    +--rw ac-profile* [name]
       +--rw name                 string
       +--rw routing-protocols
       |  +--rw routing-protocol* [id]
       |     +--rw id      string
       |     +--rw type?   identityref
       |     +--rw bgp {vpn-common:rtg-bgp}?
```

```
        |     |  +--rw peer-groups
        |     |     +--rw peer-group* [name]
        |     |        +--rw name                        string
        |     |        +--rw description?                string
        |     |        +--rw apply-policy
        |     |        |  +--rw import-policy*             leafref
        |     |        |  +--rw default-import-policy?
        |     |        |  |       default-policy-type
        |     |        |  +--rw export-policy*             leafref
        |     |        |  +--rw default-export-policy?
        |     |        |          default-policy-type
        |     |        +--rw local-as?                   inet:as-number
        |     |        +--rw peer-as                     inet:as-number
        |     |        +--rw address-family?             identityref
        |     |        +--rw role?                       identityref
        |     |        +--rw multihop?                   uint8
        |     |        +--rw as-override?                boolean
        |     |        +--rw allow-own-as?               uint8
        |     |        +--rw prepend-global-as?          boolean
        |     |        +--rw send-default-route?         boolean
        |     |        +--rw site-of-origin?
        |     |        |       rt-types:route-origin
        |     |        +--rw ipv6-site-of-origin?
        |     |        |       rt-types:ipv6-route-origin
        |     |        +--rw redistribute-connected* [address-family]
        |     |        |  +--rw address-family    identityref
        |     |        |  +--rw enabled?          boolean
        |     |        +--rw bgp-max-prefix
        |     |        |  +--rw max-prefix?          uint32
        |     |        |  +--rw warning-threshold?   decimal64
        |     |        |  +--rw violate-action?      enumeration
        |     |        |  +--rw restart-timer?       uint32
        |     |        +--rw bgp-timers
        |     |           +--rw keepalive?   uint16
        |     |           +--rw hold-time?   uint16
        |     +--rw ospf {vpn-common:rtg-ospf}?
        |     |  +--rw address-family?   identityref
        |     |  +--rw area-id           yang:dotted-quad
        |     |  +--rw metric?           uint16
        |     |  +--rw max-lsa?          uint32
        |     |  +--rw passive?          boolean
        |     +--rw isis {vpn-common:rtg-isis}?
        |     |  +--rw address-family?   identityref
        |     |  +--rw area-address      area-address
        |     |  +--rw level?            identityref
        |     |  +--rw metric?           uint32
        |     |  +--rw passive?          boolean
        |     +--rw rip {vpn-common:rtg-rip}?
        |     |  +--rw address-family?   identityref
        |     |  +--rw timers
        |     |  |  +--rw update-interval?     uint16
        |     |  |  +--rw invalid-interval?    uint16
        |     |  |  +--rw holddown-interval?   uint16
        |     |  |  +--rw flush-interval?      uint16
        |     |  +--rw default-metric?   uint8
        |     +--rw vrrp {vpn-common:rtg-vrrp}?
        |        +--rw address-family?   identityref
        |        +--rw ping-reply?       boolean
```

```
        +--rw oam
           +--rw bfd {vpn-common:bfd}?
              +--rw session-type?               identityref
              +--rw desired-min-tx-interval?    uint32
              +--rw required-min-rx-interval?   uint32
              +--rw local-multiplier?           uint8
              +--rw holdtime?                   uint32
   augment /nw:networks/nw:network/nw:node:
     +--rw ac* [name]
        +--rw name                string
        +--rw svc-ref?            ac-svc:attachment-circuit-reference
        +--rw profile* [ac-profile-ref]
        |  +--rw ac-profile-ref    leafref
        |  +--rw network-ref?      -> /nw:networks/network/network-id
        +--rw parent-ref
        |  +--rw ac-ref?        leafref
        |  +--rw node-ref?      leafref
        |  +--rw network-ref?   -> /nw:networks/network/network-id
        +--ro child-ref
        |  +--ro ac-ref*        leafref
        |  +--ro node-ref?      leafref
        |  +--ro network-ref?   -> /nw:networks/network/network-id
        +--rw peer-sap-id*         string
        +--rw group* [group-id]
        |  +--rw group-id      string
        |  +--rw precedence?   identityref
        +--rw status
        |  +--rw admin-status
        |  |  +--rw status?        identityref
        |  |  +--ro last-change?   yang:date-and-time
        |  +--ro oper-status
        |     +--ro status?        identityref
        |     +--ro last-change?   yang:date-and-time
        +--rw description?         string
        +--rw l2-connection {ac-common:layer2-ac}?
