Stream: Internet Engineering Task Force (IETF)
RFC: 9834
Category: Standards Track
Published: September 2025
ISSN: 2070-1721
Authors:
M. Boucadair, Ed. *Orange*
R. Roberts, Ed. *Juniper*
O. Gonzalez de Dios *Telefonica*
S. Barguil *Nokia*
B. Wu *Huawei Technologies*

RFC 9834
YANG Data Models for Bearers and Attachment Circuits as a Service (ACaaS)

Abstract

Delivery of network services assumes that appropriate setup is provisioned over the links that connect customer termination points and a provider network. The required setup to allow successful data exchange over these links is referred to as an attachment circuit (AC), while the underlying link is referred to as a "bearer".

This document specifies a YANG service data model for ACs. This model can be used for the provisioning of ACs before or during service provisioning (e.g., RFC 9543 Network Slice Service).

The document also specifies a YANG service data model for managing bearers over which ACs are established.

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/rfc9834.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

1.1. Scope and Intended Use

Connectivity services are provided by networks to customers via dedicated termination points, such as Service Functions (SFs) [RFC7665], Customer Edges (CEs), peer Autonomous System Border Routers (ASBRs), data centers gateways, or Internet Exchange Points (IXPs). A connectivity service is basically about ensuring data transfer received from or destined to a given termination point to or from other termination points. The objectives for the connectivity service can be negotiated and agreed upon between the customer and the network provider. To facilitate data transfer within the provider network, it is assumed that the appropriate setup is provisioned over the links that connect customer termination points and a provider network (usually via a Provider Edge (PE)), allowing data to be successfully exchanged over these links. The required setup is referred to in this document as an attachment circuit (AC), while the underlying link is referred to as a "bearer".

When a customer requests a new service, the service can be bound to existing ACs or trigger the instantiation of new ACs. The provisioning of a service should, thus, accommodate both deployments.

Also, because the instantiation of an AC requires coordinating the provisioning of endpoints that might not belong to the same administrative entity (customer vs. provider or distinct operational teams within the same provider, etc.), providing programmatic means to expose Attachment Circuits as a Service (ACaaS) greatly simplifies the provisioning of services delivered over an AC. For example, management systems of adjacent domains that need to connect via an AC will use such means to agree upon the resources that are required for the activation of both sides of an AC (e.g., Layer 2 tags, IP address family, or IP subnets).

This document specifies a YANG service data model ("ietf-ac-svc") for managing ACs that are exposed by a network to its customers, such as an enterprise site, an SF, a hosting infrastructure, or a peer network provider. The model can be used for the provisioning of ACs prior to or during advanced service provisioning (e.g., RFC 9543 Network Slice Service defined in "A Framework for Network Slices in Networks Built from IETF Technologies" [RFC9543]).

The "ietf-ac-svc" module (Section 6.2) includes a set of reusable groupings. Whether a service model that wants to describe the ACs associated with the service reuses structures defined in the "ietf-ac-svc" or simply includes an AC reference (that was communicated during AC service instantiation) is a design choice of these service models. Relying upon the AC service model to manage ACs over which services are delivered has the merit of decorrelating the management of the (core) service from the ACs. This allows upgrades (to reflect recent AC technologies or new features such as new encryption schemes or additional routing protocols) to be done in just one place rather than in each (core) service model. This document favors the approach of completely relying upon the AC service model instead of duplicating data nodes into specific modules of advanced services that are delivered over an AC.

Since the provisioning of an AC requires a bearer to be in place, this document introduces a new module called "ietf-bearer-svc", which enables customers to manage their bearers (Section 6.1). The customers can then retrieve a provider-assigned bearer reference that they will include in their AC service requests. Likewise, a customer may learn whether their bearers support a synchronization mechanism such as Sync Ethernet (SyncE) [ITU-T-G.781]. An example of retrieving a bearer reference is provided in Appendix A.1.

An AC service request can provide a reference to a bearer or a set of peer Service Attachment Points (SAPs) specified in "A YANG Network Data Model for Service Attachment Points (SAPs)" [RFC9408]. Both schemes are supported in the AC service model. When several bearers are available, the AC service request may filter them based on the bearer type, synchronization support, etc.

Each AC is identified with a unique identifier within a provider domain. From a network provider standpoint, an AC can be bound to a single or multiple SAPs [RFC9408]. Likewise, the same SAP can be bound to one or multiple ACs. However, the mapping between an AC and a PE in the provider network that terminates that AC is hidden to the application that makes use of the AC service model. Such mapping information is internal to the network controllers. As such, the details about the (node-specific) attachment interfaces are not exposed in the AC service model. However, these details are exposed at the network model per "A Network YANG Data Model for Attachment Circuits" [RFC9835]. "A YANG Data Model for Augmenting VPN Service and Network Models with Attachment Circuits" [RFC9836] specifies augmentations to the L2VPN Service Model (L2SM) [RFC8466] and the L3VPN Service Model (L3SM) [RFC8299] to bind LxVPN services to ACs.

The AC service model does not make any assumptions about the internal structure or even the nature of the services that will be delivered over an AC or a set of ACs. Customers do not have access to that network view other than the ACs that they ordered. For example, the AC service model can be used to provision a set of ACs to connect multiple sites (Site1, Site2, ..., SiteX) for a customer who also requested VPN services. If the provisioning of these services requires specific configuration on ASBR nodes, such configuration is handled at the network level and is not exposed to the customer at the service level. However, the network controller will have access to such a view, as the service points in these ASBRs will be exposed as SAPs with 'role' set to 'ietf-sap-ntw:nni' [RFC9408].

The AC service model can be used in a variety of contexts, such as (but not limited to) those provided in Appendix A:

- Create an AC over an existing bearer (Appendix A.2).
- Request an AC for a known peer SAP (Appendix A.3).
- Instantiate multiple ACs over the same bearer (Appendix A.4).
- Control the precedence over multiple ACs (Appendix A.5).
- Create multiple ACs bound to multiple CEs (Appendix A.6).
- Bind an RFC 9543 Network Slice Service to a set of pre-provisioned ACs (Appendix A.7).
- Connect an enterprise network to a provider network using BGP (Appendix A.9).

- Connect a Cloud Infrastructure to a service provider network (Appendix A.8).
- Interconnect provider networks (e.g., [RFC8921] or [PEERING-API]). Such ACs are identified with a 'role' set to 'ac-common:nni' or 'ac-common:public-nni'. See Appendix A.10 to illustrate the use of the AC model for interconnection/peering.
- Manage connectivity for complex containerized or virtualized functions in the cloud (Appendix A.11).
- Manage AC redundancy with static addressing (Appendix A.12).

The document adheres to the principles discussed in "Service Models Explained" (Section 3 of [RFC8309]) for the encoding and communication protocols used for the interaction between a customer and a provider. Also, consistent with "A Framework for Automating Service and Network Management with YANG" [RFC8969], the service models defined in the document can be used independently of the Network Configuration Protocol (NETCONF) / RESTCONF.

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

1.2. Positioning ACaaS vs. Other Data Models

The AC model specified in this document is not a network model [RFC8969]. As such, the model does not expose details related to specific nodes in the provider's network that terminate an AC (e.g., network node identifiers). The mapping between an AC as seen by a customer and the network implementation of an AC is maintained by the network controllers and is not exposed to the customer. This mapping can be maintained using a variety of network models, such as an augmented SAP AC network model [RFC9835].

The AC service model is not a device model. A network provider may use a variety of device models (e.g., "A YANG Data Model for Routing Management (NMDA Version)" [RFC8349] or "YANG Model for Border Gateway Protocol (BGP-4)" [BGP4-YANG]) to provision an AC service in relevant network nodes.

The AC service model reuses common types and structures defined in "A Common YANG Data Model for Layer 2 and Layer 3 VPNs" [RFC9181].

1.2.1. Why Not Use the L2SM as a Reference Data Model for ACaaS?

The L2VPN Service Model (L2SM) [RFC8466] covers some AC-related considerations. Nevertheless, the L2SM structure is primarily focused on Layer 2 aspects. For example, the L2SM does not cover Layer 3 provisioning, which is required for the typical AC instantiation.

1.2.2. Why Not Use the L3SM as a Reference Data Model for ACaaS?

Like the L2SM, the L3VPN Service Model (L3SM) [RFC8299] addresses certain AC-related aspects. However, the L3SM structure does not sufficiently address Layer 2 provisioning requirements. Additionally, the L3SM is primarily designed for conventional L3VPN deployments and, as such, has some limitations for instantiating ACs in other deployment contexts (e.g., cloud environments). For example, the L3SM does not provide the capability to provision multiple BGP peer groups over the same AC.

2. Conventions and Definitions

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

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

LxSM refers to both the L2SM and the L3SM.

LxVPN refers to both Layer 2 VPN (L2VPN) and Layer 3 VPN (L3VPN).

LxNM refers to both the L2VPN Network Model (L2NM) and the L3VPN Network Model (L3NM).

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 (e.g., Link Aggregation Group (LAG) [IEEE802.1AX]). The bearer type can be specified by a customer.

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

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

One or multiple ACs may be hosted over the same bearer (e.g., multiple VLANs on the same bearer that is provided by a physical link).

Customer Edge (CE): Equipment that is dedicated to a customer and is connected to one or more PEs via ACs.

A CE can be a router, a bridge, a switch, etc.

Provider Edge (PE): Equipment owned and managed by the service provider that can support multiple services for different customers.

Per "Provider Provisioned Virtual Private Network (VPN) Terminology" (Section 5.2 of [RFC4026]), a PE is a device located at the edge of the service network with the functionality that is needed to interface with the customer.

A PE is connected to one or more CEs via ACs.

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.

Network Function (NF): Used to refer to the same concept as SF (Section 1.4 of [RFC7665]).

NF is also used in this document, as this term is widely used outside the IETF.

NF and SF are used interchangeably.

Parent Bearer: Refers to a bearer (e.g., LAG) that is used to build other bearers. These bearers (called child bearers) inherit the parent bearer properties.

Parent AC: Refers to an AC that is used to build other ACs. These ACs (called Child ACs) inherit the Parent AC properties.

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

A service orchestrator is typically responsible for the ACs, the PE selection, and requesting the activation of the requested service 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 RFC 9543 Network Slice Services).

Service provider: An entity that offers network services (e.g., LxVPN or RFC 9543 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
inet	ietf-inet-types	Section 4 of [RFC6991]
key-chain	ietf-key-chain	[RFC8177]
nacm	ietf-netconf-acm	[RFC8341]
vpn-common	ietf-vpn-common	[RFC9181]

Table 1: Modules and Their Associated Prefixes

3. Relationship to Other AC Data Models

Figure 1 depicts the relationship between the various AC data models:

- "ietf-ac-common" [RFC9833]
- "ietf-bearer-svc" (Section 6.1)
- "ietf-ac-svc" (Section 6.2)
- "ietf-ac-ntw" [RFC9835]
- "ietf-ac-glue" [RFC9836]

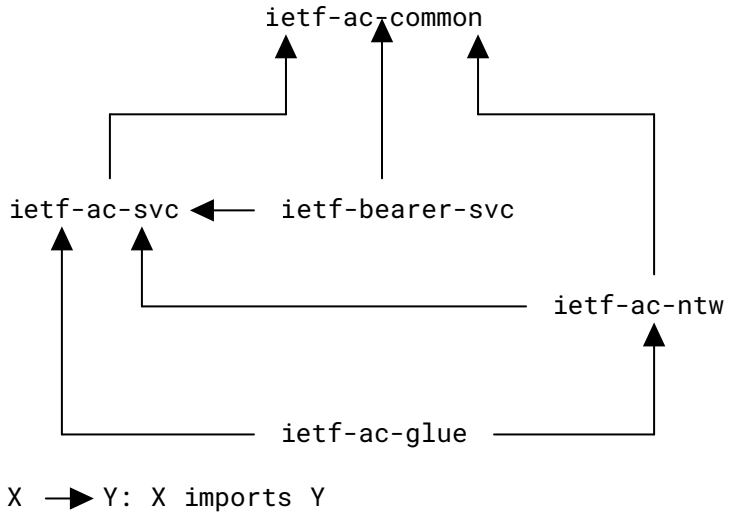


Figure 1: AC Data Models

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 L2VPN or L3VPN 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 Data Models

4.1. ACs Terminated by One or Multiple CEs

Figure 2 depicts two target topology flavors that involve ACs. These topologies have the following characteristics:

- A CE can be either a physical device or a logical entity. Such logical entity is typically a software component (e.g., a virtual SF that is hosted within the provider's network or a third-party infrastructure). A CE is seen by the network as a peer SAP.
- An AC service request may include one or multiple ACs, which may be associated to a single CE or multiple CEs.
- CEs may be either dedicated to one single connectivity service or host multiple connectivity services (e.g., CEs with roles of SFs [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 AC. 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.
- 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 provider network (i.e., select which PE(s) to use) and also whether to enable specific capabilities (e.g., Virtual Router Redundancy Protocol (VRRP) [RFC9568]). Note that placement constraints may also be requested during the instantiation of the underlying bearers (Section 5.1).

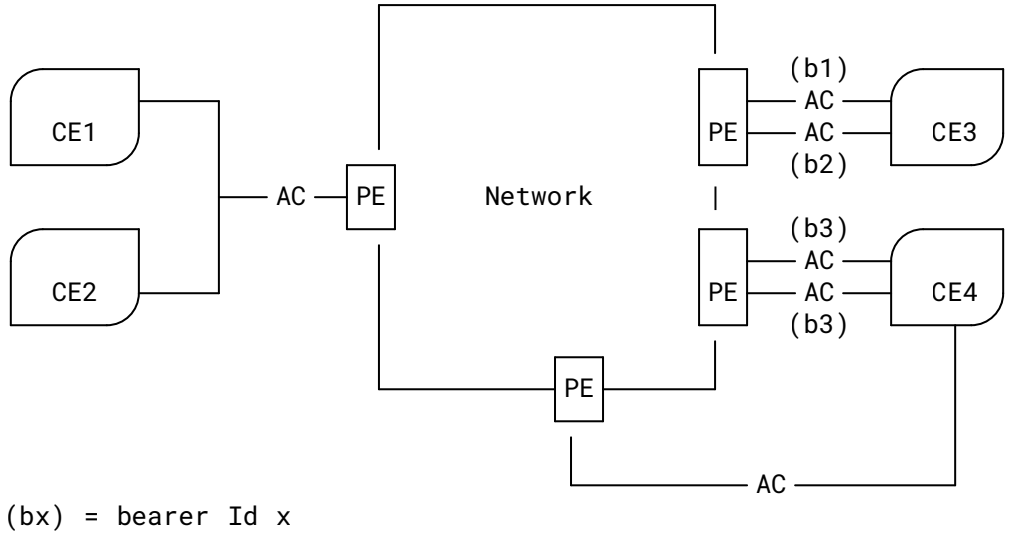


Figure 2: Examples of ACs

4.2. Separate AC Provisioning vs. Actual Service Provisioning

The procedure to provision a service in a service provider network may depend on the practices adopted by a service provider. This includes the workflow put in place for the provisioning of network services and how they are bound to an AC. For example, a single AC may be used to host multiple connectivity services. In order to avoid service interference and redundant information in various locations, a service provider may expose an interface to manage ACs network-wide. Customers can then request a bearer or an AC to be put in place and then refer to that bearer or AC when requesting services that are bound to the bearer or AC. [RFC9836] specifies augmentations to the L2SM and the L3SM to bind LxVPN services to ACs.

4.3. Sample Deployment Models

Figure 3 illustrates an example of how the bearer/AC service models can be used between a customer and a provider. Internals to the provider orchestration domain (or customer orchestration domain) are hidden to the customer (or provider).

Resources that are needed to activate an AC (e.g., Layer 2 or Layer 3 identifiers) are typically imposed by the provider. However, the deployment model assumes that the customer may supply a specific identifier (e.g., selected from a pool that was pre-provisioned by the provider) in a service request. The provider may accept or reject such request.

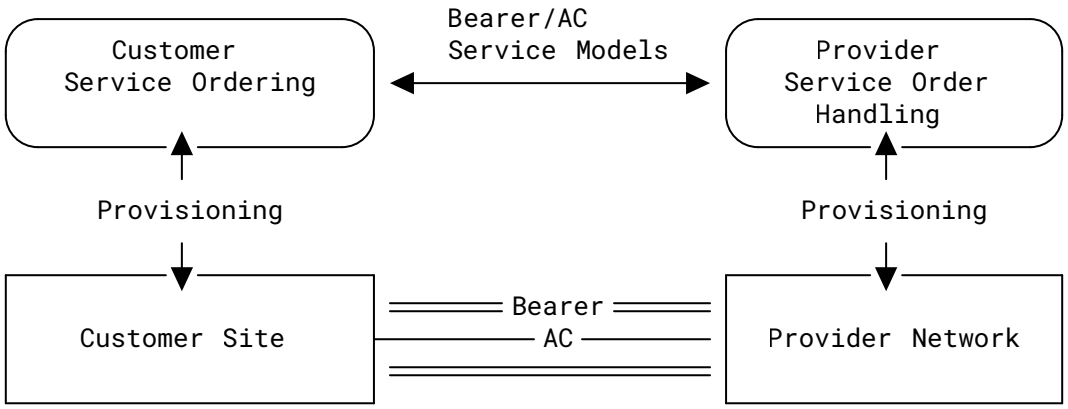
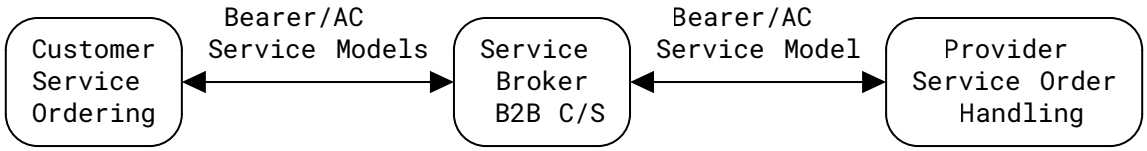


Figure 3: Example of Interaction Between Customer and Provider Orchestrations

Figure 4 illustrates an example of how the bearer/AC service models involve a third party. This deployment model follows a recursive approach, but other client/server alternative modes with a third party can be considered. In a recursive deployment, the Service Broker exposes a server to a customer for the ordering of AC services, but it also acts as a client when communicating with a provider. How the Service Broker decides to terminate a recursion for a given service request or create child service requests is specific to each deployment.



B2B C/S: Back-to-Back Client/Server

Figure 4: Example of Recursive Deployment

Figure 5 shows the positioning of the AC service model in the overall service delivery process, with a focus on the provider.

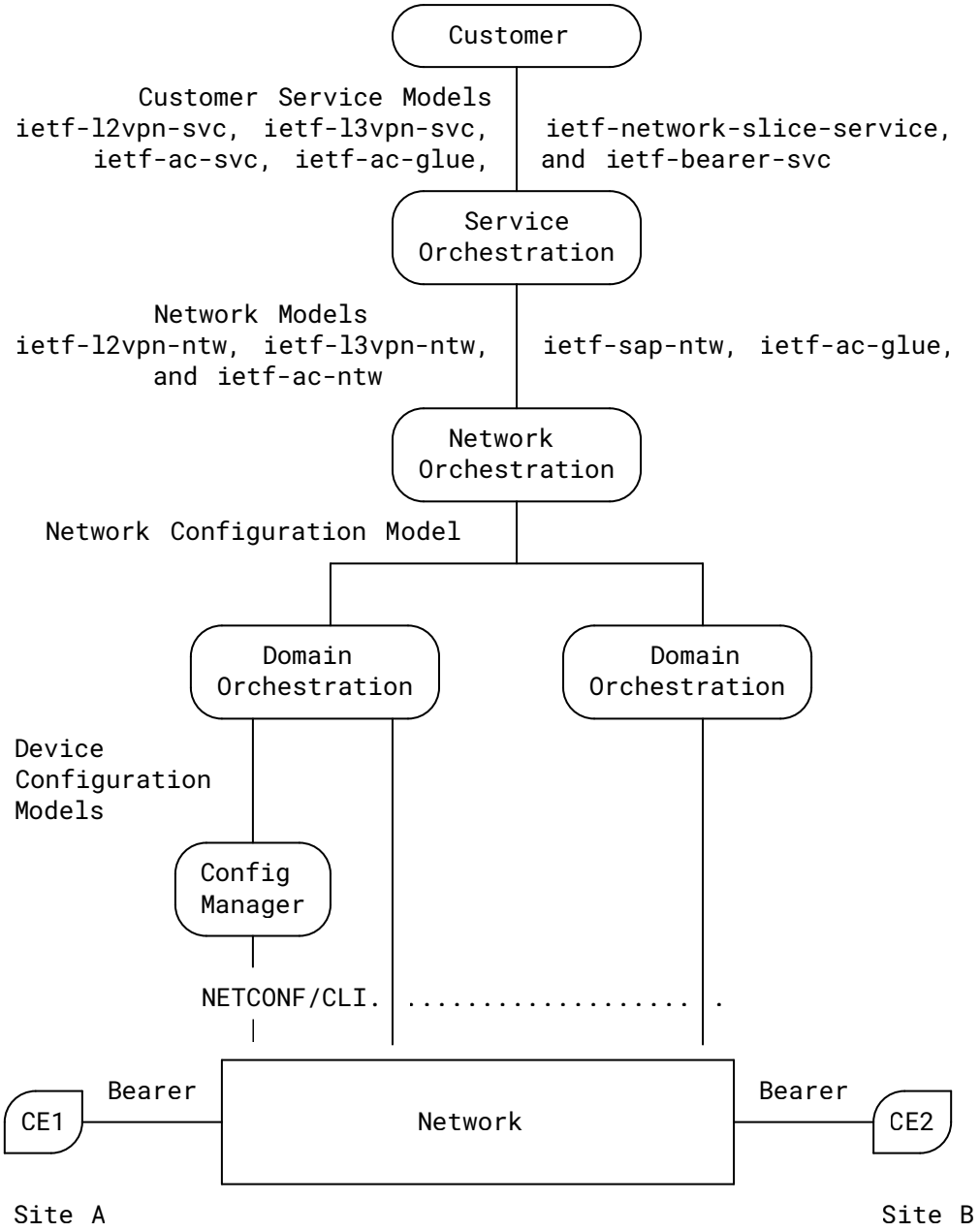


Figure 5: An Example of AC Model Usage (Focus on the Provider's Internals)

In order to ease the mapping between the service model and underlying network models (e.g., the L3VPN Network Model (L3NM) and SAP), the name conventions used in existing network data models are reused as much as possible. For example, 'local-address' is used rather than 'provider-address' (or similar) to refer to an IP address used in the provider network. This approach is consistent with the automation framework defined in [RFC8969].

5. Description of the Data Models

5.1. The Bearer Service ("ietf-bearer-svc") YANG Module

Figure 6 shows the tree for managing the bearers (that is, the properties of an attachment that are below Layer 3). A bearer can be a physical or logical link (e.g., LAG [IEEE802.1AX]). Also, a bearer can be a wireless or wired link. A reference to a bearer is generated by the operator. Such a reference can be used, e.g., in a subsequent service request to create an AC. The anchoring of the AC can also be achieved by indicating (with or without a bearer reference) a peer SAP identifier (e.g., an identifier of an SF).

```
module: ietf-bearer-svc
  +--rw locations
  |  +--rw customer* [name peer-as]
  |     +--rw name        string
  |     +--rw peer-as     inet:as-number
  |     +--ro location* [name]
  |        +--ro name            string
  |        +--ro address?        string
  |        +--ro city?           string
  |        +--ro postal-code?    string
  |        +--ro state?          string
  |        +--ro country-code?   string
  +--rw bearers
     +--rw requested-start?         yang:date-and-time
     +--rw requested-stop?          yang:date-and-time
     +--ro actual-start?            yang:date-and-time
     +--ro actual-stop?             yang:date-and-time
     +--rw placement-constraints
     |  +--rw constraint* [constraint-type]
     |          {vpn-common:placement-diversity}?
     |     +--rw constraint-type    identityref
     |     +--rw target
     |        +--rw (target-flavor)?
     |           +--:(id)
     |           |  +--rw group* [group-id]
     |           |     +--rw group-id    string
     |           +--:(all-bearers)
     |           |  +--rw all-other-bearers?   empty
     |           +--:(all-groups)
     |              +--rw all-other-groups?    empty
     +--rw bearer* [name]
        +--rw name                          string
        +--rw description?                  string
        +--rw customer-name?                string
        +--rw groups
        |  +--rw group* [group-id]
        |     +--rw group-id    string
        +--rw op-comment?                   string
        +--rw bearer-parent-ref?            bearer-svc:bearer-ref
        +--ro bearer-lag-member*            bearer-svc:bearer-ref
        +--ro sync-phy-capable?             boolean
        +--rw sync-phy-enabled?             boolean
        +--rw sync-phy-type?                identityref
        +--rw provider-location-reference?  string
        +--rw customer-point
        |  +--rw identified-by?   identityref
        |  +--rw device
        |  |  +--rw device-id?   string
        |  |  +--rw location
        |  |     +--rw name?           string
        |  |     +--rw address?        string
        |  |     +--rw city?           string
        |  |     +--rw postal-code?    string
        |  |     +--rw state?          string
        |  |     +--rw country-code?   string
        |  +--rw site
        |  |  +--rw site-id?    string
```

```
         |  |  +--rw location
         |  |     +--rw name?           string
         |  |     +--rw address?        string
         |  |     +--rw city?           string
         |  |     +--rw postal-code?    string
         |  |     +--rw state?          string
         |  |     +--rw country-code?   string
         |  +--rw custom-id?       string
         +--rw type?                           identityref
         +--rw test-only?                      empty
         +--ro bearer-reference?               string
         |       {ac-common:server-assigned-reference}?
         +--ro ac-svc-ref*
         |       ac-svc:attachment-circuit-reference
         +--rw requested-start?                yang:date-and-time
         +--rw requested-stop?                 yang:date-and-time
         +--ro actual-start?                   yang:date-and-time
         +--ro actual-stop?                    yang:date-and-time
         +--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
```

Figure 6: Bearer Service Tree Structure

In some deployments, a customer may first retrieve a list of available presence locations before placing an order for a bearer creation. The request is filtered based upon a customer name and an Autonomous System Number (ASN). The reserved value "AS 0" [RFC7607] is used for customers with no ASN. The retrieved location names may then be referenced in a bearer creation request ('provider-location-reference'). See the example provided in Appendix A.10.1.

The same customer site (CE, SF, etc.) can terminate one or multiple bearers; each of them is uniquely identified by a reference that is assigned by the network provider. These bearers can terminate on the same or distinct network nodes. CEs that terminate multiple bearers are called multi-homed CEs.

A bearer can be created, modified, or discovered from the network. For example, the following deployment options can be considered:

Greenfield creation: In this scenario, bearers are created from scratch using specific requests made to a network controller. This method allows providers to tailor bearer creation to meet customer-specific needs. For example, a bearer request may indicate some hints about the placement constraints ('placement-constraints'). These constraints are used by a provider to determine how/where to terminate a bearer in the network side (e.g., Point of Presence (PoP) or PE selection).

Auto-discovery using network protocols: Devices can use specific protocols (e.g., Link Layer Discovery Protocol (LLDP) [IEEE802.1AB]) to automatically discover and connect to available network resources. A network controller can use such reported information to expose discovered bearers from the network using the same bearer data model structure.

A request to create a bearer may include a set of constraints ('placement-constraints') that are used by a controller to decide the network terminating side of a bearer (e.g., PE selection, PE redundancy, or PoP selection). Future placement criteria ('constraint-type') may be defined to accommodate specific deployment contexts. A request may also include some timing constraints ('requested-start', 'requested-stop') that are applicable for a set of bearers. The timing constraints can be adjusted at the 'bearer' level. These adjusted values take precedence over the global values.

The descriptions of the bearer data nodes are as follows:

'name': Used to uniquely identify a bearer. This name is typically selected by the client when requesting a bearer.

'customer-name': Indicates the name of the customer who ordered the bearer.

'description': Includes a textual description of the bearer.

'group': Tags a bearer with one or more identifiers that are used to group a set of bearers.

'op-comment': Includes operational comments that may be useful for managing the bearer (building, level, etc.). No structure is associated with this data node to accommodate all deployments.

'bearer-parent-ref': Specifies the parent bearer. This data node can be used, e.g., if a bearer is a member of a LAG.

'bearer-lag-member': Lists the bearers that are members of a LAG. Members can be declared as part of a LAG using 'bearer-parent-ref'.

'sync-phy-capable': Reports whether a synchronization physical (Sync PHY) mechanism is supported for this bearer.

'sync-phy-enabled': Indicates whether a Sync PHY mechanism is enabled for a bearer. It only applies when 'sync-phy-capable' is set to 'true'.

'sync-phy-type': Specifies the Sync PHY mechanism (e.g., SyncE [ITU-T-G.781]) enabled for the bearer.

'provider-location-reference': Indicates a location identified by a provider-assigned reference.

'customer-point': Specifies the customer termination point for the bearer. A bearer request can indicate a device, a site, a combination thereof, or custom information. All these schemes are supported in the model.

'type': Specifies the bearer type (Ethernet, wireless, LAG, etc.).

'test-only': Indicates that a request is only for test validation and not for enforcement, even if there are no errors. This is used for feasibility checks. This data node is applicable only when the data model is used with protocols that do not have built-in support of such option. For example, this data node is redundant with the "test-only" value of the `<test-option>` parameter in the NETCONF `<edit-config>` operation (Section 7.2 of [RFC6241]).

'bearer-reference': Returns an internal reference for the service provider to uniquely identify the bearer. This reference can be used when requesting services. Appendix A.1 provides an example about how this reference can be retrieved by a customer.

Whether the 'bearer-reference' mirrors the content of the 'name' is deployment-specific. The module does not assume nor preclude such schemes.

'ac-svc-ref': Specifies the set of ACs that are bound to the bearer.

'requested-start': Specifies the requested date and time when the bearer is expected to be active.

'requested-stop': Specifies the requested date and time when the bearer is expected to be disabled.

'actual-start': Reports the actual date and time when the bearer was enabled.

'actual-stop': Reports the actual date and time when the bearer was disabled.

'status': Used to track the overall status of a given bearer. Both the operational and administrative status are maintained together with a timestamp.

The 'admin-status' attribute is typically configured by a network operator to indicate whether the service is enabled, disabled, or subjected to additional testing or pre-deployment checks. These additional options, such as 'admin-testing' and 'admin-pre-deployment', provide the operators the flexibility to conduct additional validations on the bearer before deploying services over that connection.

'oper-status': Reflects the operational state of a bearer as observed. As a bearer can contain multiple services, the operational status should only reflect the status of the bearer connection. To obtain network-level service status, specific network models, such as those in Section 7.3 of [RFC9182] or Section 7.3 of [RFC9291], should be consulted.

It is important to note that the 'admin-status' attribute should remain independent of the 'oper-status'. In other words, the setting of the intended administrative state (e.g., 'admin-up' or 'admin-testing') **MUST NOT** be influenced by the current operational state. If the bearer is administratively set to 'admin-down', it is expected that the bearer will also be operationally 'op-down' as a result of this administrative decision.

To assess the service delivery status for a given bearer comprehensively, it is recommended to consider both administrative and operational service status values in conjunction. This holistic approach allows a network controller or operator to identify anomalies effectively.

For instance, when a bearer is administratively enabled but the 'operational-status' of that bearer is reported as 'op-down', it should be expected that the 'oper-status' of services transported over that bearer is also down. These status values differing should trigger the detection of an anomaly condition.

See "A Common YANG Data Model for Layer 2 and Layer 3 VPNs" [RFC9181] for more details.

5.2. The Attachment Circuit Service ("ietf-ac-svc") YANG Module

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

5.2.1. Overall Structure

The overall tree structure of the AC service module is shown in Figure 7.

```
+--rw specific-provisioning-profiles
|  ...
+--rw service-provisioning-profiles
|  ...
+--rw attachment-circuits
   +--rw ac-group-profile* [name]
   |  ...
   +--rw placement-constraints
   |  ...
   +--rw ac* [name]
      ...
      +--rw l2-connection  {ac-common:layer2-ac}?
      |  ...
      +--rw ip-connection  {ac-common:layer3-ac}?
      |  ...
      +--rw routing-protocols
      |  ...
      +--rw oam
      |  ...
      +--rw security
      |  ...
      +--rw service
         ...
```

Figure 7: Overall AC Service Tree Structure

The rationale for deciding whether a reusable grouping is included in this document or moved into the AC common module [RFC9833] is as follows:

- Groupings that are reusable among the AC service module, AC network module, and other service models and network models are included in the AC common module.

- Groupings that are reusable only by other service models are maintained in the "ietf-ac-svc" module.

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

The AC service model uses groupings and types defined in the AC common model [RFC9833] ('op-instructions', 'dot1q', 'qinq', 'priority-tagged', 'l2-tunnel-service', etc.). Therefore, the descriptions of these nodes are not reiterated in the following subsections.

Features are used to tag conditional portions of the model in order to accommodate various deployments (support of layer 2 ACs, Layer 3 ACs, IPv4, IPv6, routing protocols, Bidirectional Forwarding Detection (BFD), etc.).

5.2.2. Service Profiles

5.2.2.1. Description

The 'specific-provisioning-profiles' container (Figure 8) can be used by a service provider to maintain a set of reusable profiles. The profiles definitions are similar to those defined in [RFC9181], including: Quality of Service (QoS), BFD, forwarding, and routing profiles. The exact definition of the profiles is local to each service provider. The model only includes an identifier for these profiles in order to facilitate identifying and binding local policies when building an AC.

```
module: ietf-ac-svc
  +--rw specific-provisioning-profiles
  |  +--rw valid-provider-identifiers
  |     +--rw 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 service-provisioning-profiles
  |  +--rw service-profile-identifier* [id]
  |     +--rw id    string
  +--rw attachment-circuits
     +--rw ac-group-profile* [name]
     |  ...
     +--rw placement-constraints
     |  ...
     +--rw ac* [name]
        ...
        +--rw l2-connection  {ac-common:layer2-ac}?
        |  ...
        +--rw ip-connection  {ac-common:layer3-ac}?
        |  ...
        +--rw routing-protocols
        |  ...
        +--rw oam
        |  ...
        +--rw security
        |  ...
        +--rw service
           ...
```

Figure 8: Service Profiles

As shown in Figure 8, two profile types can be defined: 'specific-provisioning-profiles' and 'service-provisioning-profiles'. Whether only specific profiles, service profiles, or a combination thereof are used is local to each service provider.

The following specific provisioning profiles can be defined as follows:

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

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

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

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

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

5.2.2.2. Referencing Service/Specific Profiles

All the above mentioned profiles are uniquely identified by the provider server. To ease referencing these profiles by other data models, specific typedefs are defined for each of these profiles. Likewise, an AC reference typedef is defined when referencing a (global) AC by its name is required. These typedefs **SHOULD** be used when other modules need a reference to one of these profiles or ACs.

5.2.3. Attachment Circuits Profiles

The 'ac-group-profile' defines reusable parameters for a set of ACs. Each profile is identified by 'name'. Some of the data nodes can be adjusted at the 'ac' level. These adjusted values take precedence over the global values. The structure of 'ac-group-profile' is similar to the one used to model each 'ac' (Figure 10).

5.2.4. AC Placement Constraints

The 'placement-constraints' specifies the placement constraints of an AC. For example, this container can be used to request avoidance of connecting two ACs to the same PE. The full set of supported constraints is defined in [RFC9181] (see 'placement-diversity', in particular).

The structure of 'placement-constraints' is shown in Figure 9.

```
+--rw specific-provisioning-profiles
|  ...
+--rw service-provisioning-profiles
|  ...
+--rw attachment-circuits
   +--rw ac-group-profile* [name]
   |  ...
   +--rw placement-constraints
   |  +--rw constraint* [constraint-type]
   |     +--rw constraint-type    identityref
   |     +--rw target
   |        +--rw (target-flavor)?
   |           +--:(id)
   |           |  +--rw group* [group-id]
   |           |     +--rw group-id    string
   |           +--:(all-accesses)
   |           |  +--rw all-other-accesses?   empty
   |           +--:(all-groups)
   |              +--rw all-other-groups?     empty
   +--rw ac* [name]
      ...
```

Figure 9: Placement Constraints Subtree Structure

5.2.5. Attachment Circuits

The structure of 'attachment-circuits' is shown in Figure 10.

```
+--rw specific-provisioning-profiles
|  ...
+--rw service-provisioning-profiles
|  ...
+--rw attachment-circuits
   +--rw ac-group-profile* [name]
   |  ...
   +--rw placement-constraints
   |  ...
   +--rw customer-name?           string
   +--rw requested-start?         yang:date-and-time
   +--rw requested-stop?          yang:date-and-time
   +--ro actual-start?            yang:date-and-time
   +--ro actual-stop?             yang:date-and-time
   +--rw ac* [name]
      +--rw customer-name?       string
      +--rw description?         string
      +--rw test-only?           empty
      +--rw requested-start?     yang:date-and-time
      +--rw requested-stop?      yang:date-and-time
      +--ro actual-start?        yang:date-and-time
      +--ro actual-stop?         yang:date-and-time
      +--rw role?                identityref
      +--rw peer-sap-id*         string
      +--rw group-profile-ref*    ac-group-reference
      +--rw parent-ref*       ac-svc:attachment-circuit-reference
      +--ro child-ref*        ac-svc:attachment-circuit-reference
      +--rw group* [group-id]
      |  +--rw group-id      string
      |  +--rw precedence?   identityref
      +--ro service-ref* [service-type service-id]
      |  +--ro service-type    identityref
      |  +--ro service-id      string
      +--ro server-reference?    string
      |       {ac-common:server-assigned-reference}?
      +--rw name                 string
      +--rw service-profile*     service-profile-reference
      +--rw l2-connection  {ac-common:layer2-ac}?
      |  ...
      +--rw ip-connection  {ac-common:layer3-ac}?
      |  ...
      +--rw routing-protocols
      |  ...
      +--rw oam
      |  ...
      +--rw security
      |  ...
      +--rw service
         ...
```

Figure 10: Attachment Circuits Tree Structure

A request may also include some timing constraints ('requested-start', 'requested-stop') that are applicable for a set of ACs. The timing constraints can be adjusted at the 'ac' level. These adjusted values take precedence over the global values.

The 'ac' data nodes are described as follows:

'customer-name': Indicates the name of the customer who ordered the AC or a set of ACs.

'description': Includes a textual description of the AC.

'test-only': Indicates that a request is only for a validation test and not for enforcement, even if there are no errors. This is used for feasibility checks. This data node is applicable only when the data model is used with protocols that do not have built-in support of such option.

'requested-start': Specifies the requested date and time when the AC is expected to be active.

'requested-stop': Specifies the requested date and time when the AC is expected to be disabled.

'actual-start': Reports the actual date and time when the AC was enabled.

'actual-stop': Reports the actual date and time when the AC was disabled.

'role': Specifies whether an AC is used, e.g., as User-to-Network Interface (UNI) or Network-to-Network Interface (NNI).

'peer-sap-id': Includes references to the remote endpoints of an AC [RFC9408]. 'peer' is drawn here from the perspective of the provider network. That is, a 'peer-sap' will refer to a customer node.

'group-profile-ref': Indicates references to one or more profiles that are defined in Section 5.2.3.

'parent-ref': Specifies an AC that is inherited by an AC.

In contexts where dynamic termination points are managed for a given AC, a Parent AC can be defined with a set of stable and common information, while Child ACs are defined to track dynamic information. These Child ACs are bound to the Parent AC, which is exposed to services (as a stable reference).

Whenever a Parent AC is deleted, all its Child ACs **MUST** be deleted.

A Child AC **MAY** rely upon more than one Parent AC (e.g., parent Layer 2 AC and parent Layer 3 AC). In such cases, these ACs **MUST NOT** be overlapping. An example to illustrate the use of multiple Parent ACs is provided in Appendix A.12.

'child-ref': Lists one or more references of Child ACs that rely upon this AC as a Parent AC.

'group': Lists the groups to which an AC belongs [RFC9181]. For example, the 'group-id' is used to associate redundancy or protection constraints of ACs. An example is provided in Appendix A.5.

'service-ref': Reports the set of services that are bound to the AC. The services are indexed by their type.

'server-reference': Reports the internal reference that is assigned by the provider for this AC. This reference is used to accommodate deployment contexts (e.g., Section 9.1.2 of [RFC8921]) where an identifier is generated by the provider to identify a service order locally.

'name': Associates a name that uniquely identifies an AC within a service provider network.

'service-profile': References a set of service-specific profiles.

'l2-connection': See Section 5.2.5.1.

'ip-connection': See Section 5.2.5.2.

'routing': See Section 5.2.5.3.

'oam': See Section 5.2.5.4.

'security': See Section 5.2.5.5.

'service': See Section 5.2.5.6.