        |  +--rw encapsulation
        |  |  +--rw encap-type?        identityref
        |  |  +--rw dot1q
        |  |  |  +--rw tag-type?         identityref
        |  |  |  +--rw cvlan-id?         uint16
        |  |  |  +--rw tag-operations
        |  |  |     +--rw (op-choice)?
        |  |  |     |  +--:(pop)
        |  |  |     |  |  +--rw pop?         empty
        |  |  |     |  +--:(push)
        |  |  |     |  |  +--rw push?        empty
        |  |  |     |  +--:(translate)
        |  |  |     |     +--rw translate?   empty
        |  |  |     +--rw tag-1?          dot1q-types:vlanid
        |  |  |     +--rw tag-1-type?     dot1q-types:dot1q-tag-type
        |  |  |     +--rw tag-2?          dot1q-types:vlanid
        |  |  |     +--rw tag-2-type?     dot1q-types:dot1q-tag-type
        |  |  +--rw priority-tagged
        |  |  |  +--rw tag-type?   identityref
        |  |  +--rw qinq
        |  |     +--rw tag-type?         identityref
        |  |     +--rw svlan-id?         uint16
        |  |     +--rw cvlan-id?         uint16
```

```
        |  |     +--rw tag-operations
        |  |        +--rw (op-choice)?
        |  |        |  +--:(pop)
        |  |        |  |  +--rw pop?         uint8
        |  |        |  +--:(push)
        |  |        |  |  +--rw push?        empty
        |  |        |  +--:(translate)
        |  |        |     +--rw translate?   uint8
        |  |        +--rw tag-1?          dot1q-types:vlanid
        |  |        +--rw tag-1-type?     dot1q-types:dot1q-tag-type
        |  |        +--rw tag-2?          dot1q-types:vlanid
        |  |        +--rw tag-2-type?     dot1q-types:dot1q-tag-type
        |  +--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 l2-termination-point?      string
        |  +--rw local-bridge-reference?    string
        |  +--rw bearer-reference?          string
        |  |       {ac-common:server-assigned-reference}?
        |  +--rw lag-interface {vpn-common:lag-interface}?
        |     +--rw lag-interface-id?   string
        |     +--rw member-link-list
        |        +--rw member-link* [name]
        |           +--rw name    string
        +--rw ip-connection {ac-common:layer3-ac}?
        |  +--rw l3-termination-point?   string
        |  +--rw ipv4 {vpn-common:ipv4}?
        |  |  +--rw local-address?
        |  |  |       inet:ipv4-address
        |  |  +--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
        |  |     |  |  +--:(service-type)
        |  |     |  |     +--rw (service-type)?
        |  |     |  |        +--:(relay)
        |  |     |  |           +--rw server-ip-address*
        |  |     |  |                   inet:ipv4-address
        |  |     |  +--rw (dhcp-relay)?
        |  |     |     +--:(customer-dhcp-servers)
        |  |     |        +--rw customer-dhcp-servers
        |  |     |           +--rw server-ip-address*
        |  |     |                   inet:ipv4-address
        |  |     +--:(static-addresses)
        |  |        +--rw address* [address-id]
        |  |           +--rw address-id                       string
        |  |           +--rw customer-address?
        |  |           |       inet:ipv4-address
        |  |           +--rw failure-detection-profile-ref?   leafref
        |  |           +--rw network-ref?
        |  |                   -> /nw:networks/network/network-id
        |  +--rw ipv6 {vpn-common:ipv6}?
        |     +--rw local-address?
        |     |       inet:ipv6-address
        |     +--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
        |        |  |  +--:(service-type)
        |        |  |     +--rw (service-type)?
        |        |  |        +--:(relay)
        |        |  |           +--rw server-ip-address*
        |        |  |                   inet:ipv6-address
        |        |  +--rw (dhcp-relay)?
        |        |     +--:(customer-dhcp-servers)
        |        |        +--rw customer-dhcp-servers
        |        |           +--rw server-ip-address*
        |        |                   inet:ipv6-address
        |        +--:(static-addresses)
        |           +--rw address* [address-id]
        |              +--rw address-id                       string
```

```
        |              +--rw customer-address?