5.2.5.1. Layer 2 Connection Structure

The 'l2-connection' container (Figure 11) is used to configure the relevant Layer 2 properties of an AC, including encapsulation details and tunnel terminations. For the encapsulation details, the model supports the definition of the type as well as the identifiers (e.g., VLAN-IDs) of each of the encapsulation-type defined. For the second case, attributes for pseudowire, Virtual Private LAN Service (VPLS), and Virtual eXtensible Local Area Network (VXLAN) tunnel terminations are included.

'bearer-reference' is used to link an AC with a bearer over which the AC is instantiated.

This structure relies upon the common groupings defined in [RFC9833].

```
+--rw specific-provisioning-profiles
|  ...
+--rw service-provisioning-profiles
|  ...
+--rw attachment-circuits
   +--rw ac-group-profile* [name]
   |  ...
   +--rw placement-constraints
   |  ...
   +--rw ac* [name]
      ...
      +--rw name                  string
      +--rw l2-connection  {ac-common:layer2-ac}?
      |  +--rw encapsulation
      |  |  +--rw type?              identityref
      |  |  +--rw dot1q
      |  |  |  +--rw tag-type?   identityref
      |  |  |  +--rw cvlan-id?   uint16
      |  |  +--rw priority-tagged
      |  |  |  +--rw tag-type?   identityref
      |  |  +--rw qinq
      |  |     +--rw tag-type?   identityref
      |  |     +--rw svlan-id?   uint16
      |  |     +--rw cvlan-id?   uint16
      |  +--rw (l2-service)?
      |  |  +--:(l2-tunnel-service)
      |  |  |  +--rw l2-tunnel-service
      |  |  |     +--rw type?         identityref
      |  |  |     +--rw pseudowire
      |  |  |     |  +--rw vcid?      uint32
      |  |  |     |  +--rw far-end?   union
      |  |  |     +--rw vpls
      |  |  |     |  +--rw vcid?      uint32
      |  |  |     |  +--rw far-end*   union
      |  |  |     +--rw vxlan
      |  |  |        +--rw vni-id?            uint32
      |  |  |        +--rw peer-mode?         identityref
      |  |  |        +--rw peer-ip-address*   inet:ip-address
      |  |  +--:(l2vpn)
      |  |     +--rw l2vpn-id?            vpn-common:vpn-id
      |  +--rw bearer-reference?          string
      |          {vpn-common:bearer-reference}?
      +--rw ip-connection  {ac-common:layer3-ac}?
      |  ...
      +--rw routing-protocols
      |  ...
      +--rw oam
      |  ...
      +--rw security
      |  ...
      +--rw service
         ...
```

Figure 11: Layer 2 Connection Tree Structure

5.2.5.2. IP Connection Structure

The 'ip-connection' container is used to configure the relevant IP properties of an AC. The model supports the usage of dynamic and static addressing. This structure relies upon the common groupings defined in Section 4.3 of [RFC9833]. Both IPv4 and IPv6 parameters are supported.

For ACs that require Layer 3 tunnel establishment, the ACaaS includes a provision for future augmentations to define tunnel-specific data nodes ('l3-tunnel-service'). Such augmentations **MUST** be conditional based on the tunnel type ('type').

Figure 12 shows the structure of the IPv4 connection.

```
        | ...
        +--rw ip-connection {ac-common:layer3-ac}?
        |  +--rw ipv4 {vpn-common:ipv4}?
        |  |  +--rw local-address?
        |  |  |       inet:ipv4-address
        |  |  +--rw virtual-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
        |  |     |  +--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?
        |  |                   failure-detection-profile-reference
        |  |                   {vpn-common:bfd}?
        |  +--rw ipv6 {vpn-common:ipv6}?
        |  |  ...
        |  +--rw (l3-service)?
        |     +--:(l3-tunnel-service)
        |        +--rw l3-tunnel-service
        |           +--rw type?   identityref
```

Figure 12: Layer 3 Connection Tree Structure (IPv4)

Figure 13 shows the structure of the IPv6 connection.

```
        | ...
        +--rw ip-connection  {ac-common:layer3-ac}?
        |  +--rw ipv4 {vpn-common:ipv4}?
        |  |  ...
        |  +--rw ipv6 {vpn-common:ipv6}?
        |  |  +--rw local-address?
        |  |  |       inet:ipv6-address
        |  |  +--rw virtual-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
        |  |     |  +--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?
        |  |                   failure-detection-profile-reference
        |  |                   {vpn-common:bfd}?
        |  +--rw (l3-service)?
        |     +--:(l3-tunnel-service)
        |        +--rw l3-tunnel-service
        |           +--rw type?   identityref
        | ...
```

Figure 13: Layer 3 Connection Tree Structure (IPv6)

5.2.5.3. Routing

As shown in the tree depicted in Figure 14, the 'routing-protocols' container defines the required parameters to enable the desired routing features for an AC. One or more routing protocols can be associated with an AC. Such routing protocols will then be enabled between a PE and the

customer termination points. Each routing instance is uniquely identified by the combination of the 'id' and 'type' to accommodate scenarios where multiple instances of the same routing protocol have to be configured on the same link.

In addition to static routing (Section 5.2.5.3.1), the module supports BGP (Section 5.2.5.3.2), OSPF (Section 5.2.5.3.3), IS-IS (Section 5.2.5.3.4), and RIP (Section 5.2.5.3.5). It also includes a reference to the 'routing-profile-identifier' defined in Section 5.2.2, so that additional constraints can be applied to a specific instance of each routing protocol. Moreover, the module supports VRRP (Section 5.2.5.3.6).

```
+--rw specific-provisioning-profiles
|  ...
+--rw service-provisioning-profiles
|  ...
+--rw attachment-circuits
   +--rw ac-group-profile* [name]
   |  ...
   +--rw placement-constraints
   |  ...
   +--rw ac* [name]
      ...
      +--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-profiles* [id]
      |     |  +--rw id      routing-profile-reference
      |     |  +--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 14: Routing Tree Structure

5.2.5.3.1. Static Routing

The static tree structure is shown in Figure 15.

```
        |  ...
        +--rw routing-protocols
        |  +--rw routing-protocol* [id]
        |     +--rw id                  string
        |     +--rw type?               identityref
        |     +--rw routing-profiles* [id]
        |     |  +--rw id      routing-profile-reference
        |     |  +--rw type?   identityref
        |     +--rw static
        |     |  +--rw cascaded-lan-prefixes
        |     |     +--rw ipv4-lan-prefix* [lan next-hop]
        |     |     |       {vpn-common:ipv4}?
        |     |     |  +--rw lan
        |     |     |  |       inet:ipv4-prefix
        |     |     |  +--rw lan-tag?                     string
        |     |     |  +--rw next-hop                     union
        |     |     |  +--rw metric?                      uint32
        |     |     |  +--rw failure-detection-profile?
        |     |     |  |       failure-detection-profile-reference
        |     |     |  |       {vpn-common:bfd}?
        |     |     |  +--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 failure-detection-profile?
        |     |        |       failure-detection-profile-reference
        |     |        |       {vpn-common:bfd}?
        |     |        +--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}?
        |        ...
```

Figure 15: Static Routing Tree Structure

As depicted in Figure 15, 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.

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

'failure-detection-profile': Indicates a failure detection profile (e.g., BFD) that applies for this entry.

'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.

5.2.5.3.2. BGP

An AC service activation with BGP routing [RFC4271] **SHOULD** include at least the customer's AS Number (ASN) and the provider's ASN. Additional information can be supplied by a customer in a request or exposed by a provider in a response to a query request in order to ease the process of automating the provisioning of BGP sessions (the customer does not use the primary IP address to establish the underlying BGP session, communicate the provider's IP address used to establish the BGP session, share authentication parameters, bind the session to a forwarding protection profile, etc.).

The BGP tree structure is shown in Figure 16.

```
        |  ...
        +--rw routing-protocols
        |  +--rw routing-protocol* [id]
        |     +--rw id                  string
        |     +--rw type?               identityref
        |     +--rw routing-profiles* [id]
        |     |  +--rw id      routing-profile-reference
        |     |  +--rw type?   identityref
        |     +--rw static
        |     |  ...
        |     +--rw bgp {vpn-common:rtg-bgp}?
        |     |  +--rw peer-groups
        |     |  |  +--rw peer-group* [name]
        |     |  |     +--rw name              string
        |     |  |     +--rw local-as?         inet:as-number
        |     |  |     +--rw peer-as?          inet:as-number
        |     |  |     +--rw address-family?   identityref
        |     |  |     +--rw role?             identityref
        |     |  |     +--rw local-address?    inet:ip-address
        |     |  |     +--rw bgp-max-prefix
        |     |  |     |  +--rw max-prefix?   uint32
        |     |  |     +--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* [id]
        |     |     +--rw id                  string
        |     |     +--ro server-reference?   string
        |     |     |       {ac-common:server-assigned-reference}?
        |     |     +--rw remote-address?     inet:ip-address
        |     |     +--rw local-address?      inet:ip-address
        |     |     +--rw local-as?           inet:as-number
        |     |     +--rw peer-as?            inet:as-number
        |     |     +--rw address-family?     identityref
        |     |     +--rw role?               identityref
        |     |     +--rw bgp-max-prefix
        |     |     |  +--rw max-prefix?   uint32
        |     |     +--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 requested-start?    yang:date-and-time
          |     |     +--rw requested-stop?     yang:date-and-time
          |     |     +--ro actual-start?       yang:date-and-time
          |     |     +--ro actual-stop?        yang:date-and-time
          |     |     +--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 peer-group?
          |     |     |       -> ../../peer-groups/peer-group/name
          |     |     +--rw failure-detection-profile?
          |     |                   failure-detection-profile-reference
          |     |                   {vpn-common:bfd}?
          |     +--rw ospf {vpn-common:rtg-ospf}?
          |     |  ...
          |     +--rw isis {vpn-common:rtg-isis}?
          |     |  ...
          |     +--rw rip {vpn-common:rtg-rip}?
          |     |  ...
          |     +--rw vrrp {vpn-common:rtg-vrrp}?
          |        ...
```

Figure 16: BGP Tree Structure

For deployment cases where an AC service request includes a list of neighbors with redundant information, the ACaaS allows factorizing shared data by means of 'peer-group'. Thus, the presence of 'peer-groups' in a service request is optional.

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

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

'local-as': Reports the provider's ASN. This information is used at the customer side to configure the BGP session with the provider network.

'peer-as': Indicates the customer's ASN. This information is used at the provider side to configure the BGP session with the customer equipment.

'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]. This parameter is useful for interconnection scenarios.

This is an optional parameter.

'local-address': Reports a provider's IP address to use to establish the BGP transport session.

'bgp-max-prefix': Indicates the maximum number of BGP prefixes allowed in a session for this group.

'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.

Similar to [RFC9182], this version of the ACaaS assumes that parameters specific to the TCP-AO are preconfigured as part of the key chain that is referenced in the ACaaS. 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 "YANG Data Model for Key Chains" [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:

'server-reference': Reports the internal reference that is assigned by the provider for this BGP session. This is an optional parameter.

'remote-address': Specifies the customer's IP address used to establish this BGP session. If not present, this means that the primary customer IP address is used as the remote IP address.

'requested-start': Specifies the requested date and time when the BGP session is expected to be active.

'requested-stop': Specifies the requested date and time when the BGP session is expected to be disabled.

'actual-start': Reports the actual date and time when the BGP session was enabled.

'actual-stop': Reports the actual date and time when the BGP session was disabled.

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

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

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

This is an optional parameter.

'failure-detection-profile': Indicates a failure detection profile (BFD) that applies for a BGP neighbor. This is an optional parameter.

5.2.5.3.3. OSPF

The OSPF tree structure is shown in Figure 17.

```
|  ...
+--rw routing-protocols
|  +--rw routing-protocol* [id]
|     +--rw id                    string
|     +--rw type?                 identityref
|     +--rw routing-profiles* [id]
|     |  +--rw id      routing-profile-reference
|     |  +--rw type?   identityref
|     +--rw static
|     |  ...
|     +--rw 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 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}?
|        ...
```

Figure 17: OSPF Tree Structure

The following OSPF data nodes are supported:

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

'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]).

'authentication': Controls the authentication schemes to be enabled for the OSPF instance. The model supports authentication options that are common to both OSPF versions: the Authentication Trailer for OSPFv2 [RFC5709][RFC7474] and OSPFv3 [RFC7166].

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

5.2.5.3.4. IS-IS

The IS-IS tree structure is shown in Figure 18.

```
|  ...
+--rw routing-protocols
|  +--rw routing-protocol* [id]
|     +--rw id                  string
|     +--rw type?               identityref
|     +--rw routing-profiles* [id]
|     |  +--rw id      routing-profile-reference
|     |  +--rw type?   identityref
|     +--rw static
|     |  ...
|     +--rw 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 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}?
|      ...
```

Figure 18: IS-IS 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.

'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.

5.2.5.3.5. RIP

The RIP tree structure is shown in Figure 19.

```
        |  ...
        +--rw routing-protocols
        |  +--rw routing-protocol* [id]
        |     +--rw id                  string
        |     +--rw type?               identityref
        |     +--rw routing-profiles* [id]
        |     |  +--rw id      routing-profile-reference
        |     |  +--rw type?   identityref
        |     +--rw static
        |     |  ...
        |     +--rw 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 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}?
        |      ...
```

Figure 19: RIP Tree Structure

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

5.2.5.3.6. VRRP

The model supports the Virtual Router Redundancy Protocol (VRRP) [RFC9568] on an AC (Figure 20).

```
|  ...
+--rw routing-protocols
|  +--rw routing-protocol* [id]
|     +--rw id                  string
|     +--rw type?               identityref
|     +--rw routing-profiles* [id]
|     |  +--rw id      routing-profile-reference
|     |  +--rw type?   identityref
|     +--rw static
|     |  ...
|     +--rw 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 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
```

Figure 20: VRRP Tree Structure

The following 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.

'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]).

5.2.5.4. Operations, Administration, and Maintenance (OAM)

As shown in the tree depicted in Figure 21, the 'oam' container defines OAM-related parameters of an AC.

```
+--rw specific-provisioning-profiles
|  ...
+--rw service-provisioning-profiles
|  ...
+--rw attachment-circuits
   +--rw ac-group-profile* [name]
   |  ...
   +--rw placement-constraints
   |  ...
   +--rw ac* [name]
      ...
      +--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* [id]
      |        +--rw id                string
      |        +--rw local-address?    inet:ip-address
      |        +--rw remote-address?   inet:ip-address
      |        +--rw profile?
      |        |       failure-detection-profile-reference
      |        +--rw holdtime?         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 security
      |  ...
      +--rw service
         ...
```

Figure 21: OAM Tree Structure

This version of the module supports BFD. The following BFD data nodes can be specified:

'id': An identifier that uniquely identifies a BFD session.

'local-address': Indicates the provider's IP address used for a BFD session.

'remote-address': Indicates the customer's IP address used for a BFD session.

'profile': Refers to a BFD profile.

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

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

5.2.5.5. Security

As shown in the tree depicted in Figure 22, the 'security' container defines a set of AC security parameters.

```
+--rw specific-provisioning-profiles
|  ...
+--rw service-provisioning-profiles
|  ...
+--rw attachment-circuits
   +--rw ac-group-profile* [name]
   |  ...
   +--rw placement-constraints
   |  ...
   +--rw ac* [name]
      ...
      +--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 provider-profile?
      |        |           encryption-profile-reference
      |        +--:(customer-profile)
      |           +--rw customer-key-chain?
      |                    key-chain:key-chain-ref
      +--rw service
         ...
```

Figure 22: Security Tree Structure

The 'security' container specifies a minimum set of encryption-related parameters that can be requested to be applied to traffic for a given AC. Typically, 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 (see Section 5.2.2.1). For example, a service provider may use IPsec when a customer requests Layer 3 encryption for an AC.

5.2.5.6. Service

The structure of the 'service' container is depicted in Figure 23.

```
+--rw specific-provisioning-profiles
|  ...
+--rw service-provisioning-profiles
|  ...
+--rw attachment-circuits
   +--rw ac-group-profile* [name]
   |  ...
   +--rw placement-constraints
   |  ...
   +--rw ac* [name]
      ...
      +--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* [profile]
            |        +--rw profile      qos-profile-reference
            |        +--rw direction?   identityref
            +--rw access-control-list
               +--rw acl-profiles
                  +--rw acl-profile* [profile]
                     +--rw profile    forwarding-profile-reference
```

Figure 23: Bandwidth Tree Structure

The 'service' container defines the following data nodes:

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

'svc-pe-to-ce-bandwidth' and 'svc-ce-to-pe-bandwidth':

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

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

Both 'svc-pe-to-ce-bandwidth' and 'svc-ce-to-pe-bandwidth' can be represented using the Committed Information Rate (CIR), the Excess Information Rate (EIR), or the Peak Information Rate (PIR). Both reuse the 'bandwidth-per-type' grouping defined in [RFC9833].

'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 Modules

6.1. The Bearer Service ("ietf-bearer-svc") YANG Module

This module uses types defined in [RFC6991], [RFC9181], and [RFC9833].

```
<CODE BEGINS> file "ietf-bearer-svc@2025-09-29.yang"

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

```
  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types, Section 4";
  }
  import ietf-vpn-common {
    prefix vpn-common;
    reference
      "RFC 9181: A Common YANG Data Model for Layer 2 and Layer 3
                 VPNs";
  }
  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>
     Editor:   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 generic YANG module for exposing
     network bearers as a service.

     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 xxx; see the
     RFC itself for full legal notices.";

  revision 2025-09-29 {
```

```
    description
      "Initial revision.";
    reference
      "RFC 9834: YANG Data Models for Bearers and Attachment
                 Circuits as a Service (ACaaS)";
  }

  // Identities

  identity identification-type {
    description
      "Base identity for identification of bearers.";
  }

  identity device-id {
    base identification-type;
    description
      "Identification of bearers based on device.";
  }

  identity site-id {
    base identification-type;
    description
      "Identification of bearers based on site.";
  }

  identity site-and-device-id {
    base identification-type;
    description
      "Identification of bearers based on site and device.";
  }

  identity custom {
    base identification-type;
    description
      "Identification of bearers based on other custom criteria.";
  }

  identity bearer-type {
    description
      "Base identity for bearers type.";
  }

  identity ethernet {
    base bearer-type;
    description
      "Ethernet.";
  }

  identity wireless {
    base bearer-type;
    description
      "Wireless.";
  }

  identity lag {
    base bearer-type;
    description
```

```
      "Link Aggregation Group (LAG).";
  }

  identity network-termination-hint {
    base vpn-common:placement-diversity;
    description
      "A hint about the termination at the network side
       is provided (e.g., geoproximity).";
  }

  identity sync-phy-type {
    description
      "Base identity for physical layer synchronization.";
  }

  identity sync-e {
    base sync-phy-type;
    description
      "Sync Ethernet (SyncE).";
    reference
      "ITU-T G.781: Synchronization layer functions for frequency
                    synchronization based on the physical layer";
  }

  // Typedef to ease referencing cross-modules

  typedef bearer-ref {
    type leafref {
      path "/bearer-svc:bearers/bearer-svc:bearer/bearer-svc:name";
    }
    description
      "Defines a type to reference a bearer.";
  }

  // Reusable groupings

  grouping location-information {
    description
      "Basic location information.";
    leaf name {
      type string;
      description
        "Provides a location name.  This data node can be mapped,
         e.g., to the 3GPP NRM IOC ManagedElement.";
    }
    leaf address {
      type string;
      description
        "Address (number and street) of the device/site.";
    }
    leaf city {
      type string;
      description
        "City of the device/site.";
    }
    leaf postal-code {
      type string;
      description
```

```
        "Postal code of the device/site.";
    }
    leaf state {
      type string;
      description
        "State of the device/site.  This leaf can also be used to
         describe a region for a country that does not have
         states.";
    }
    leaf country-code {
      type string {
        pattern '[A-Z]{2}';
      }
      description
        "Country of the device/site.
         Expressed as ISO ALPHA-2 code.";
    }
  }

  grouping placement-constraints {
    description
      "Constraints related to placement of a bearer.";
    list constraint {
      if-feature "vpn-common:placement-diversity";
      key "constraint-type";
      description
        "List of constraints.";
      leaf constraint-type {
        type identityref {
          base vpn-common:placement-diversity;
        }
        must "not(derived-from-or-self(current(), "
           + "'vpn-common:bearer-diverse') or "
           + "derived-from-or-self(current(), "
           + "'vpn-common:same-bearer'))" {
          error-message "Only bearer-specific diversity"
                      + "constraints must be provided.";
        }
        description
          "Diversity constraint type for bearers.";
      }
      container target {
        description
          "The constraint will apply against this list of
           groups.";
        choice target-flavor {
          description
            "Choice for the group definition.";
          case id {
            list group {
              key "group-id";
              description
                "List of groups.";
              leaf group-id {
                type string;
                description
                  "The constraint will apply against this
                   particular group ID.";
```

```
              }
            }
          }
          case all-bearers {
            leaf all-other-bearers {
              type empty;
              description
                "The constraint will apply against all other
                 bearers of a site.";
            }
          }
          case all-groups {
            leaf all-other-groups {
              type empty;
              description
                "The constraint will apply against all other
                 groups managed by the customer.";
            }
          }
        }
      }
    }
  }

  container locations {
    description
      "Retrieves the available provider locations for terminating
       bearers for a given customer.";

    list customer {
      key "name peer-as";
      description
        "List of locations per customer.";
      leaf name {
        type string;
        description
          "Indicates the name of the customer.";
      }
      leaf peer-as {
        type inet:as-number;
        description
          "Indicates the customer's ASN.
           0 is used when the customer does not have an ASN.";
        reference
          "RFC 7607: Codification of AS 0 Processing";
      }
      list location {
        key "name";
        config false;
        description
          "Reports the list of available locations.";
        uses location-information;
      }
    }
  }
  container bearers {
    description
      "Main container for the bearers.  The timing constraints
```

```
        indicated at the bearer level take precedence over the
        global values indicated at the bearers level.";
    uses ac-common:op-instructions;
    container placement-constraints {
      description
        "Diversity constraint type.";
      uses placement-constraints;
    }
    list bearer {
      key "name";
      description
        "Maintains a list of bearers.";
      leaf name {
        type string;
        description
          "A name that uniquely identifies a bearer for
           a given customer.";
      }
      leaf description {
        type string;
        description
          "A description of this bearer.";
      }
      leaf customer-name {
        type string;
        description
          "Indicates the name of the customer that requested this
           bearer.";
      }
      uses vpn-common:vpn-components-group;
      leaf op-comment {
        type string;
        description
          "Includes comments that can be shared with operational
           teams and that may be useful for the activation of a
           bearer.  This may include, for example, information
           about the building, level, etc.";
      }
      leaf bearer-parent-ref {
        type bearer-svc:bearer-ref;
        description
          "Specifies the parent bearer.  This can be used, e.g.,
           for a Link Aggregation Group (LAG).";
      }
      leaf-list bearer-lag-member {
        type bearer-svc:bearer-ref;
        config false;
        description
          "Reports LAG members.";
      }
      leaf sync-phy-capable {
        type boolean;
        config false;
        description
          "Indicates, when set to true, that a mechanism for physical
           layer synchronization is supported for this bearer.
           No such mechanism is supported if set to false.";
      }
```

```
        leaf sync-phy-enabled {
          type boolean;
          description
            "Indicates, when set to true, that a mechanism for physical
             layer synchronization is enabled for this bearer.  No such
             mechanism is enabled if set to false.";
        }
        leaf sync-phy-type {
          when "../sync-phy-enabled='true'";
          type identityref {
            base sync-phy-type;
          }
          description
            "Type of the physical layer synchronization that is enabled
             for this bearer.";
        }
        leaf provider-location-reference {
          type string;
          description
            "Specifies the provider's location reference.";
        }
        container customer-point {
          description
            "Base container to link the bearer existence.";
          leaf identified-by {
            type identityref {
              base identification-type;
            }
            description
              "Specifies how the customer point is identified.";
          }
          container device {
            when "derived-from-or-self(../identified-by, "
               + "'bearer-svc:device-id') or "
               + "derived-from-or-self(../identified-by, "
               + "'bearer-svc:site-and-device-id')" {
              description
                "Only applicable if identified-by is device.";
            }
            description
              "Bearer is linked to device.";
            leaf device-id {
              type string;
              description
                "Identifier for the device where that bearer belongs.";
            }
            container location {
              description
                "Location of the node.";
              uses location-information;
            }
          }
          container site {
            when "derived-from-or-self(../identified-by, "
               + "'bearer-svc:site-id') or "
               + "derived-from-or-self(../identified-by, "
               + "'bearer-svc:site-and-device-id')" {
              description
```

```
            "Only applicable if identified-by is site.";
        }
        description
          "Bearer is linked to a site.";
        leaf site-id {
          type string;
          description
            "Identifier for the site or sites where that bearer
             belongs.";
        }
        container location {
          description
            "Location of the node.";
          uses location-information;
        }
      }
      leaf custom-id {
        when "derived-from-or-self(../identified-by, "
           + "'bearer-svc:custom')" {
          description
            "Only enabled if identified-by is custom.";
        }
        type string;
        description
          "The semantics of this identifier is shared between the
           customer/provider using out-of-band means.";
      }
    }
    leaf type {
      type identityref {
        base bearer-type;
      }
      description
        "Type of the bearer (e.g., Ethernet or wireless).";
    }
    leaf test-only {
      type empty;
      description
        "When present, this indicates that this is a feasibility
         check request.  No resources are committed for such bearer
         requests.";
    }
    leaf bearer-reference {
      if-feature "ac-common:server-assigned-reference";
      type string;
      config false;
      description
        "This is an internal reference for the service provider
         to identify the bearers.";
    }
    leaf-list ac-svc-ref {
      type ac-svc:attachment-circuit-reference;
      config false;
      description
        "Specifies the set of ACs that are bound to the bearer.";
    }
    uses ac-common:op-instructions;
    uses ac-common:service-status;
```

```
    }
  }
}

<CODE ENDS>
```

6.2. The AC Service ("ietf-ac-svc") YANG Module

This module uses types defined in [RFC6991], [RFC9181], [RFC8177], and [RFC9833]. Also, the module uses the extensions defined in [RFC8341].

```
<CODE BEGINS> file "ietf-ac-svc@2025-09-29.yang"

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

  import ietf-ac-common {
    prefix ac-common;
    reference
      "RFC 9833: A Common YANG Data Model for Attachment Circuits";
  }
  import ietf-vpn-common {
    prefix vpn-common;
    reference
      "RFC 9181: A Common YANG Data Model for Layer 2 and Layer 3
                 VPNs";
  }
  import ietf-netconf-acm {
    prefix nacm;
    reference
      "RFC 8341: Network Configuration Access Control Model";
  }
  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types, Section 4";
  }
  import ietf-key-chain {
    prefix key-chain;
    reference
      "RFC 8177: YANG Data Model for Key Chains";
  }

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

     Editor:   Mohamed Boucadair
               <mailto:mohamed.boucadair@orange.com>
     Editor:   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 module for exposing
       Attachment Circuits as a Service (ACaaS).

       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 9834; see the
       RFC itself for full legal notices.";

    revision 2025-09-29 {
      description
        "Initial revision.";
      reference
        "RFC 9834: YANG Data Models for Bearers and Attachment
                   Circuits as a Service (ACaaS)";
    }

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

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

    typedef ac-group-reference {
      type leafref {
        path "/ac-svc:attachment-circuits/ac-svc:ac-group-profile"
           + "/ac-svc:name";
      }
      description
        "Defines a reference to an AC profile.";
    }

    typedef encryption-profile-reference {
      type leafref {
        path "/ac-svc:specific-provisioning-profiles"
           + "/ac-svc:valid-provider-identifiers"
           + "/ac-svc:encryption-profile-identifier/ac-svc:id";
      }
      description
```

```
      "Defines a reference to an encryption profile.";
  }

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

  typedef failure-detection-profile-reference {
    type leafref {
      path "/ac-svc:specific-provisioning-profiles"
         + "/ac-svc:valid-provider-identifiers"
         + "/ac-svc:failure-detection-profile-identifier"
         + "/ac-svc:id";
    }
    description
      "Defines a reference to a BFD profile.";
  }

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

  typedef routing-profile-reference {
    type leafref {
      path "/ac-svc:specific-provisioning-profiles"
         + "/ac-svc:valid-provider-identifiers"
         + "/ac-svc:routing-profile-identifier/ac-svc:id";
    }
    description
      "Defines a reference to a routing profile.";
  }

  typedef service-profile-reference {
    type leafref {
      path "/ac-svc:service-provisioning-profiles"
         + "/ac-svc:service-profile-identifier"
         + "/ac-svc:id";
    }
    description
      "Defines a reference to a service profile.";
  }

  /******************** Reusable groupings ********************/
  // Basic Layer 2 connection

  grouping l2-connection-basic {
```

```
    description
      "Defines Layer 2 protocols and parameters that can be
       factorized when provisioning Layer 2 connectivity
       among multiple ACs.";
    container encapsulation {
      description
        "Container for Layer 2 encapsulation.";
      leaf type {
        type identityref {
          base vpn-common:encapsulation-type;
        }
        description
          "Encapsulation type.";
      }
      container dot1q {
        when "derived-from-or-self(../type, 'vpn-common:dot1q')" {
          description
            "Only applies when the type of the tagged interface
             is 'dot1q'.";
        }
        description
          "Tagged interface.";
        uses ac-common:dot1q;
      }
      container qinq {
        when "derived-from-or-self(../type, 'vpn-common:qinq')" {
          description
            "Only applies when the type of the tagged interface
             is 'qinq'.";
        }
        description
          "Includes QinQ parameters.";
        uses ac-common:qinq;
      }
    }
  }

  // Full Layer 2 connection

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

```
        }
        description
          "Tagged interface.";
        uses ac-common:dot1q;
      }
      container priority-tagged {
        when "derived-from-or-self(../type, "
           + "'vpn-common:priority-tagged')" {
          description
            "Only applies when the type of the tagged interface is
             'priority-tagged'.";
        }
        description
          "Priority-tagged interface.";
        uses ac-common:priority-tagged;
      }
      container qinq {
        when "derived-from-or-self(../type, 'vpn-common:qinq')" {
          description
            "Only applies when the type of the tagged interface
             is 'qinq'.";
        }
        description
          "Includes QinQ parameters.";
        uses ac-common:qinq;
      }
    }
    choice l2-service {
      description
        "The Layer 2 connectivity service can be provided by
         indicating a pointer to an L2VPN or by specifying a
         Layer 2 tunnel service.";
      container l2-tunnel-service {
        description
          "Defines a Layer 2 tunnel termination.
           It is only applicable when a tunnel is required.";
        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.";
        }
      }
    }
    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.";
    }
  }

  // Basic IP connection
```

```
  grouping ip-connection-basic {
    description
      "Defines basic IP connection parameters.";
    container ipv4 {
      if-feature "vpn-common:ipv4";
      description
        "IPv4-specific parameters.";
      uses ac-common:ipv4-connection-basic;
    }
    container ipv6 {
      if-feature "vpn-common:ipv6";
      description
        "IPv6-specific parameters.";
      uses ac-common:ipv6-connection-basic;
    }
  }

  // Full IP connection

  grouping ip-connection {
    description
      "Defines IP connection parameters.";
    container ipv4 {
      if-feature "vpn-common:ipv4";
      description
        "IPv4-specific parameters.";
      uses ac-common:ipv4-connection {
        augment "ac-svc:allocation-type/static-addresses/address" {
          leaf failure-detection-profile {
            if-feature "vpn-common:bfd";
            type failure-detection-profile-reference;
            description
              "Points to a failure detection profile.";
          }
          description
            "Adds a failure detection profile.";
        }
      }
    }
    container ipv6 {
      if-feature "vpn-common:ipv6";
      description
        "IPv6-specific parameters.";
      uses ac-common:ipv6-connection {
        augment "ac-svc:allocation-type/static-addresses/address" {
          leaf failure-detection-profile {
            if-feature "vpn-common:bfd";
            type failure-detection-profile-reference;
            description
              "Points to a failure detection profile.";
          }
          description
            "Adds a failure detection profile.";
        }
      }
    }
    choice l3-service {
      description
```

```
        "The Layer 3 connectivity service can be provided by
         specifying a Layer 3 tunnel service.";
      container l3-tunnel-service {
        description
          "Defines a Layer 3 tunnel termination.
           It is only applicable when a tunnel is required.";
        leaf type {
          type identityref {
            base ac-common:l3-tunnel-type;
          }
          description
            "Selects the tunnel termination type for an AC.";
        }
      }
    }
  }

  // Routing protocol list

  grouping routing-protocol-list {
    description
      "List of routing protocols used on the AC.";
    leaf type {
      type identityref {
        base vpn-common:routing-protocol-type;
      }
      description
        "Type of routing protocol.";
    }
    list routing-profiles {
      key "id";
      description
        "Routing profiles.";
      leaf id {
        type routing-profile-reference;
        description
          "Reference to the routing profile to be used.";
      }
      leaf type {
        type identityref {
          base vpn-common:ie-type;
        }
        description
          "Import, export, or both.";
      }
    }
  }

  // Static routing with BFD

  grouping ipv4-static-rtg-with-bfd {
    description
      "Configuration specific to IPv4 static routing with
       failure protection (e.g., BFD).";
    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;
      leaf failure-detection-profile {
        if-feature "vpn-common:bfd";
        type failure-detection-profile-reference;
        description
          "Points to a failure detection profile.";
      }
      uses ac-common:service-status;
    }
  }

  grouping ipv6-static-rtg-with-bfd {
    description
      "Configuration specific to IPv6 static routing with
       failure protection (e.g., BFD).";
    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;
      leaf failure-detection-profile {
        if-feature "vpn-common:bfd";
        type failure-detection-profile-reference;
        description
          "Points to a failure detection profile.";
      }
      uses ac-common:service-status;
    }
  }

  //  BGP Service

  grouping bgp-neighbor-without-name {
    description
      "A grouping with generic parameters for configuring a BGP
       neighbor.";
    leaf remote-address {
      type inet:ip-address;
      description
        "The remote IP address of this entry's BGP peer.  This is
         a customer IP address.

         If this leaf is not present, this means that the primary
         customer IP address is used as the remote IP address.";
    }
    leaf local-address {
      type inet:ip-address;
      description
        "The provider's IP address that will be used to establish
         the BGP session.";
    }
    uses ac-common:bgp-peer-group-without-name;
    container bgp-max-prefix {
      description
        "A container for the maximum number of BGP prefixes
         allowed in the BGP session.";
```

```
      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.";
        reference
          "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
                     Section 8.2.2";
      }
    }
    uses ac-common:bgp-authentication;
    uses ac-common:op-instructions;
    uses ac-common:service-status;
  }

  grouping bgp-neighbor-with-name {
    description
      "A grouping with generic parameters for configuring a BGP
       neighbor with an identifier.";
    leaf id {
      type string;
      description
        "An identifier that uniquely identifies a neighbor.";
    }
    uses ac-svc:bgp-neighbor-without-name;
  }

  grouping bgp-neighbor-with-server-reference {
    description
      "A grouping with generic parameters for configuring a BGP
       neighbor with a reference generated by the provider.";
    leaf server-reference {
      if-feature "ac-common:server-assigned-reference";
      type string;
      config false;
      description
        "This is an internal reference for the service provider
         to identify the BGP session.";
    }
    uses ac-svc:bgp-neighbor-without-name;
  }

  grouping bgp-neighbor-with-name-server-reference {
    description
      "A grouping with generic parameters for configuring a BGP
       neighbor with an identifier and a reference generated by
       the provider.";
    leaf id {
      type string;
      description
        "An identifier that uniquely identifiers a neighbor.";
    }
    uses ac-svc:bgp-neighbor-with-server-reference;
  }
```

```
  grouping bgp-svc {
    description
      "Configuration specific to BGP.";
    container peer-groups {
      description
        "Configuration for BGP peer-groups";
      list peer-group {
        key "name";
        description
          "List of BGP peer-groups configured on the local
           system -- uniquely identified by peer-group name.";
        uses ac-common:bgp-peer-group-with-name;
        leaf local-address {
          type inet:ip-address;
          description
            "The provider's local IP address that will be used to
             establish the BGP session.";
        }
        container bgp-max-prefix {
          description
            "A container for the maximum number of BGP prefixes
             allowed in the BGP session.";
          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.";
            reference
              "RFC 4271: A Border Gateway Protocol 4 (BGP-4),
                         Section 8.2.2";
          }
        }
        uses ac-common:bgp-authentication;
      }
    }
    list neighbor {
      key "id";
      description
        "List of BGP neighbors.";
      uses ac-svc:bgp-neighbor-with-name-server-reference;
      leaf peer-group {
        type leafref {
          path "../../peer-groups/peer-group/name";
        }
        description
          "The peer-group with which this neighbor is associated.";
      }
      leaf failure-detection-profile {
        if-feature "vpn-common:bfd";
        type failure-detection-profile-reference;
        description
          "Points to a failure detection profile.";
      }
    }
  }
```

```
//  OSPF Service

grouping ospf-svc {
  description
    "Service configuration specific to OSPF.";
  uses ac-common:ospf-basic;
  uses ac-common:ospf-authentication;
  uses ac-common:service-status;
}

//  IS-IS Service

grouping isis-svc {
  description
    "Service configuration specific to IS-IS.";
  uses ac-common:isis-basic;
  uses ac-common:isis-authentication;
  uses ac-common:service-status;
}

//  RIP Service

grouping rip-svc {
  description
    "Service configuration specific to RIP routing.";
  leaf address-family {
    type identityref {
      base vpn-common:address-family;
    }
    description
      "Indicates whether IPv4, IPv6, or both address families
       are to be activated.";
  }
  uses ac-common:rip-authentication;
  uses ac-common:service-status;
}

//  VRRP Service

grouping vrrp-svc {
  description
    "Service 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.";
  }
  uses ac-common:service-status;
}

// Basic routing parameters
```

```
  grouping routing-basic {
    description
      "Defines basic parameters 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.";
      }
      uses routing-protocol-list;
      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";
          list peer-group {
            key "name";
            description
              "List of BGP peer-groups configured on the local
               system -- uniquely identified by peer-group
               name.";
            uses ac-common:bgp-peer-group-with-name;
          }
        }
      }
      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 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;
      }
```

```
      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.";
        leaf address-family {
          type identityref {
            base vpn-common:address-family;
          }
          description
            "Indicates whether IPv4, IPv6, or both address families
             are to be activated.";
        }
      }
      container vrrp {
        when "derived-from-or-self(../type, "
           + "'vpn-common:vrrp-routing')" {
          description
            "Only applies when the protocol is the Virtual Router
             Redundancy Protocol (VRRP).";
        }
        if-feature "vpn-common:rtg-vrrp";
        description
          "Configuration specific to VRRP.";
        leaf address-family {
          type identityref {
            base vpn-common:address-family;
          }
          description
            "Indicates whether IPv4, IPv6, or both address families
             are to be enabled.";
        }
      }
    }
  }

  // Full routing parameters

  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.";
      }
      uses routing-protocol-list;
      container static {
```

```
          when "derived-from-or-self(../type, "
             + "'vpn-common:static-routing')" {
            description
              "Only applies when the protocol is the static
               routing protocol.";
          }
          description
            "Configuration specific to static routing.";
          container cascaded-lan-prefixes {
            description
              "LAN prefixes from the customer.";
            uses ipv4-static-rtg-with-bfd;
            uses ipv6-static-rtg-with-bfd;
          }
        }
        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.";
          uses bgp-svc;
        }
        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 ospf-svc;
        }
        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 isis-svc;
        }
        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-svc;
      }
      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.";
        uses vrrp-svc;
      }
    }
  }

  // Encryption choice

  grouping encryption-choice {
    description
      "Container for the encryption profile.";
    choice profile {
      description
        "Choice for the encryption profile.";
      case provider-profile {
        leaf provider-profile {
          type encryption-profile-reference;
          description
            "Reference to a provider encryption profile.";
        }
      }
      case customer-profile {
        leaf customer-key-chain {
          type key-chain:key-chain-ref;
          description
            "Customer-supplied key chain.";
        }
      }
    }
  }

  // Basic security parameters

  grouping ac-security-basic {
    description
      "AC-specific security parameters.";
    container encryption {
      if-feature "vpn-common:encryption";
      description
        "Container for AC security 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.";
      uses encryption-choice;
    }
  }

  // Bandwidth parameters

  grouping bandwidth {
    description
      "Container for bandwidth.";
    container svc-pe-to-ce-bandwidth {
      if-feature "vpn-common:inbound-bw";
      description
        "From the customer site's perspective, the inbound
         bandwidth of the AC or download bandwidth from the
         service provider to the site.";
      uses ac-common:bandwidth-per-type;
    }
    container svc-ce-to-pe-bandwidth {
      if-feature "vpn-common:outbound-bw";
      description
        "From the customer site's perspective, the outbound
         bandwidth of the AC or upload bandwidth from
         the CE to the PE.";
      uses ac-common:bandwidth-per-type;
    }
  }