        |              |       inet:ipv6-address
        |              +--rw failure-detection-profile-ref?   leafref
        |              +--rw network-ref?
        |                      -> /nw:networks/network/network-id
        +--rw routing-protocols
        |  +--rw routing-protocol* [id]
        |     +--rw id                string
        |     +--rw type?             identityref
        |     +--rw routing-profile* [routing-profile-ref]
        |     |  +--rw routing-profile-ref    leafref
        |     |  +--rw network-ref?
        |     |  |       -> /nw:networks/network/network-id
        |     |  +--rw type?                  identityref
        |     +--rw static
        |     |  +--rw cascaded-lan-prefixes
        |     |     +--rw ipv4-lan-prefix* [lan next-hop]
        |     |     |       {vpn-common:ipv4}?
        |     |     |  +--rw lan           inet:ipv4-prefix
        |     |     |  +--rw lan-tag?      string
        |     |     |  +--rw next-hop      union
        |     |     |  +--rw metric?       uint32
        |     |     |  +--rw bfd {vpn-common:bfd}?
        |     |     |  |  +--rw enabled?
        |     |     |  |  |       boolean
        |     |     |  |  +--rw failure-detection-profile-ref?
        |     |     |  |  |       leafref
        |     |     |  |  +--rw network-ref?
        |     |     |  |          -> /nw:networks/network/network-id
        |     |     |  +--rw preference?   uint32
        |     |     |  +--rw status
        |     |     |     +--rw admin-status
        |     |     |     |  +--rw status?        identityref
        |     |     |     |  +--ro last-change?   yang:date-and-time
        |     |     |     +--ro oper-status
        |     |     |        +--ro status?        identityref
        |     |     |        +--ro last-change?   yang:date-and-time
        |     |     +--rw ipv6-lan-prefix* [lan next-hop]
        |     |             {vpn-common:ipv6}?
        |     |        +--rw lan           inet:ipv6-prefix
        |     |        +--rw lan-tag?      string
        |     |        +--rw next-hop      union
        |     |        +--rw metric?       uint32
        |     |        +--rw bfd {vpn-common:bfd}?
        |     |        |  +--rw enabled?
        |     |        |  |       boolean
        |     |        |  +--rw failure-detection-profile-ref?
        |     |        |  |       leafref
        |     |        |  +--rw network-ref?
        |     |        |          -> /nw:networks/network/network-id
        |     |        +--rw preference?   uint32
        |     |        +--rw status
        |     |           +--rw admin-status
        |     |           |  +--rw status?        identityref
        |     |           |  +--ro last-change?   yang:date-and-time
        |     |           +--ro oper-status
        |     |              +--ro status?        identityref
        |     |              +--ro last-change?   yang:date-and-time
```

```
        |     +--rw bgp {vpn-common:rtg-bgp}?
        |     |  +--rw peer-groups
        |     |  |  +--rw peer-group* [name]
        |     |  |     +--rw name                       string
        |     |  |     +--rw local-address?             union
        |     |  |     +--rw description?               string
        |     |  |     +--rw apply-policy
        |     |  |     |  +--rw import-policy*            leafref
        |     |  |     |  +--rw default-import-policy?
        |     |  |     |  |       default-policy-type
        |     |  |     |  +--rw export-policy*            leafref
        |     |  |     |  +--rw default-export-policy?
        |     |  |     |          default-policy-type
        |     |  |     +--rw local-as?                  inet:as-number
        |     |  |     +--rw peer-as                    inet:as-number
        |     |  |     +--rw address-family?            identityref
        |     |  |     +--rw role?                      identityref
        |     |  |     +--rw multihop?                  uint8
        |     |  |     +--rw as-override?               boolean
        |     |  |     +--rw allow-own-as?              uint8
        |     |  |     +--rw prepend-global-as?         boolean
        |     |  |     +--rw send-default-route?        boolean
        |     |  |     +--rw site-of-origin?
        |     |  |     |       rt-types:route-origin
        |     |  |     +--rw ipv6-site-of-origin?