  // Basic AC parameters

  grouping ac-basic {
```

```
    description
      "Grouping for basic parameters for an AC.";
    leaf name {
      type string;
      description
        "A name that uniquely identifies the 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-basic;
    }
    container ip-connection {
      if-feature "ac-common:layer3-ac";
      description
        "Defines IP connection parameters.";
      uses ip-connection-basic;
    }
    container routing-protocols {
      description
        "Defines routing protocols.";
      uses routing-basic;
    }
    container oam {
      description
        "Defines the Operations, Administration, and Maintenance
         (OAM) mechanisms used.";
      container bfd {
        if-feature "vpn-common:bfd";
        description
          "Container for BFD.";
        uses ac-common:bfd;
      }
    }
    container security {
      description
        "AC-specific security parameters.";
      uses ac-security-basic;
    }
    container service {
      description
        "AC-specific bandwidth parameters.";
      leaf mtu {
        type uint32;
        units "bytes";
        description
          "Layer 2 MTU.";
      }
      uses bandwidth;
    }
  }


  // Full AC parameters

  grouping ac {
```

```
      description
        "Grouping for an AC.";
      leaf name {
        type string;
        description
          "A name of the AC.  Data models that need to reference
           an AC should use 'attachment-circuit-reference'.";
      }
      leaf-list service-profile {
        type service-profile-reference;
        description
          "A reference to a service profile.";
      }
      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;
      }
      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.";
        container bfd {
          if-feature "vpn-common:bfd";
          description
            "Container for BFD.";
          list session {
            key "id";
            description
              "List of BFD sessions.";
            leaf id {
              type string;
              description
                "A unique identifier for the BFD session.";
            }
            leaf local-address {
              type inet:ip-address;
              description
                "Provider's IP address of the BFD session.";
            }
            leaf remote-address {
              type inet:ip-address;
              description
                "Customer's IP address of the BFD session.";
            }
            leaf profile {
```

```
            type failure-detection-profile-reference;
            description
              "Points to a BFD profile.";
          }
          uses ac-common:bfd;
          uses ac-common:service-status;
        }
      }
    }
    container security {
      description
        "AC-specific security parameters.";
      uses ac-security-basic;
    }
    container service {
      description
        "AC-specific bandwidth parameters.";
      leaf mtu {
        type uint32;
        units "bytes";
        description
          "Layer 2 MTU.";
      }
      uses bandwidth;
      container qos {
        if-feature "vpn-common:qos";
        description
          "QoS configuration.";
        container qos-profiles {
          description
            "QoS profile configuration.";
          list qos-profile {
            key "profile";
            description
              "Points to a QoS profile.";
            leaf profile {
              type qos-profile-reference;
              description
                "QoS profile to be used.";
            }
            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 "profile";
```

```
            description
              "Points to an ACL profile.";
            leaf profile {
              type forwarding-profile-reference;
              description
                "Forwarding profile to be used.";
            }
          }
        }
      }
    }
  }

  // Parent and Child ACs

  grouping ac-hierarchy {
    description
      "Container for parent and Child AC references.";
    leaf-list parent-ref {
      type ac-svc:attachment-circuit-reference;
      description
        "Specifies a Parent AC that is inherited by an AC.
         In contexts where dynamic termination points are
         bound to the same AC, a Parent AC with stable
         information is created with a set of Child ACs
         to track dynamic AC information.";
    }
    leaf-list child-ref {
      type ac-svc:attachment-circuit-reference;
      config false;
      description
        "Specifies a Child AC that relies upon a Parent AC.";
    }
  }

  /******************** Main AC containers ********************/

  container specific-provisioning-profiles {
    description
      "Contains a set of valid profiles to reference for an AC.";
    uses ac-common:ac-profile-cfg;
  }
  container service-provisioning-profiles {
    description
      "Contains a set of valid profiles to reference for an AC.";
    list service-profile-identifier {
      key "id";
      description
        "List of generic service profile identifiers.";
      leaf id {
        type string;
        description
          "Identification of the service profile to be used.
           The profile only has significance within the service
           provider's administrative domain.";
      }
    }
    nacm:default-deny-write;
```

```
    }
    container attachment-circuits {
      description
        "Main container for the ACs.
         The timing constraints indicated at the 'ac' level take
         precedence over the values indicated at the
         'attachment-circuits' level.";
      list ac-group-profile {
        key "name";
        description
          "Maintains a list of profiles that are shared among
           a set of ACs.";
        uses ac;
      }
      container placement-constraints {
        description
          "Diversity constraint type.";
        uses vpn-common:placement-constraints;
      }
      leaf customer-name {
        type string;
        description
          "Indicates the name of the customer that requested these
           ACs.";
      }
      uses ac-common:op-instructions;
      list ac {
        key "name";
        description
          "Provisioning of an AC.";
        leaf customer-name {
          type string;
          description
            "Indicates the name of the customer that requested this
             AC.";
        }
        leaf description {
          type string;
          description
            "Associates a description with an AC.";
        }
        leaf test-only {
          type empty;
          description
            "When present, this indicates that this is a feasibility
             check request.  No resources are committed for such AC
             requests.";
        }
        uses ac-common:op-instructions;
        leaf role {
          type identityref {
            base ac-common:role;
          }
          description
            "Indicates whether this AC is used as UNI, NNI, etc.";
        }
        leaf-list peer-sap-id {
          type string;
```

```
        description
          "One or more peer SAPs can be indicated.";
      }
      leaf-list group-profile-ref {
        type ac-group-reference;
        description
          "A reference to an AC profile.";
      }
      uses ac-hierarchy;
      uses ac-common:redundancy-group;
      list service-ref {
        key "service-type service-id";
        config false;
        description
          "Reports the set of services that are bound to the AC.";
        leaf service-type {
          type identityref {
            base vpn-common:service-type;
          }
          description
            "Indicates the service type (e.g., L3VPN or RFC 9543
             Network Slice Service).";
          reference
            "RFC 9408: A YANG Network Data Model for Service
                       Attachment Points (SAPs), Section 5";
        }
        leaf service-id {
          type string;
          description
            "Indicates an identifier of a service instance
             of a given type that uses the AC.";
        }
      }
      leaf server-reference {
        if-feature "ac-common:server-assigned-reference";
        type string;
        config false;
        description
          "Reports an internal reference for the service provider
           to identify the AC.";
      }
      uses ac;
    }
  }
}

<CODE ENDS>
```

7. Security Considerations

This section is modeled after the template described in Section 3.7.1 of [YANG-GUIDELINES].

The "ietf-bearer-svc" and "ietf-ac-svc" YANG modules define data models that are 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.

Servers **MUST** verify that requesting clients are entitled to access and manipulate a given bearer or AC. For example, a given customer must not have access to bearers/ACs of other customers. 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 these YANG modules 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 in the "ietf-bearer-svc" module:

'placement-constraints': An attacker who is able to access this data node can modify the attributes to influence how a service is delivered to a customer, and this leads to Service Level Agreement (SLA) violations.

'bearer': An attacker who is able to access this data node can modify the attributes of bearer and thus hinder how ACs are built.

In addition, an attacker could attempt to add a new bearer or delete existing ones. An attacker may also change the requested type, whether it is for test-only, or the activation scheduling.

The following subtrees and data nodes have particular sensitivities/vulnerabilities in the "ietf-ac-svc" module:

'specific-provisioning-profiles': This container includes a set of sensitive data that influences how an AC will be 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 profiles are defined with "nacm:default-deny-write" tagging [RFC9833].

'service-provisioning-profiles': An attacker who has access to these data nodes may be able to manipulate service-specific policies to be applied for an AC.

This container is defined with "nacm:default-deny-write" tagging.

'ac': An attacker who is able to access this data node can modify the attributes of an AC (e.g., QoS, bandwidth, routing protocols, keying material), leading to malfunctioning of services that will be delivered over that AC and therefore to SLA violations. In addition, an attacker could attempt to add a new AC.

Some of the readable data nodes in these YANG modules 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 in the "ietf-bearer-svc" module:

'customer-name', 'customer-point' and 'locations': An attacker can retrieve privacy-related information about locations from where the customer is connected or can be serviced. Disclosing such information may be used to infer the identity of the customer.

The following subtrees and data nodes have particular sensitivities/vulnerabilities in the "ietf-ac-svc" module:

'customer-name', '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': An attacker can retrieve the cryptographic keys protecting the underlying connectivity services (routing, in particular). These keys could be used to inject spoofed routing advertisements.

There are no particularly sensitive RPC or action operations.

Several data nodes ('bgp', 'ospf', 'isis', 'rip', and 'customer-key-chain') rely upon the key chains described in [RFC8177] for authentication purposes. As such, the AC service 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 service model because it is not supported by the underlying device modules (e.g., [RFC8695]).

Section 5.2.5.5 specifies a set of encryption-related parameters that can be applied to traffic for a given AC.

8. IANA Considerations

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

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

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

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

Name: ietf-bearer-svc
Maintained by IANA? N
Namespace: urn:ietf:params:xml:ns:yang:ietf-bearer-svc
Prefix: bearer-svc
Reference: RFC 9834

Name: ietf-ac-svc
Maintained by IANA? N
Namespace: urn:ietf:params:xml:ns:yang:ietf-ac-svc
Prefix: ac-svc
Reference: RFC 9834

9. References

9.1. Normative References

[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>.

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

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

[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>.

[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>.

[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>.

[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>.

[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>.

[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>.

[RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <https://www.rfc-editor.org/info/rfc8792>.

[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, S., and B. Wu, "A Common YANG Data Model for Attachment Circuits", RFC 9833, DOI 10.17487/RFC9833, September 2025, <https://www.rfc-editor.org/info/rfc9833>.

9.2. Informative References

[BGP4-YANG] Jethanandani, M., Patel, K., Hares, S., and J. Haas, "YANG Model for Border Gateway Protocol (BGP-4)", Work in Progress, Internet-Draft, draft-ietf-idr-bgp-model-18, 21 October 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-bgp-model-18>.

[IEEE802.1AB] IEEE, "IEEE Standard for Local and metropolitan area networks - Station and Media Access Control Connectivity Discovery", IEEE Std 802.1AB-2016, DOI 10.1109/IEEESTD.2016.7433915, January 2016, <https://doi.org/10.1109/IEEESTD.2016.7433915>.

[IEEE802.1AX] IEEE, "IEEE Standard for Local and Metropolitan Area Networks--Link Aggregation", IEEE Std 802.1AX-2020, DOI 10.1109/IEEESTD.2020.9105034, May 2020, <https://doi.org/10.1109/IEEESTD.2020.9105034>.

[Instance-Data] "Example of AC SVC Instance Data", Commit 8081bb7, August 2024, <https://github.com/boucadair/attachment-circuit-model/blob/main/xml-examples/svc-full-instance.xml>.

[ITU-T-G.781] ITU-T, "Synchronization layer functions for frequency synchronization based on the physical layer", ITU-T Recommendation G.781, January 2024, <https://www.itu.int/rec/T-REC-G.781>.

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

[PEERING-API] Aguado, C., Griswold, M., Ramseyer, J., Servin, A., Strickx, T., and Q. Misell, "Peering API", Work in Progress, Internet-Draft, draft-ietf-grow-peering-api-01, 4 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-grow-peering-api-01>.

[RFC0826] Plummer, D., "An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware", STD 37, RFC 826, DOI 10.17487/RFC0826, November 1982, <https://www.rfc-editor.org/info/rfc826>.

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

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

[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>.

[RFC4026] Andersson, L. and T. Madsen, "Provider Provisioned Virtual Private Network (VPN) Terminology", RFC 4026, DOI 10.17487/RFC4026, March 2005, <https://www.rfc-editor.org/info/rfc4026>.

[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>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <https://www.rfc-editor.org/info/rfc4861>.

[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>.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <https://www.rfc-editor.org/info/rfc6151>.

[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>.

[RFC6952] Jethanandani, M., Patel, K., and L. Zheng, "Analysis of BGP, LDP, PCEP, and MSDP Issues According to the Keying and Authentication for Routing Protocols (KARP) Design Guide", RFC 6952, DOI 10.17487/RFC6952, May 2013, <https://www.rfc-editor.org/info/rfc6952>.

[RFC7607] Kumari, W., Bush, R., Schiller, H., and K. Patel, "Codification of AS 0 Processing", RFC 7607, DOI 10.17487/RFC7607, August 2015, <https://www.rfc-editor.org/info/rfc7607>.

[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>.

[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>.

[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>.

[RFC8309] Wu, Q., Liu, W., and A. Farrel, "Service Models Explained", RFC 8309, DOI 10.17487/RFC8309, January 2018, <https://www.rfc-editor.org/info/rfc8309>.

[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>.

[RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <https://www.rfc-editor.org/info/rfc8349>.

[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>.

[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>.

[RFC8921] Boucadair, M., Ed., Jacquenet, C., Zhang, D., and P. Georgatsos, "Dynamic Service Negotiation: The Connectivity Provisioning Negotiation Protocol (CPNP)", RFC 8921, DOI 10.17487/RFC8921, October 2020, <https://www.rfc-editor.org/info/rfc8921>.

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

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

[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>.

[RFC9835] Boucadair, M., Ed., Roberts, R., Gonzalez de Dios, O., Barguil, S., and B. Wu, "A Network YANG Data Model for Attachment Circuits", RFC 9835, DOI 10.17487/RFC9835, September 2025, <https://www.rfc-editor.org/info/rfc9835>.

[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, September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-ac-lxsm-lxnm-glue-14>.

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

Appendix A. Examples

This section includes a non-exhaustive list of examples to illustrate the use of the service models defined in this document. An example of instance data can also be found at [Instance-Data].

Some of the examples below use line wrapping per [RFC8792].

A.1. Create a New Bearer

An example of a request message body to create a bearer is shown in Figure 24.

```
{
  "ietf-bearer-svc:bearers": {
    "bearer": [
      {
        "name": "a-name-chosen-by-client",
        "description": "A bearer example",
        "customer-point": {
          "identified-by": "ietf-bearer-svc:device-id",
          "device": {
            "device-id": "CE_X_SITE_Y"
          }
        },
        "type": "ietf-bearer-svc:ethernet"
      }
    ]
  }
}
```

Figure 24: Example of a Message Body to Create a New Bearer

A 'bearer-reference' is then generated by the controller for this bearer. Figure 25 shows the example of a response message body that is sent by the controller to reply to a GET request:

```
{
  "ietf-bearer-svc:bearers": {
    "bearer": [
      {
        "name": "a-name-chosen-by-client",
        "description": "A bearer example",
        "sync-phy-capable": true,
        "customer-point": {
          "identified-by": "ietf-bearer-svc:device-id",
          "device": {
            "device-id": "CE_X_SITE_Y"
          }
        },
        "type": "ietf-bearer-svc:ethernet",
        "bearer-reference": "line-156"
      }
    ]
  }
}
```

Figure 25: Example of a Response Message Body with the Bearer Reference

Note that the response also indicates that Sync Phy mechanism is supported for this bearer.

A.2. Create an AC over an Existing Bearer

An example of a request message body to create a simple AC over an existing bearer is shown in Figure 26. The bearer reference is assumed to be known to both the customer and the network provider. Such a reference can be retrieved, e.g., following the example described in Appendix A.1 or using other means (including exchanged out-of-band or via proprietary APIs).

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

Figure 26: Example of a Message Body to Request an AC over an Existing Bearer

Figure 27 shows the message body of a GET response received from the controller and that indicates the 'cvlan-id' that was assigned for the requested AC.

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

Figure 27: Example of a Message Body of a Response to Assign a Customer VLAN (C-VLAN) ID

A.3. Create an AC for a Known Peer SAP

An example of a request to create a simple AC, when the peer SAP is known, is shown in Figure 28. In this example, the peer SAP identifier points to an identifier of an SF. The (topological) location of that SF is assumed to be known to the network controller. For example, this can be determined as part of an on-demand procedure to instantiate an SF in a cloud. That instantiated SF can be granted a connectivity service via the provider network.

```
{
  "ietf-ac-svc:attachment-circuits": {
    "ac": [
      {
        "name": "ac4585",
        "description": "An AC for a known peer SAP",
        "requested-start": "2025-12-12T05:00:00.00Z",
        "peer-sap-id": [
          "nf-termination-ip"
        ]
      }
    ]
  }
}
```

Figure 28: Example of a Message Body to Request an AC with a Peer SAP

Figure 29 shows the received GET response with the required information to connect the SF.


```
{
  "ietf-ac-svc:attachment-circuits": {
    "ac": [
      {
        "name": "ac4585",
        "description": "An AC for a known peer SAP",
        "actual-start": "2025-12-12T05:00:00.00Z",
        "peer-sap-id": [
          "nf-termination-ip"
        ],
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "tag-type": "ietf-vpn-common:c-vlan",
              "cvlan-id": 550
            }
          }
        }
      }
    ]
  }
}
```


Figure 29: Example of a Message Body of a Response to Create an AC with a Peer SAP

A.4. One CE, Two ACs

Let us consider the example of an eNodeB (CE) that is directly connected to the access routers of the mobile backhaul (see Figure 30). In this example, two ACs are needed to service the eNodeB (e.g., distinct VLANs for control and user planes).

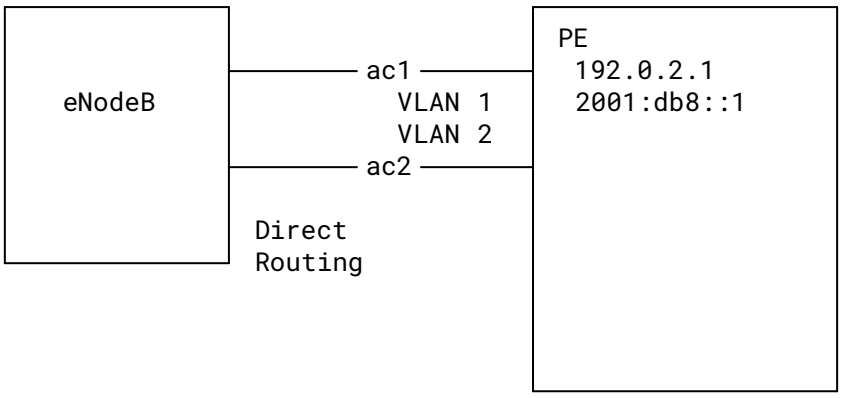


Figure 30: Example of CE-PE ACs

An example of a request to create the ACs to service the eNodeB is shown in Figure 31. This example assumes that static addressing is used for both ACs.

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

{
  "ietf-ac-svc:attachment-circuits": {
    "ac": [
      {
        "name": "ac1",
        "description": "a first AC with a same peer node",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q"
          },
          "bearer-reference": "line-156"
        },
        "ip-connection": {
          "ipv4": {
            "address-allocation-type": "ietf-ac-common:static-\
                                                          address"
          },
          "ipv6": {
            "address-allocation-type": "ietf-ac-common:static-\
                                                          address"
          }
        },
        "routing-protocols": {
          "routing-protocol": [
            {
              "id": "1",
              "type": "ietf-vpn-common:direct-routing"
            }
          ]
        }
      },
      {
        "name": "ac2",
        "description": "a second AC with a same peer node",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q"
          },
          "bearer-reference": "line-156"
        },
        "ip-connection": {
          "ipv4": {
            "address-allocation-type": "ietf-ac-common:static-\
                                                          address"
          },
          "ipv6": {
            "address-allocation-type": "ietf-ac-common:static-\
                                                          address"
          }
        },
        "routing-protocols": {
          "routing-protocol": [
            {
              "id": "1",
              "type": "ietf-vpn-common:direct-routing"
```

```
              }
            ]
          }
        }
      ]
    }
}
```

Figure 31: Example of a Message Body to Request Two ACs on the Same Link (Not Recommended)

Figure 32 shows the message body of a GET response received from the controller.

```
{
  "ietf-ac-svc:attachment-circuits": {
    "ac": [
      {
        "name": "ac1",
        "description": "a first AC with a same peer node",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "cvlan-id": 1
            }
          },
          "bearer-reference": "line-156"
        },
        "ip-connection": {
          "ipv4": {
            "local-address": "192.0.2.1",
            "prefix-length": 30,
            "address": [
              {
                "address-id": "1",
                "customer-address": "192.0.2.2"
              }
            ]
          },
          "ipv6": {
            "local-address": "2001:db8::1",
            "prefix-length": 64,
            "address": [
              {
                "address-id": "1",
                "customer-address": "2001:db8::2"
              }
            ]
          }
        },
        "routing-protocols": {
          "routing-protocol": [
            {
              "id": "1",
              "type": "ietf-vpn-common:direct-routing"
            }
          ]
        }
      },
      {
        "name": "ac2",
        "description": "a second AC with a same peer node",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "cvlan-id": 2
            }
          },
          "bearer-reference": "line-156"
```

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

Figure 32: Example of a Message Body of a Response to Create Two ACs on the Same Link (Not Recommended)

The example shown Figure 32 is not optimal as it includes many redundant data. Figure 33 shows a more compact request that factorizes all the redundant data.

```
{
  "ietf-ac-svc:attachment-circuits": {
    "ac-group-profile": [
      {
        "name": "simple-node-profile",
        "l2-connection": {
          "bearer-reference": "line-156"
        },
        "ip-connection": {
          "ipv4": {
            "local-address": "192.0.2.1",
            "prefix-length": 30,
            "address": [
              {
                "address-id": "1",
                "customer-address": "192.0.2.2"
              }
            ]
          },
          "ipv6": {
            "local-address": "2001:db8::1",
            "prefix-length": 64,
            "address": [
              {
                "address-id": "1",
                "customer-address": "2001:db8::2"
              }
            ]
          }
        },
        "routing-protocols": {
          "routing-protocol": [
            {
              "id": "1",
              "type": "ietf-vpn-common:direct-routing"
            }
          ]
        }
      }
    ],
    "ac": [
      {
        "name": "ac1",
        "description": "a first AC with a same peer node",
        "group-profile-ref": ["simple-node-profile"],
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "cvlan-id": 1
            }
          }
        }
      },
      {
        "name": "ac2",
        "description": "a second AC with a same peer node",
```

```
        "group-profile-ref": ["simple-node-profile"],
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "cvlan-id": 2
            }
          }
        }
      }
    ]
  }
}
```

Figure 33: Example of a Message Body to Request Two ACs on the Same Link (Node Profile)

A customer may request adding a new AC by simply referring to an existing per-node AC profile as shown in Figure 34. This AC inherits all the data that was enclosed in the indicated per-node AC profile (IP addressing, routing, etc.).