        |     |  |     |       rt-types:ipv6-route-origin
        |     |  |     +--rw redistribute-connected* [address-family]
        |     |  |     |  +--rw address-family    identityref
        |     |  |     |  +--rw enabled?          boolean
        |     |  |     +--rw bgp-max-prefix
        |     |  |     |  +--rw max-prefix?          uint32
        |     |  |     |  +--rw warning-threshold?   decimal64
        |     |  |     |  +--rw violate-action?      enumeration
        |     |  |     |  +--rw restart-timer?       uint32
        |     |  |     +--rw bgp-timers
        |     |  |     |  +--rw keepalive?   uint16
        |     |  |     |  +--rw hold-time?   uint16
        |     |  |     +--rw authentication
        |     |  |        +--rw enabled?           boolean
        |     |  |        +--rw keying-material
        |     |  |           +--rw (option)?
        |     |  |              +--:(ao)
        |     |  |              |  +--rw enable-ao?          boolean
        |     |  |              |  +--rw ao-keychain?
        |     |  |              |          key-chain:key-chain-ref
        |     |  |              +--:(md5)
        |     |  |              |  +--rw md5-keychain?
        |     |  |              |          key-chain:key-chain-ref
        |     |  |              +--:(explicit)
        |     |  |                 +--rw key-id?             uint32
        |     |  |                 +--rw key?                string
        |     |  |                 +--rw crypto-algorithm?
        |     |  |                         identityref
        |     |  +--rw neighbor* [remote-address]
        |     |     +--rw remote-address            inet:ip-address
        |     |     +--rw local-address?            union
        |     |     +--rw peer-group?
        |     |     |       -> ../../peer-groups/peer-group/name
```

```
        |     |     +--rw description?              string
        |     |     +--rw apply-policy
        |     |     |  +--rw import-policy*           leafref
        |     |     |  +--rw default-import-policy?
        |     |     |  |       default-policy-type
        |     |     |  +--rw export-policy*           leafref
        |     |     |  +--rw default-export-policy?
        |     |     |          default-policy-type
        |     |     +--rw local-as?                  inet:as-number
        |     |     +--rw peer-as                    inet:as-number
        |     |     +--rw address-family?            identityref
        |     |     +--rw role?                      identityref
        |     |     +--rw multihop?                  uint8
        |     |     +--rw as-override?               boolean
        |     |     +--rw allow-own-as?              uint8
        |     |     +--rw prepend-global-as?         boolean
        |     |     +--rw send-default-route?        boolean
        |     |     +--rw site-of-origin?
        |     |     |       rt-types:route-origin
        |     |     +--rw ipv6-site-of-origin?
        |     |     |       rt-types:ipv6-route-origin
        |     |     +--rw redistribute-connected* [address-family]
        |     |     |  +--rw address-family    identityref
        |     |     |  +--rw enabled?          boolean
        |     |     +--rw bgp-max-prefix
        |     |     |  +--rw max-prefix?          uint32
        |     |     |  +--rw warning-threshold?   decimal64
        |     |     |  +--rw violate-action?      enumeration
        |     |     |  +--rw restart-timer?       uint32
        |     |     +--rw bgp-timers
        |     |     |  +--rw keepalive?   uint16
        |     |     |  +--rw hold-time?   uint16
        |     |     +--rw bfd {vpn-common:bfd}?
        |     |     |  +--rw enabled?                          boolean
        |     |     |  +--rw failure-detection-profile-ref?   leafref
        |     |     |  +--rw network-ref?
        |     |     |          -> /nw:networks/network/network-id
        |     |     +--rw authentication
        |     |     |  +--rw enabled?           boolean
        |     |     |  +--rw keying-material
        |     |     |     +--rw (option)?
        |     |     |        +--:(ao)
        |     |     |        |  +--rw enable-ao?          boolean
        |     |     |        |  +--rw ao-keychain?
        |     |     |        |          key-chain:key-chain-ref
        |     |     |        +--:(md5)
        |     |     |        |  +--rw md5-keychain?
        |     |     |        |          key-chain:key-chain-ref
        |     |     |        +--:(explicit)
        |     |     |           +--rw key-id?             uint32
        |     |     |           +--rw key?                string
        |     |     |           +--rw crypto-algorithm?   identityref
        |     |     +--rw status
        |     |        +--rw admin-status
        |     |        |  +--rw status?        identityref
        |     |        |  +--ro last-change?   yang:date-and-time
        |     |        +--ro oper-status
        |     |           +--ro status?        identityref
```

```
|     |           +--ro last-change?   yang:date-and-time
|     +--rw ospf {vpn-common:rtg-ospf}?
|     |  +--rw address-family?   identityref
|     |  +--rw area-id           yang:dotted-quad
|     |  +--rw metric?           uint16
|     |  +--rw sham-links {vpn-common:rtg-ospf-sham-link}?
|     |  |  +--rw sham-link* [target-site]
|     |  |     +--rw target-site    string
|     |  |     +--rw metric?        uint16
|     |  +--rw max-lsa?          uint32
|     |  +--rw passive?          boolean
|     |  +--rw authentication
|     |  |  +--rw enabled?           boolean
|     |  |  +--rw keying-material
|     |  |     +--rw (option)?
|     |  |        +--:(auth-key-chain)
|     |  |        |  +--rw key-chain?
|     |  |        |          key-chain:key-chain-ref
|     |  |        +--:(auth-key-explicit)
|     |  |           +--rw key-id?             uint32
|     |  |           +--rw key?                string
|     |  |           +--rw crypto-algorithm?   identityref
|     |  +--rw status
|     |     +--rw admin-status
|     |     |  +--rw status?        identityref
|     |     |  +--ro last-change?   yang:date-and-time
|     |     +--ro oper-status
|     |        +--ro status?        identityref
|     |        +--ro last-change?   yang:date-and-time
|     +--rw isis {vpn-common:rtg-isis}?
|     |  +--rw address-family?   identityref
|     |  +--rw area-address      area-address
|     |  +--rw level?            identityref
|     |  +--rw metric?           uint32
|     |  +--rw passive?          boolean
|     |  +--rw authentication
|     |  |  +--rw enabled?           boolean
|     |  |  +--rw keying-material
|     |  |     +--rw (option)?
|     |  |        +--:(auth-key-chain)
|     |  |        |  +--rw key-chain?
|     |  |        |          key-chain:key-chain-ref
|     |  |        +--:(auth-key-explicit)
|     |  |           +--rw key-id?             uint32
|     |  |           +--rw key?                string
|     |  |           +--rw crypto-algorithm?   identityref
|     |  +--rw status
|     |     +--rw admin-status
|     |     |  +--rw status?        identityref
|     |     |  +--ro last-change?   yang:date-and-time
|     |     +--ro oper-status
|     |        +--ro status?        identityref
|     |        +--ro last-change?   yang:date-and-time
|     +--rw rip {vpn-common:rtg-rip}?
|     |  +--rw address-family?   identityref
|     |  +--rw timers
|     |  |  +--rw update-interval?     uint16
|     |  |  +--rw invalid-interval?    uint16
```

```
|     |  |  +--rw holddown-interval?   uint16
|     |  |  +--rw flush-interval?      uint16
|     |  +--rw default-metric?   uint8
|     |  +--rw authentication
|     |  |  +--rw enabled?           boolean
|     |  |  +--rw keying-material
|     |  |     +--rw (option)?
|     |  |        +--:(auth-key-chain)
|     |  |        |  +--rw key-chain?
|     |  |        |          key-chain:key-chain-ref
|     |  |        +--:(auth-key-explicit)
|     |  |           +--rw key?                string
|     |  |           +--rw crypto-algorithm?   identityref
|     |  +--rw status
|     |     +--rw admin-status
|     |     |  +--rw status?        identityref
|     |     |  +--ro last-change?   yang:date-and-time
|     |     +--ro oper-status
|     |        +--ro status?        identityref
|     |        +--ro last-change?   yang:date-and-time
|     +--rw vrrp {vpn-common:rtg-vrrp}?
|        +--rw address-family?       identityref
|        +--rw vrrp-group?           uint8
|        +--rw backup-peer?          inet:ip-address
|        +--rw virtual-ip-address*   inet:ip-address
|        +--rw priority?             uint8
|        +--rw ping-reply?           boolean
|        +--rw status
|           +--rw admin-status
|           |  +--rw status?        identityref
|           |  +--ro 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 session* [dest-addr]
|        +--rw dest-addr              inet:ip-address
|        +--rw source-address?                 union
|        +--rw failure-detection-profile-ref?   leafref
|        +--rw network-ref?
|        |       -> /nw:networks/network/network-id
|        +--rw session-type?                   identityref
|        +--rw desired-min-tx-interval?        uint32
|        +--rw required-min-rx-interval?       uint32
|        +--rw local-multiplier?               uint8
|        +--rw holdtime?                       uint32
|        +--rw authentication!