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

Figure 34: Example of a Message Body to Add a New AC over an Existing Link (Node Profile)

A.5. Control Precedence over Multiple ACs

When multiple ACs are requested by the same customer for the same site, the request can tag one of these ACs as 'primary' and the other ones as 'secondary'. An example of such a request is shown in Figure 36. In this example, both ACs are bound to the same 'group-id', and the 'precedence' data node is set as a function of the intended role of each AC (primary or secondary).

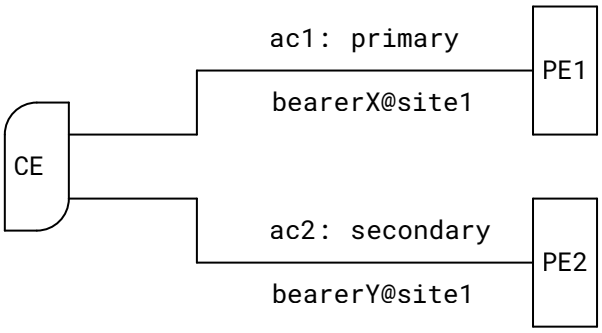


Figure 35: An Example Topology for AC Precedence Enforcement

```
{
  "ietf-ac-svc:attachment-circuits": {
    "ac": [
      {
        "name": "ac1",
        "description": "A primary AC",
        "group": [
          {
            "group-id": "1",
            "precedence": "ietf-ac-common:primary"
          }
        ],
        "l2-connection": {
          "bearer-reference": "bearerX@site1"
        }
      },
      {
        "name": "ac2",
        "description": "A secondary AC",
        "group": [
          {
            "group-id": "1",
            "precedence": "ietf-ac-common:secondary"
          }
        ],
        "l2-connection": {
          "bearer-reference": "bearerY@site1"
        }
      }
    ]
  }
}
```

Figure 36: Example of a Message Body to Associate a Precedence Level with ACs

A.6. Create Multiple ACs Bound to Multiple CEs

Figure 37 shows an example of CEs that are interconnected by a service provider network.

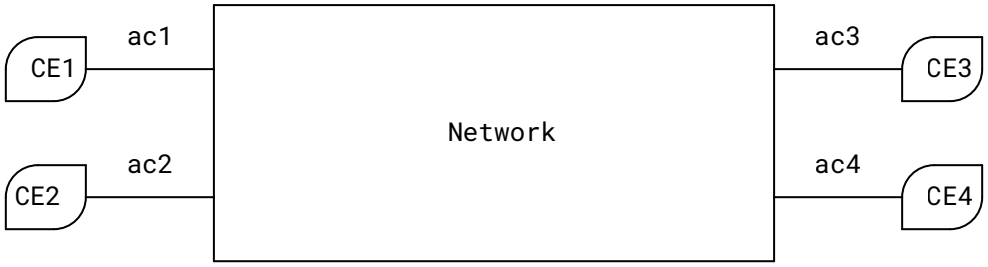


Figure 37: Network Topology Example

Let's assume that a request to instantiate the various ACs that are shown in Figure 37 is sent by the customer. Figure 38 depicts the example of the message body of a GET response that is received from the controller.

```
{
  "ietf-ac-svc:attachment-circuits": {
    "ac-group-profile": [
      {
        "name": "simple-profile",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "cvlan-id": 1
            }
          }
        }
      }
    ],
    "ac": [
      {
        "name": "ac1",
        "description": "First site",
        "group-profile-ref": [
          "simple-profile"
        ],
        "l2-connection": {
          "bearer-reference": "ce1-network"
        }
      },
      {
        "name": "ac2",
        "description": "Second Site",
        "group-profile-ref": [
          "simple-profile"
        ],
        "l2-connection": {
          "bearer-reference": "ce2-network"
        }
      },
      {
        "name": "ac3",
        "description": "Third site",
        "group-profile-ref": [
          "simple-profile"
        ],
        "l2-connection": {
          "bearer-reference": "ce3-network"
        }
      },
      {
        "name": "ac4",
        "description": "Another site",
        "group-profile-ref": [
          "simple-profile"
        ],
        "l2-connection": {
          "bearer-reference": "ce4-network"
        }
      }
    ]
```

```
    }
  }
```

Figure 38: Example of a Message Body of a Request to Create Multiple ACs Bound to Multiple CEs

A.7. Binding Attachment Circuits to an IETF Network Slice

This example shows how the AC service model complements the model defined in "A YANG Data Model for the RFC 9543 Network Slice Service" [NSSM] to connect a site to an RFC 9543 Network Slice Service.

First, Figure 39 describes the end-to-end network topology as well as the orchestration scopes:

- The topology is made up of two sites ("site1" and "site2"), interconnected via a Transport Network (e.g., IP/MPLS network). An SF is deployed within each site in a dedicated IP subnet.
- A 5G Service Management and Orchestration (SMO) is responsible for the deployment of SFs and the indirect management of a local gateway (i.e., CE).
- An IETF Network Slice Controller (NSC) [RFC9543] is responsible for the deployment of IETF Network Slices across the Transport Network.

SFs are deployed within each site.

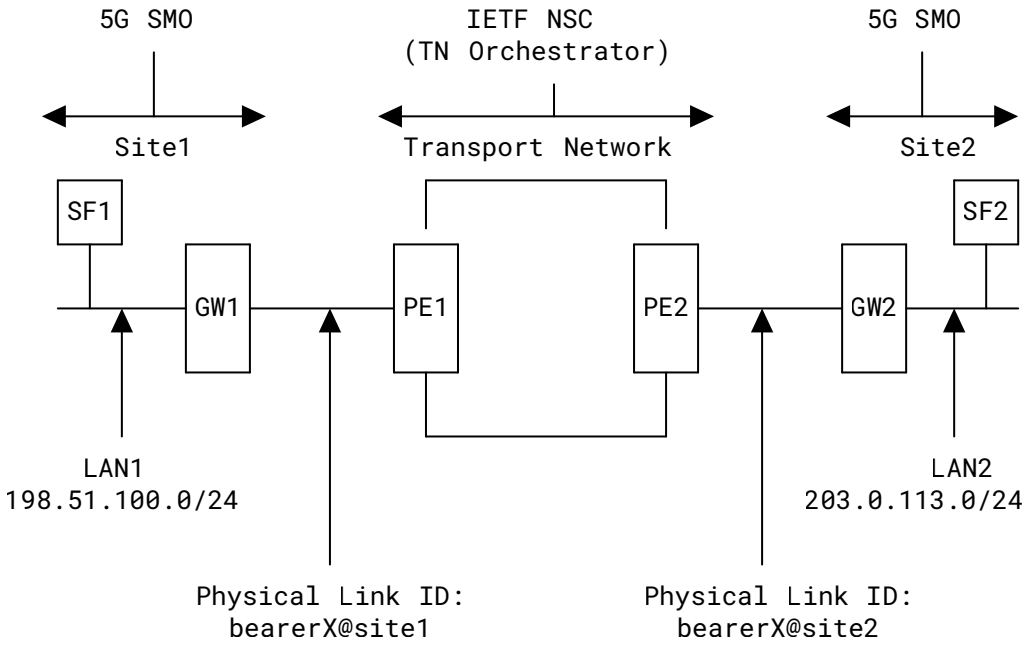
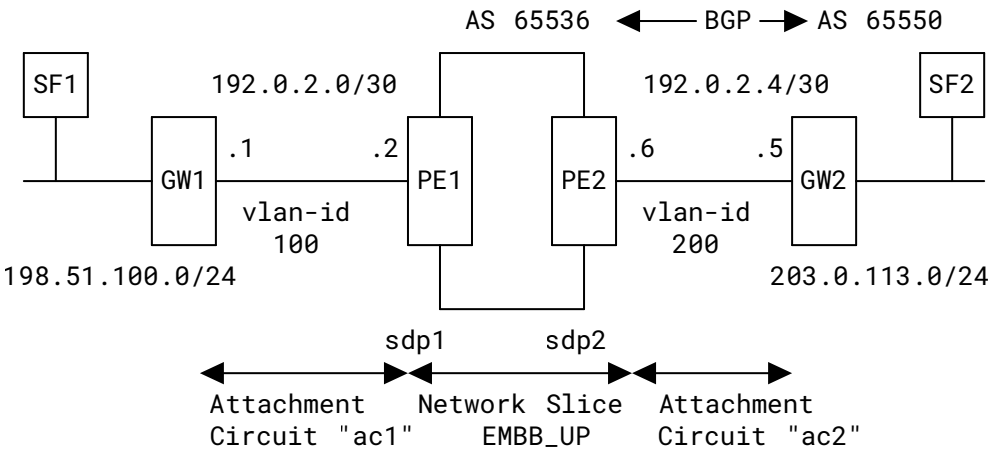


Figure 39: An Example of a Network Topology Used to Deploy Slices

Figure 40 describes the logical connectivity enforced thanks to both IETF Network Slice and ACaaS models.




```
● "ac1" properties:
- bearer-reference: bearerX@site1
- vlan-id: 100
- CE address (GW1): 192.0.2.1/30
- PE address: 192.0.2.2/30
- Routing: static 198.51.100.0/24 via
           192.0.2.1 tag primary_UP_slice

● "ac2" properties:
- bearer-reference: bearerY@site2
- vlan-id: 200
- CE address (GW2): 192.0.2.5/30
- PE address: 192.0.2.6/30
- Routing: BGP local-as: 65536 (Provider ASN)
               peer-as: 65550 (customer ASN)
               remote-address: 192.0.2.5 (Customer address)
```

Figure 40: Logical Overview

Figure 41 shows the message body of the request to create the required ACs using the ACaaS module.

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

{
  "ietf-ac-svc:attachment-circuits": {
    "ac": [
      {
        "name": "ac1",
        "description": "Connection to site1 on vlan 100",
        "requested-start": "2023-12-12T05:00:00.00Z",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "tag-type": "ietf-vpn-common:c-vlan"
            }
          },
          "bearer-reference": "bearerX@site1"
        },
        "ip-connection": {
          "ipv4": {
            "address-allocation-type": "ietf-ac-common:static-\
                                                              address"
          }
        },
        "routing-protocols": {
          "routing-protocol": [
            {
              "id": "1",
              "type": "ietf-vpn-common:static-routing",
              "static": {
                "cascaded-lan-prefixes": {
                  "ipv4-lan-prefix": [
                    {
                      "lan": "198.51.100.0/24",
                      "next-hop": "192.0.2.1",
                      "lan-tag": "primary_UP_slice"
                    }
                  ]
                }
              }
            }
          ]
        }
      },
      {
        "name": "ac2",
        "description": "Connection to site2 on vlan 200",
        "requested-start": "2023-12-12T05:00:00.00Z",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "tag-type": "ietf-vpn-common:c-vlan"
            }
          },
          "bearer-reference": "bearerY@site2"
        },
```

```
        "ip-connection": {
          "ipv4": {
            "address-allocation-type": "ietf-ac-common:static-\
                                                          address"
          }
        },
        "routing-protocols": {
          "routing-protocol": [
            {
              "id": "1",
              "type": "ietf-vpn-common:bgp-routing",
              "bgp": {
                "neighbor": [
                  {
                    "id": "1",
                    "peer-as": 65550
                  }
                ]
              }
            }
          ]
        }
      }
    ]
  }
}
```

Figure 41: Message Body of a Request to Create Required ACs

Figure 42 shows the message body of a response to a GET request received from the controller.

```
{
  "ietf-ac-svc:attachment-circuits": {
    "ac": [
      {
        "name": "ac1",
        "description": "Connection to site1 on vlan 100",
        "actual-start": "2023-12-12T05:00:00.00Z",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "tag-type": "ietf-vpn-common:c-vlan",
              "cvlan-id": 100
            }
          },
          "bearer-reference": "bearerX@site1"
        },
        "ip-connection": {
          "ipv4": {
            "local-address": "192.0.2.2",
            "prefix-length": 30,
            "address": [
              {
                "address-id": "1",
                "customer-address": "192.0.2.1"
              }
            ]
          }
        },
        "routing-protocols": {
          "routing-protocol": [
            {
              "id": "1",
              "type": "ietf-vpn-common:static-routing",
              "static": {
                "cascaded-lan-prefixes": {
                  "ipv4-lan-prefix": [
                    {
                      "lan": "198.51.100.0/24",
                      "next-hop": "192.0.2.1",
                      "lan-tag": "primary_UP_slice"
                    }
                  ]
                }
              }
            }
          ]
        }
      },
      {
        "name": "ac2",
        "description": "Connection to site2 on vlan 200",
        "actual-start": "2023-12-12T05:00:00.00Z",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
```

```
              "tag-type": "ietf-vpn-common:c-vlan",
              "cvlan-id": 200
            }
          },
          "bearer-reference": "bearerY@site2"
        },
        "ip-connection": {
          "ipv4": {
            "local-address": "192.0.2.6",
            "prefix-length": 30,
            "address": [
              {
                "address-id": "1",
                "customer-address": "192.0.2.5"
              }
            ]
          }
        },
        "routing-protocols": {
          "routing-protocol": [
            {
              "id": "1",
              "type": "ietf-vpn-common:bgp-routing",
              "bgp": {
                "neighbor": [
                  {
                    "id": "1",
                    "peer-as": 65550,
                    "local-as": 65536
                  }
                ]
              }
            }
          ]
        }
      }
    ]
  }
}
```

Figure 42: Example of a Message Body of a Response Indicating the Creation of the ACs

Figure 43 shows the message body of the request to create an RFC 9543 Network Slice Service bound to the ACs created using Figure 41. Only references to these ACs are included in the RFC 9543 Network Slice Service request.

```
{
  "ietf-network-slice-service:network-slice-services": {
    "slo-sle-templates": {
      "slo-sle-template": [
        {
          "id": "low-latency-template",
          "description": "Lowest latency forwarding behavior"
        }
      ]
    },
    "slice-service": [
      {
        "id": "Slice URLLC_UP",
        "description": "Dedicated TN Slice for URLLC-UP",
        "slo-sle-template": "low-latency-template",
        "status": {},
        "sdps": {
          "sdp": [
            {
              "id": "sdp1",
              "ac-svc-name": [
                "ac1"
              ]
            },
            {
              "id": "sdp2",
              "ac-svc-name": [
                "ac2"
              ]
            }
          ]
        }
      }
    ]
  }
}
```

Figure 43: Message Body of a Request to Create an RFC 9543 Network Slice Service Referring to the ACs

A.8. Connecting a Virtualized Environment Running in a Cloud Provider

This example (Figure 44) shows how the AC service model can be used to connect a Cloud Infrastructure to a service provider network. This example makes the following assumptions:

1. A customer (e.g., Mobile Network Team or partner) has a virtualized infrastructure running in a Cloud Provider. A simplistic deployment is represented here with a set of Virtual Machines (VMs) running in a Virtual Private Environment. The deployment and management of this infrastructure is achieved via private APIs that are supported by the Cloud Provider; this realization is out of the scope of this document.
2. The connectivity to the data center is achieved thanks to a service based on direct attachment (physical connection), which is delivered upon ordering via an API exposed by

the Cloud Provider. When ordering that connection, a unique "Connection Identifier" is generated and returned via the API.

3. The customer provisions the networking logic within the Cloud Provider based on that unique Connection Identifier (i.e., logical interfaces, IP addressing, and routing).

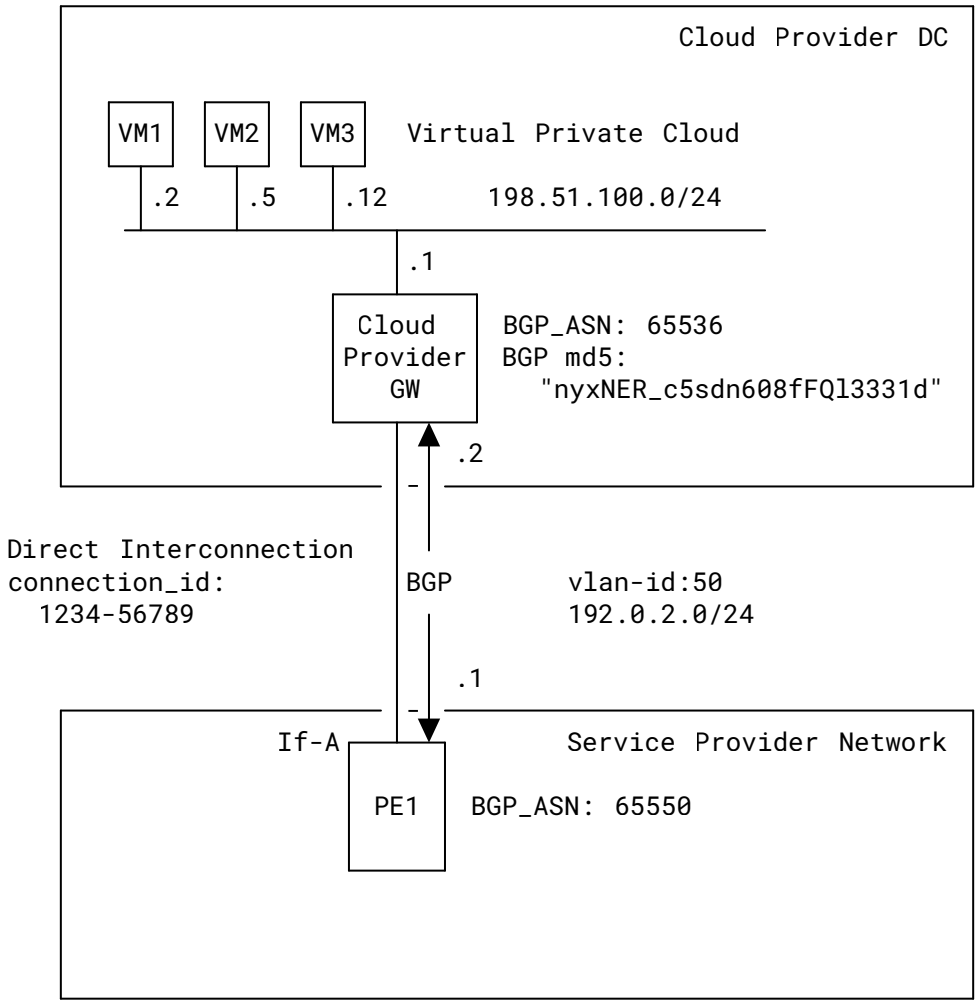


Figure 44: An Example of Realization for Connecting a Cloud Site

Figure 45 illustrates the pre-provisioning logic for the physical connection to the Cloud Provider. After this connection is delivered to the service provider, the network inventory is updated with 'bearer-reference' set to the value of the Connection Identifier.

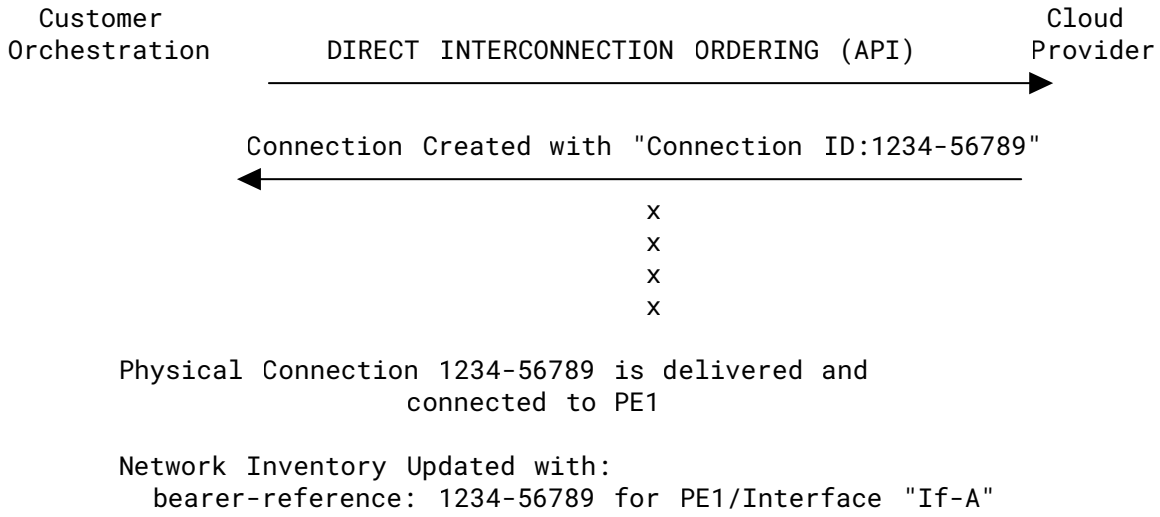


Figure 45: Illustration of Pre-Provisioning

Next, API workflows can be initiated by:

- The Cloud Provider for the configuration per item 3 above.
- The service provider network via the ACaaS model. This request can be used in conjunction with additional requests based on the L3SM (VPN provisioning) or RFC 9543 Network Slice Service model (5G hybrid cloud deployment).

Figure 46 shows the message body of the request to create the required ACs to connect the virtualized Cloud Provider (VM) using the ACaaS module.

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

{
  "ietf-ac-svc:attachment-circuits": {
    "ac": [
      {
        "name": "ac--BXT-DC-customer-VPC-foo",
        "description": "Connection to Cloud Provider BXT on \
                                                connection 1234-56789",
        "requested-start": "2023-12-12T05:00:00.00Z",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q"
          },
          "bearer-reference": "1243-56789"
        },
        "ip-connection": {
          "ipv4": {
            "address-allocation-type": "ietf-ac-common:static-\
                                                              address"
          }
        },
        "routing-protocols": {
          "routing-protocol": [
            {
              "id": "1",
              "type": "ietf-vpn-common:bgp-routing",
              "bgp": {
                "neighbor": [
                  {
                    "id": "1",
                    "peer-as": 65536
                  }
                ]
              }
            }
          ]
        }
      }
    ]
  }
}
```

Figure 46: Message Body of a Request to Create the ACs for Connecting to the Cloud Provider

Figure 47 shows the message body of the response received from the provider as a response to a query message. Note that this Cloud Provider mandates the use of MD5 authentication for establishing BGP connections.

> The module supports MD5 to basically accommodate the installed BGP base (including by some Cloud Providers). Note that MD5 suffers from the security weaknesses discussed in Section 2 of [RFC6151] and Section 2.1 of [RFC6952].

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

{
  "ietf-ac-svc:attachment-circuits": {
    "ac": [
      {
        "name": "ac--BXT-DC-customer-VPC-foo",
        "description": "Connection to Cloud Provider BXT on \
                                                connection 1234-56789",
        "actual-start": "2023-12-12T05:00:00.00Z",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "tag-type": "ietf-vpn-common:c-vlan",
              "cvlan-id": 50
            }
          },
          "bearer-reference": "1243-56789"
        },
        "ip-connection": {
          "ipv4": {
            "local-address": "192.0.2.1",
            "prefix-length": 24,
            "address": [
              {
                "address-id": "1",
                "customer-address": "192.0.2.2"
              }
            ]
          }
        },
        "routing-protocols": {
          "routing-protocol": [
            {
              "id": "1",
              "type": "ietf-vpn-common:bgp-routing",
              "bgp": {
                "neighbor": [
                  {
                    "id": "1",
                    "peer-as": 65536,
                    "local-as": 65550,
                    "authentication": {
                      "enabled": true,
                      "keying-material": {
                        "md5-keychain": "nyxNER_c5sdn608fFQl3331d"
                      }
                    }
                  }
                ]
              }
            }
          ]
        }
      }
    ]
```

```
    }
  }
```

Figure 47: Message Body of a Response to the Request to Create ACs for Connecting to the Cloud Provider

A.9. Connect Customer Network Through BGP

CE-PE routing using BGP is a common scenario in the context of MPLS VPNs and is widely used in enterprise networks. In the example depicted in Figure 48, the CE routers are customer-owned devices belonging to an AS (ASN 65536). CEs are located at the edge of the provider's network (PE) and use point-to-point interfaces to establish BGP sessions. The point-to-point interfaces rely upon a physical bearer ("line-113") to reach the provider network.

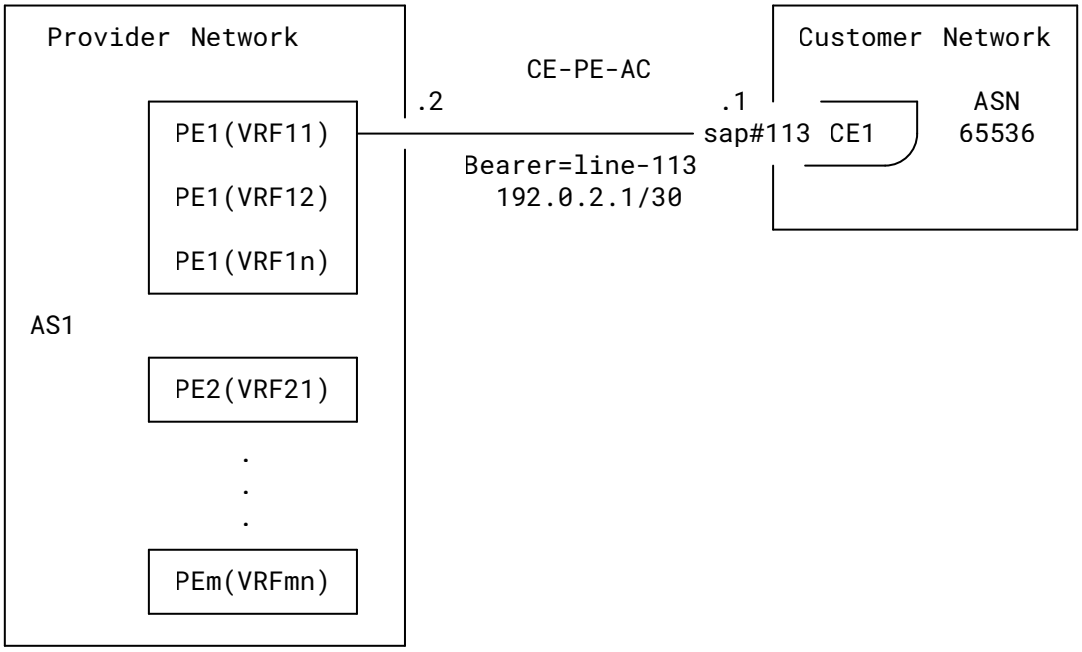


Figure 48: Illustration of Provider Network Scenario

The AC in this case uses a SAP identifier to refer to the physical interface used for the connection between the PE and the CE. The AC includes all the additional logical attributes to describe the connection between the two ends, including VLAN information and IP addressing. Also, the configuration details of the BGP session make use of peer group details instead of defining the entire configuration inside the 'neighbor' data node.

```
{
  "ietf-ac-svc:attachment-circuits": {
    "ac": [
      {
        "name": "CE-PE-AC",
        "customer-name": "Customer-4875",
        "description": "An AC between a CP and a PE",
        "peer-sap-id": [
          "sap#113"
        ],
        "ip-connection": {
          "ipv4": {
            "prefix-length": 30,
            "address": [
              {
                "address-id": "1",
                "customer-address": "192.0.2.1"
              }
            ]
          }
        },
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q"
          },
          "bearer-reference": "line-113"
        },
        "routing-protocols": {
          "routing-protocol": [
            {
              "id": "BGP-Single-Access",
              "type": "ietf-vpn-common:bgp-routing",
              "bgp": {
                "peer-groups": {
                  "peer-group": [
                    {
                      "name": "first-peer-group",
                      "peer-as": 65536,
                      "address-family": "ietf-vpn-common:ipv4"
                    }
                  ]
                },
                "neighbor": [
                  {
                    "id": "session#57",
                    "remote-address": "192.0.2.1",
                    "peer-group": "first-peer-group",
                    "status": {
                      "admin-status": {
                        "status": "ietf-vpn-common:admin-up"
                      }
                    }
                  }
                ]
              }
            }
          ]
```

```
          }
        }
      ]
    }
  }
```

Figure 49: Message Body of a Request to Create ACs for Connecting CEs to a Provider Network

This scenario allows the provider to maintain a list of ACs belonging to the same customer without requiring the full service configuration.

A.10. Interconnection via Internet Exchange Points (IXPs)

This section illustrates how to use the AC service model for interconnection purposes. To that aim, the document assumes a simplified IXP configuration without zooming into IXP deployment specifics. Let us assume that networks are interconnected via a Layer 2 facility. Let us also assume a deployment context where selective peering is in place between these networks. Networks that are interested in establishing selective BGP peerings expose a dedicated ACaaS server to the IXP to behave as an ACaaS provider. BGP is used to exchange routing information and reachability announcements between those networks. Any network operator connected to an IXP can behave as a client (i.e., initiate a BGP peering request).

This example follows the recursive deployment model depicted in Figure 4. Specifically, base AC service requests are handled locally by the IXP. However, binding BGP sessions to existing ACs involves a recursion step.

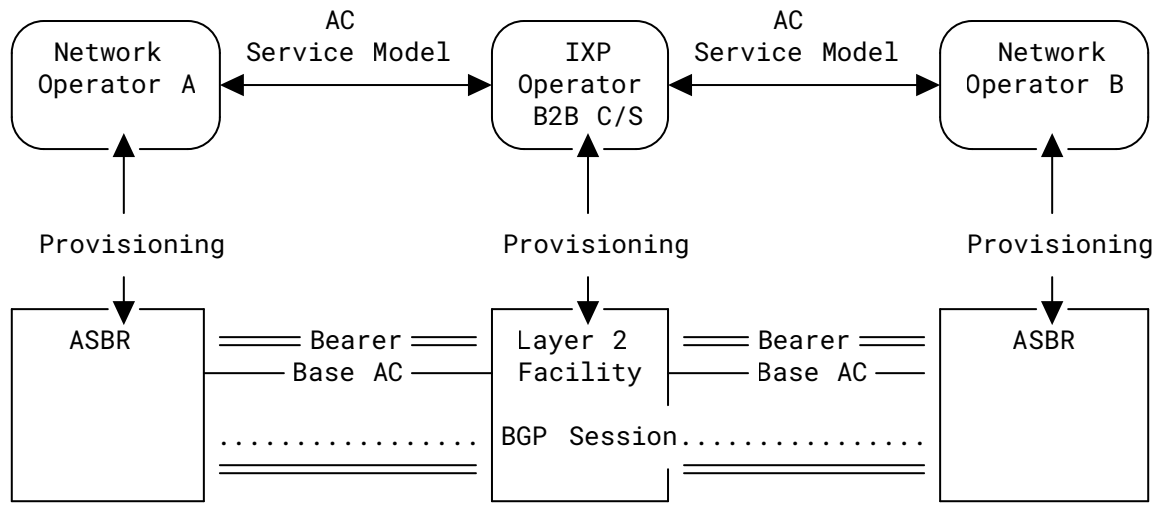


Figure 50: Recursive Deployment Example

The following subsections exemplify a deployment flow, but BGP sessions can be managed without having to systematically execute all the steps detailed hereafter.

The bearer/AC service models can be used to establish interconnection between two networks without involving an IXP.

A.10.1. Retrieve Interconnection Locations

Figure 51 shows an example message body of a request to retrieve a list of interconnection locations. The request includes a customer name and an ASN to filter out the locations.

```
{
  "ietf-bearer-svc:locations": {
    "customer": [
      {
        "name": "a future peer",
        "peer-as": 65536
      }
    ]
  }
}
```

Figure 51: Message Body of a Request to Retrieve Interconnection Locations

Figure 52 provides an example of a response to a query received from the server with a list of available interconnection locations.

```
{
  "ietf-bearer-svc:locations": {
    "customer": [
      {
        "name": "a future peer",
        "peer-as": 65536,
        "location": [
          {
            "name": "Location-X",
            "_comment": "other location attributes"
          },
          {
            "_comment": "other locations"
          }
        ]
      }
    ]
  }
}
```

Figure 52: Message Body of a Response to Retrieve Interconnection Locations

A.10.2. Create Bearers and Retrieve Bearer References

A peer can then use the location information and select the ones where it can request new bearers. As shown in Figure 53, the request includes a location reference that is known to the server (returned in Figure 52).

```
{
  "ietf-bearer-svc:bearers": {
    "bearer": [
      {
        "name": "a-name-chosen-by-client",
        "provider-location-reference": "Location-X",
        "customer-point": {
          "identified-by": "ietf-bearer-svc:device-id",
          "device": {
            "device-id": "ASBR_1_Location_X"
          }
        },
        "type": "ietf-bearer-svc:ethernet"
      }
    ]
  }
}
```

Figure 53: Message Body of a Request to Create a Bearer Using a Provider-Assigned Reference

The bearer is then activated by the server as shown in Figure 54. A 'bearer-reference' is also returned. That reference can be used for subsequent AC activation requests.

```
{
  "ietf-bearer-svc:bearers": {
    "bearer": [
      {
        "name": "a-name-chosen-by-client",
        "provider-location-reference": "Location-X",
        "customer-point": {
          "identified-by": "ietf-bearer-svc:device-id",
          "device": {
            "device-id": "ASBR_1_Location_X"
          }
        },
        "type": "ietf-bearer-svc:ethernet",
        "bearer-reference": "Location-X-Line-114",
        "status": {
          "oper-status": {
            "status": "ietf-vpn-common:op-up"
          }
        }
      }
    ]
  }
}
```

Figure 54: Message Body of a Response for a Bearer Created in a Specific Location

A.10.3. Manage ACs and BGP Sessions

As depicted in Figure 55, each network connects to the IXP switch via a bearer over which an AC is created.

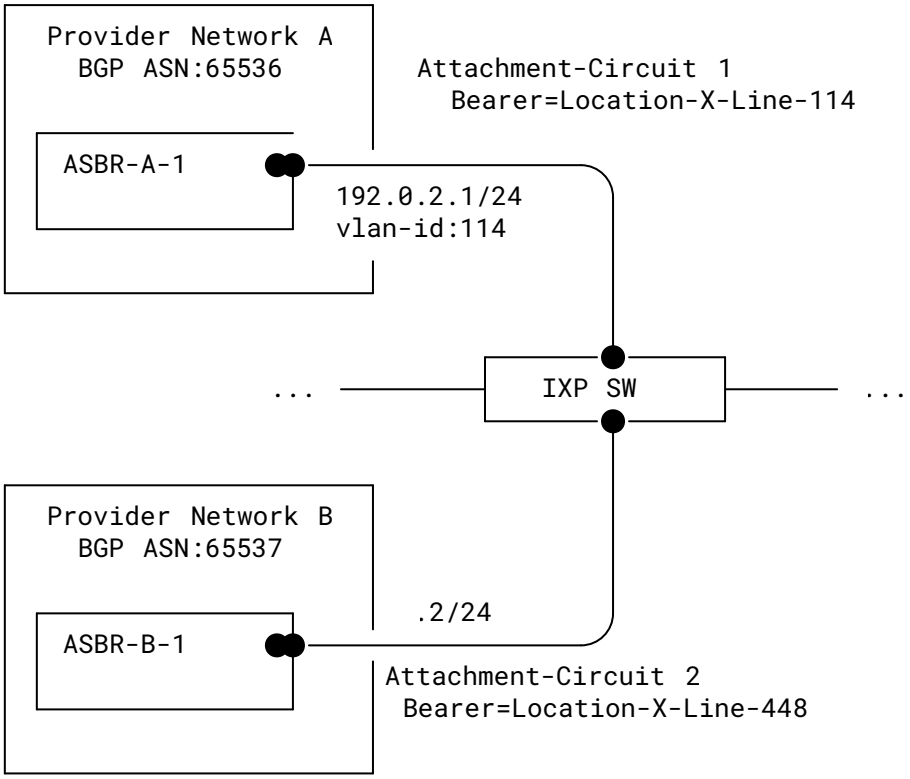


Figure 55: Simple Interconnection Topology

The AC configuration (Figure 56) includes parameters such as VLAN configuration, IP addresses, MTU, and any additional settings required for connectivity. The peering location is inferred from the 'bearer-reference'.

```
{
  "ietf-ac-svc:attachment-circuits": {
    "ac": [
      {
        "name": "Attachment Circuit 1",
        "customer-name": "Network A",
        "description": "An AC to IXP SW in Location X",
        "requested-start": "2025-12-12T05:00:00.00Z",
        "peer-sap-id": [
          "asbr-1-interface"
        ],
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q"
          },
          "bearer-reference": "Location-X-Line-114"
        }
      }
    ]
  }
}
```

Figure 56: Message Body of a Request to Create an AC to Connect to an IXP

Figure 57 shows the received response to a query with the required information for the activation of the AC.

```
{
  "ietf-ac-svc:attachment-circuits": {
    "ac": [
      {
        "name": "Attachment Circuit 1",
        "customer-name": "Network A",
        "description": "An AC to IXP SW in Location X",
        "role": "ietf-ac-common:public-nni",
        "actual-start": "2025-12-12T05:00:00.00Z",
        "peer-sap-id": [
          "asbr-1-interface"
        ],
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "tag-type": "ietf-vpn-common:c-vlan",
              "cvlan-id": 114
            }
          },
          "bearer-reference": "Location-X-Line-114"
        },
        "ip-connection": {
          "ipv4": {
            "prefix-length": 24,
            "address": [
              {
                "address-id": "1",
                "customer-address": "192.0.2.1"
              }
            ]
          }
        }
      }
    ]
  }
}
```


Figure 57: Message Body of a Response to an AC Request to Connect to an IXP

Once the ACs are established, BGP peering sessions can be configured between routers of the participating networks. BGP sessions can be established via a route server or between two networks. For the sake of illustration, let us assume that BGP sessions are established directly between two networks. Figure 58 shows an example of a request to add a BGP session to an existing AC. The properties of that AC are not repeated in this request because that information is already communicated during the creation of the AC.