|        |  +--rw key-chain?    key-chain:key-chain-ref
|        |  +--rw meticulous?   boolean
|        +--rw status
|           +--rw admin-status
|           |  +--rw status?        identityref
|           |  +--ro 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)?
        |        +--:(provider-profile)
        |        |  +--rw encryption-profile-ref?   leafref
        |        |  +--rw network-ref?
        |        |          -> /nw:networks/network/network-id
        |        +--:(customer-profile)
        |           +--rw customer-key-chain?
        |                   key-chain:key-chain-ref
        +--rw service
           +--rw mtu?                       uint32
           +--rw svc-pe-to-ce-bandwidth {vpn-common:inbound-bw}?
           |  +--rw bandwidth* [bw-type]
           |     +--rw bw-type      identityref
           |     +--rw (type)?
           |        +--:(per-cos)
           |        |  +--rw cos* [cos-id]
           |        |     +--rw cos-id    uint8
           |        |     +--rw cir?      uint64
           |        |     +--rw cbs?      uint64
           |        |     +--rw eir?      uint64
           |        |     +--rw ebs?      uint64
           |        |     +--rw pir?      uint64
           |        |     +--rw pbs?      uint64
           |        +--:(other)
           |           +--rw cir?   uint64
           |           +--rw cbs?   uint64
           |           +--rw eir?   uint64
           |           +--rw ebs?   uint64
           |           +--rw pir?   uint64
           |           +--rw pbs?   uint64
           +--rw svc-ce-to-pe-bandwidth {vpn-common:outbound-bw}?
           |  +--rw bandwidth* [bw-type]
           |     +--rw bw-type      identityref
           |     +--rw (type)?
           |        +--:(per-cos)
           |        |  +--rw cos* [cos-id]
           |        |     +--rw cos-id    uint8
           |        |     +--rw cir?      uint64
           |        |     +--rw cbs?      uint64
           |        |     +--rw eir?      uint64
           |        |     +--rw ebs?      uint64
           |        |     +--rw pir?      uint64
           |        |     +--rw pbs?      uint64
           |        +--:(other)
           |           +--rw cir?   uint64
           |           +--rw cbs?   uint64
           |           +--rw eir?   uint64
           |           +--rw ebs?   uint64
           |           +--rw pir?   uint64
           |           +--rw pbs?   uint64
           +--rw qos {vpn-common:qos}?
           |  +--rw qos-profiles
           |     +--rw qos-profile* [qos-profile-ref]
           |        +--rw qos-profile-ref    leafref
```

```
          |        +--rw network-ref?
          |        |       -> /nw:networks/network/network-id
          |        +--rw direction?         identityref
          +--rw access-control-list
             +--rw acl-profiles
                +--rw acl-profile* [forwarding-profile-ref]
                   +--rw forwarding-profile-ref    leafref
                   +--rw network-ref?
                           -> /nw:networks/network/network-id
augment /nw:networks/nw:network/nw:node/sap:service/sap:sap:
  +--rw ac* [ac-ref]
     +--rw ac-ref         leafref
     +--rw node-ref?      leafref
     +--rw network-ref?   -> /nw:networks/network/network-id
```

Acknowledgments

This document builds on [RFC9182] and [RFC9291].

Thanks to Moti Morgenstern for the review and comments.

Thanks to Martin Björklund for the YANG Doctors review, Gyan Mishra for an early RTGDIR review, Joel Halpern for the RTGDIR review, Giuseppe Fioccola for the OPSDIR review, and Russ Housley for the SECDIR review.

Thanks to Krzysztof Szarkowicz for the shepherd review.

Thanks for Mahesh Jethanandani for the AD review.

Contributors

Victor Lopez
Nokia
Email: victor.lopez@nokia.com

Ivan Bykov
Ribbon Communications
Email: Ivan.Bykov@rbbn.com

Qin Wu
Huawei
Email: bill.wu@huawei.com

Ogaki Kenichi
KDDI
Email: ke-oogaki@kddi.com

Luis Angel Munoz
Vodafone
Email: luis-angel.munoz@vodafone.com

Authors' Addresses

Mohamed Boucadair (EDITOR)
Orange
Email: mohamed.boucadair@orange.com

Richard Roberts
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