```
{
  "ietf-ac-svc:attachment-circuits": {
    "ac": [
      {
        "name": "Attachment Circuit 1",
        "routing-protocols": {
          "routing-protocol": [
            {
              "id": "BGP",
              "type": "ietf-vpn-common:bgp-routing",
              "bgp": {
                "neighbor": [
                  {
                    "id": "Session-Network-B",
                    "remote-address": "192.0.2.1",
                    "local-as": 65537,
                    "peer-as": 65536,
                    "address-family": "ietf-vpn-common:ipv4",
                    "authentication": {
                      "enabled": true,
                      "keying-material": {
                        "key-id": 1,
                        "key": "test##"
                      }
                    },
                    "status": {
                      "admin-status": {
                        "status": "ietf-vpn-common:admin-up"
                      }
                    }
                  }
                ]
              }
            }
          ]
        }
      }
    ]
  }
}
```

Figure 58: Message Body of a Request to Create a BGP Session over an AC

Figure 59 provides the example of a response that indicates that the request is awaiting validation. The response also includes a server-assigned reference for this BGP session.

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

{
  "ietf-ac-svc:attachment-circuits": {
    "ac": [
      {
        "name": "Attachment Circuit 1",
        "role": "ietf-ac-common:public-nni",
        "routing-protocols": {
          "routing-protocol": [
            {
              "id": "BGP",
              "type": "ietf-vpn-common:bgp-routing",
              "bgp": {
                "neighbor": [
                  {
                    "id": "Session-Network-B",
                    "server-reference": "peering-svc-45857",
                    "local-address": "192.0.2.2",
                    "remote-address": "192.0.2.1",
                    "local-as": 65537,
                    "peer-as": 65536,
                    "address-family": "ietf-vpn-common:ipv4",
                    "authentication": {
                      "enabled": true,
                      "keying-material": {
                        "key-id": 1,
                        "key": "test##"
                      }
                    },
                    "status": {
                      "admin-status": {
                        "status": "ietf-ac-common:awaiting-\
                                                    validation"
                      }
                    }
                  }
                ]
              }
            }
          ]
        }
      }
    ]
  }
}
```

Figure 59: Message Body of a Response for a BGP Session Awaiting Validation

Once validation is accomplished, a status update is communicated back to the requestor. The BGP session can then be established over the AC. The BGP session configuration includes parameters such as neighbor IP addresses, ASNs, authentication settings (if required), etc. The configuration is triggered at each side of the BGP connection (i.e., peer ASBRs).

```
{
  "ietf-ac-svc:routing-protocols": {
    "routing-protocol": [
      {
        "id": "BGP",
        "type": "ietf-vpn-common:bgp-routing",
        "bgp": {
          "neighbor": [
            {
              "id": "Session-Network-B",
              "server-reference": "peering-svc-45857",
              "local-address": "192.0.2.2",
              "remote-address": "192.0.2.1",
              "local-as": 65537,
              "peer-as": 65536,
              "address-family": "ietf-vpn-common:ipv4",
              "authentication": {
                "enabled": true,
                "keying-material": {
                  "key-id": 1,
                  "key": "test##"
                }
              },
              "status": {
                "admin-status": {
                  "status": "ietf-ac-common:up"
                }
              }
            },
            {
              "id": "Session-Network-C",
              "server-reference": "peering-svc-7866",
              "local-address": "192.0.2.3",
              "remote-address": "192.0.2.1",
              "local-as": 65538,
              "peer-as": 65536,
              "address-family": "ietf-vpn-common:ipv4",
              "authentication": {
                "enabled": true,
                "keying-material": {
                  "key-id": 1,
                  "key": "##test##"
                }
              },
              "status": {
                "admin-status": {
                  "status": "ietf-ac-common:up"
                }
              }
            },
            {
              "_comment": "other active BGP sessions over the AC"
            }
          ]
        }
      }
    ]
```

```
    }
  }
```

Figure 60: Message Body of a Response to Report All Active BGP Sessions over an AC

A.11. Connectivity of Cloud Network Functions

A.11.1. Scope

This section demonstrates how the AC service model permits managing connectivity requirements for complex Network Functions (NFs) -- containerized or virtualized -- that are typically deployed in telco networks. This integration leverages the concept of "Parent AC" to decouple physical and logical connectivity so that several ACs can share Layer 2 and Layer 3 resources. This approach provides flexibility, scalability, and API stability.

The NFs have the following characteristics:

- The NF is distributed on a set of compute nodes with scaled-out and redundant instances.
- The NF has two distinct type of instances: user plane ("nf-up") and routing control plane ("nf-cp").
- The user plane component can be distributed among the first 8 compute nodes ("compute-01" to "compute-08") to achieve high performance.
- The control plane is deployed in a redundant fashion on two instances running on distinct compute nodes ("compute-09" and "compute-10").
- The NF is attached to distinct networks, each making use of a dedicated VLAN. These VLANs are therefore instantiated as separate ACs. From a realization standpoint, the NF interface connectivity is generally provided thanks to MacVLAN or Single Root I/O Virtualization (SR-IOV). For the sake of simplicity, only two VLANs are presented in this example; additional VLANs are configured following a similar logic.

A.11.2. Physical Infrastructure

Figure 61 describes the physical infrastructure. The compute nodes (customer) are attached to the provider infrastructure thanks to a set of physical links on which ACs are provisioned (i.e., "compute-XX-nicY"). The provider infrastructure can be realized in multiple ways, such as IP Fabric and Layer 2/3 Edge Routers. This document does not intend to detail these aspects.

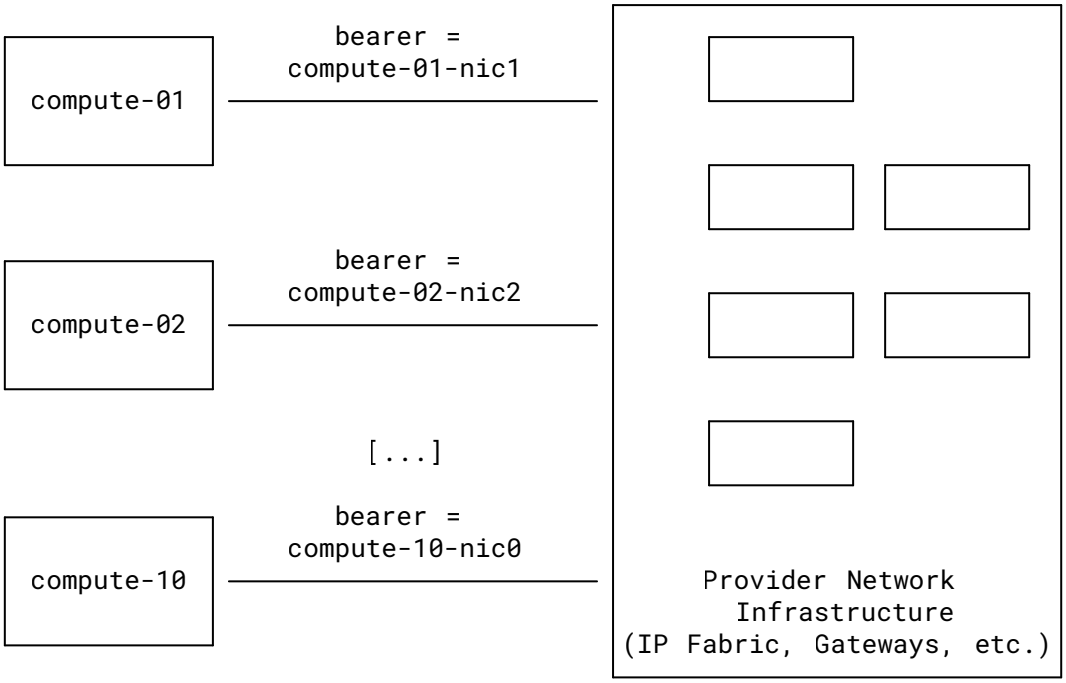


Figure 61: Example Physical Topology for Cloud Deployment

A.11.3. NFs Deployment

The NFs are deployed on this infrastructure in the following way:

- Configuration of a Parent AC as a centralized attachment for "vlan 100". The Parent AC captures Layer 2 and Layer 3 properties for this VLAN: vlan-id, IP default gateway and subnet, IP address pool for NFs endpoints, static routes with BFD to user plane, and BGP configuration to control plane NFs. In addition, the IP addresses of the user plane ("nf-up") instances are protected using BFD.
- Configuration of a Parent AC as a centralized attachment for "vlan 200". This VLAN is for Layer 2 connectivity between NFs (no IP configuration in the provider network).
- Child ACs binding bearers to Parent ACs for "vlan 100" and "vlan 200".
- The deployment of the network service to all compute nodes ("compute-01" to "compute-10"), even though the NF is not instantiated on "compute-07"/"compute-08". This approach permits handling compute failures and scale-out scenarios in a reactive and flexible fashion thanks to a pre-provisioned networking logic.

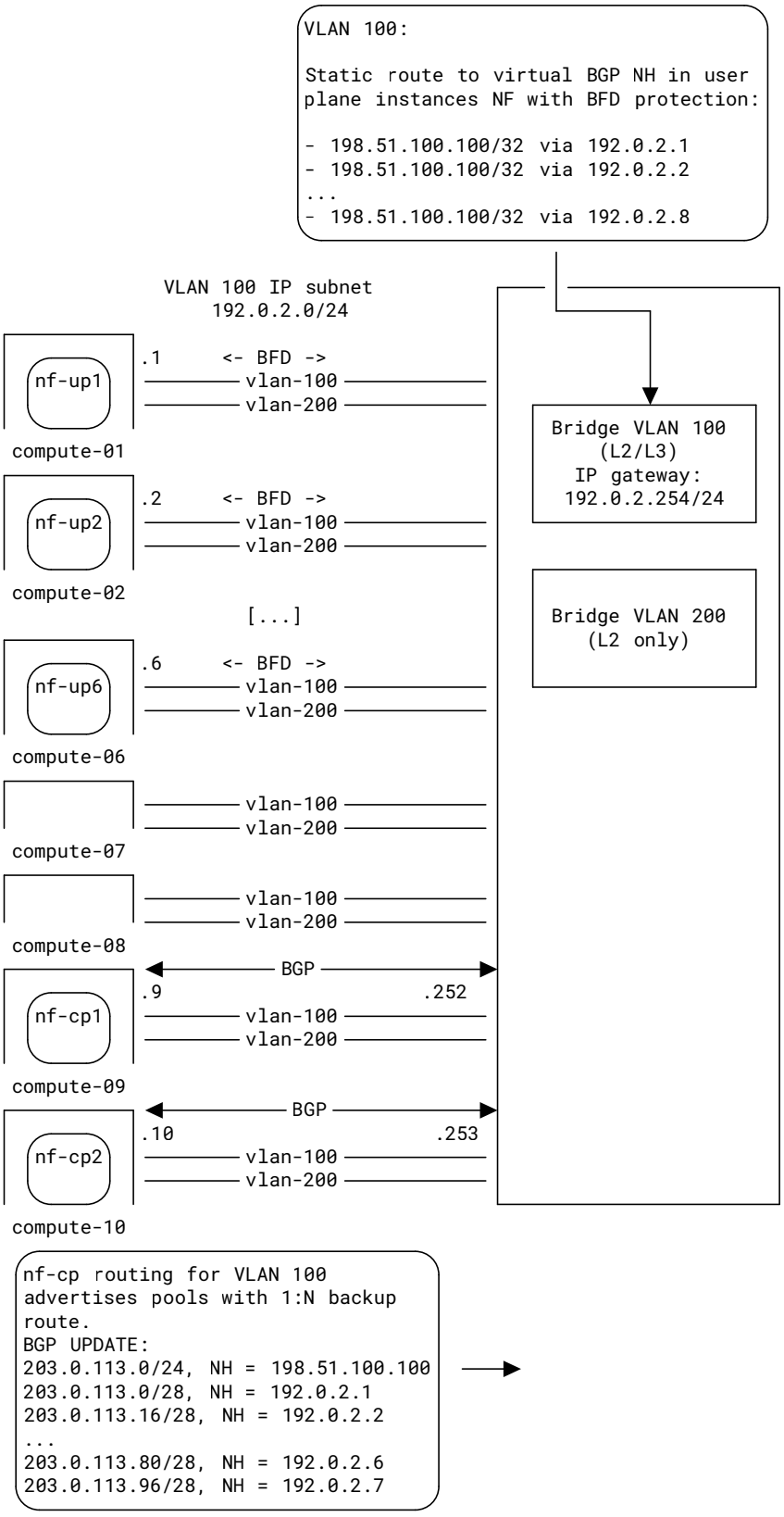

VLAN 100:
Static route to virtual BGP NH in user plane instances NF with BFD protection:
- 198.51.100.100/32 via 192.0.2.1
- 198.51.100.100/32 via 192.0.2.2
...
- 198.51.100.100/32 via 192.0.2.8
VLAN 100 IP subnet
192.0.2.0/24
.1 <- BFD ->
nf-up1
vlan-100
vlan-200
compute-01
.2 <- BFD ->
nf-up2
vlan-100
vlan-200
compute-02
[...]
.6 <- BFD ->
nf-up6
vlan-100
vlan-200
compute-06
vlan-100
vlan-200
compute-07
vlan-100
vlan-200
compute-08
BGP
.9
.252
nf-cp1
vlan-100
vlan-200
compute-09
BGP
.10
.253
nf-cp2
vlan-100
vlan-200
compute-10
Bridge VLAN 100
(L2/L3)
IP gateway:
192.0.2.254/24
Bridge VLAN 200
(L2 only)
nf-cp routing for VLAN 100
advertises pools with 1:N backup
route.
BGP UPDATE:
203.0.113.0/24, NH = 198.51.100.100
203.0.113.0/28, NH = 192.0.2.1
203.0.113.16/28, NH = 192.0.2.2
...
203.0.113.80/28, NH = 192.0.2.6
203.0.113.96/28, NH = 192.0.2.7

Figure 62: Logical Topology of the NFs Deployment

For readability, the payload is displayed as a single JSON file (Figure 63). In practice, several API calls may take place to initialize these resources (e.g., GET requests from the customer to retrieve the IP address pools for NFs on "vlan 100" thanks to parent configuration and BGP configuration and POST extra routes for user planes and BFD).

Note that no individual IP addresses are assigned for the NF user plane instances (i.e., no 'customer-address' in the Child AC). The assignment of IP addresses to the NF endpoints is managed by the Cloud Infrastructure IP Address Management (IPAM) based on the 'customer-address' IP address pool "192.0.2.1-200". Like in any conventional LAN-facing scenario, it is assumed that the actual binding of IP endpoints to logical attachments (here Child ACs) relies on a dedicated protocol logic (typically, Address Resolution Protocol (ARP) [RFC0826] or Neighbor Discovery [RFC4861]) and is not captured in the data model. Hence, the IP addresses displayed for NF user plane instances are simply examples to illustrate a realization approach. Note also that the control plane is defined with static IP address assignments on a given AC/bearer to illustrate another deployment alternative.

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

{
  "ietf-ac-svc:specific-provisioning-profiles": {
    "valid-provider-identifiers": {
      "failure-detection-profile-identifier": [
        {
          "id": "single-hop-bfd-user-plane"
        }
      ]
    }
  },
  "ietf-ac-svc:attachment-circuits": {
    "ac": [
      {
        "name": "parent-vlan-100",
        "description": "This parent represents a bridge with L3 \
                        interface (IRB) to connect NF in vlan 100",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "cvlan-id": 100
            }
          }
        },
        "ip-connection": {
          "ipv4": {
            "virtual-address": "192.0.2.254",
            "prefix-length": 24,
            "customer-addresses": {
              "address-pool": [
                {
                  "pool-id": "pool-1",
                  "start-address": "192.0.2.1",
                  "end-address": "192.0.2.200"
                }
              ]
            }
          }
        },
        "routing-protocols": {
          "routing-protocol": [
            {
              "id": "1",
              "type": "ietf-vpn-common:static-routing",
              "static": {
                "cascaded-lan-prefixes": {
                  "ipv4-lan-prefix": [
                    {
                      "lan": "198.51.100.100/32",
                      "next-hop": "192.0.2.1",
                      "lan-tag": "virtual-next-hop",
                      "failure-detection-profile": "single-hop-bfd-\
                                                   user-plane"
                    },
                    {
```

```
                    "lan": "198.51.100.100/32",
                    "next-hop": "192.0.2.2",
                    "lan-tag": "virtual-next-hop",
                    "failure-detection-profile": "single-hop-bfd-\
                                                      user-plane"
                  },
                  {
                    "_comment": "192.0.2.3-192.0.2.7 are not \
                                                       displayed"
                  },
                  {
                    "lan": "198.51.100.100/32",
                    "next-hop": "192.0.2.8",
                    "lan-tag": "virtual-next-hop",
                    "failure-detection-profile": "single-hop-bfd-\
                                                      user-plane"
                  }
                ]
              }
            }
          },
          {
            "id": "2",
            "type": "ietf-vpn-common:bgp-routing",
            "bgp": {
              "peer-groups": {
                "peer-group": [
                  {
                    "name": "peer-nf-cp-vlan-100-gw1",
                    "local-as": 65536,
                    "peer-as": 65537,
                    "local-address": "192.0.2.252"
                  },
                  {
                    "name": "peer-nf-cp-vlan-100-gw2",
                    "local-as": 65536,
                    "peer-as": 65537,
                    "local-address": "192.0.2.253"
                  }
                ]
              },
              "neighbor": [
                {
                  "id": "gw1-cp1",
                  "remote-address": "192.0.2.101",
                  "peer-group": "peer-nf-cp-vlan-100-gw1"
                },
                {
                  "id": "gw1-cp2",
                  "remote-address": "192.0.2.102",
                  "peer-group": "peer-nf-cp-vlan-100-gw1"
                },
                {
                  "id": "gw2-cp1",
                  "remote-address": "192.0.2.101",
                  "peer-group": "peer-nf-cp-vlan-100-gw2"
                },
                {
```

```
                  "id": "gw2-cp2",
                  "remote-address": "192.0.2.102",
                  "peer-group": "peer-nf-cp-vlan-100-gw2"
                }
              ]
            }
          }
        ]
      },
      "oam": {
        "bfd": {
          "session": [
            {
              "id": "bfd-gw1-nf-up1",
              "local-address": "192.0.2.252",
              "remote-address": "192.0.2.1",
              "profile": "single-hop-bfd-user-plane"
            },
            {
              "id": "bfd-gw2-nf-up1",
              "local-address": "192.0.2.253",
              "remote-address": "192.0.2.1",
              "profile": "single-hop-bfd-user-plane"
            },
            {
              "id": "bfd-gw1-nf-up2",
              "local-address": "192.0.2.252",
              "remote-address": "192.0.2.2",
              "profile": "single-hop-bfd-user-plane"
            },
            {
              "id": "bfd-gw2-nf-up2",
              "local-address": "192.0.2.253",
              "remote-address": "192.0.2.2",
              "profile": "single-hop-bfd-user-plane"
            },
            {
              "_comment": "192.0.2.3-192.0.2.7 sessions are not \
                                                        displayed"
            },
            {
              "id": "bfd-gw1-nf-up8",
              "local-address": "192.0.2.252",
              "remote-address": "192.0.2.8",
              "profile": "single-hop-bfd-user-plane"
            },
            {
              "id": "bfd-gw2-nf-up8",
              "local-address": "192.0.2.253",
              "remote-address": "192.0.2.8",
              "profile": "single-hop-bfd-user-plane"
            }
          ]
        }
      }
    },
    {
      "name": "parent-vlan-200",
```

```
        "description": "This parent represents a bridge that \
                                          connects a NF in vlan 200",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "cvlan-id": 200
            }
          }
        }
      },
      {
        "name": "ac-nf-up-01-vlan-100",
        "description": "attachment to NF-up instance 1 in vlan 100",
        "parent-ref": ["parent-vlan-100"],
        "l2-connection": {
          "bearer-reference": "compute-01-nic1"
        }
      },
      {
        "name": "ac-nf-up-02-vlan-100",
        "description": "attachment to NF-up instance 2 in vlan 100",
        "parent-ref": ["parent-vlan-100"],
        "l2-connection": {
          "bearer-reference": "compute-02-nic2"
        }
      },
      {
        "_comment": "ac-nf-up-03-vlan-100 to ac-nf-up-07-vlan-100 \
                                                        are hidden"
      },
      {
        "name": "ac-nf-up-08-vlan-100",
        "description": "attachment to NF-up instance 10 in vlan 100",
        "parent-ref": ["parent-vlan-100"],
        "l2-connection": {
          "bearer-reference": "compute-08-nic1"
        }
      },
      {
        "name": "ac-nf-cp-01-vlan-100",
        "description": "attachment to NF-CP instance 1 in vlan 100",
        "parent-ref": ["parent-vlan-100"],
        "l2-connection": {
          "bearer-reference": "compute-09-nic0"
        },
        "ip-connection": {
          "ipv4": {
            "prefix-length": 24,
            "address": [
              {
                "address-id": "1",
                "customer-address": "192.0.2.101"
              }
            ]
          }
        }
      },
```

```
      {
        "name": "ac-nf-cp-02-vlan-100",
        "description": "attachment to NF-CP instance 2 in vlan 100",
        "parent-ref": ["parent-vlan-100"],
        "l2-connection": {
          "bearer-reference": "compute-10-nic0"
        },
        "ip-connection": {
          "ipv4": {
            "prefix-length": 24,
            "address": [
              {
                "address-id": "1",
                "customer-address": "192.0.2.102"
              }
            ]
          }
        }
      },
      {
        "name": "ac-nf-up-1-vlan-200",
        "description": "attachment to NF-up instance 1 in vlan 200",
        "parent-ref": ["parent-vlan-200"],
        "l2-connection": {
          "bearer-reference": "compute-01-nic1"
        }
      },
      {
        "_comment": "ac-nf-up-2-vlan-200 to ac-nf-cp-01-vlan-200 \
                                                 are not displayed"
      },
      {
        "name": "ac-nf-cp-2-vlan-200",
        "description": "attachment to NF-CP instance 2 in vlan 200",
        "parent-ref": ["parent-vlan-200"],
        "l2-connection": {
          "bearer-reference": "compute-10-nic0"
        }
      }
    ]
  }
}
```

Figure 63: Message Body for the Configuration of the NF ACs

A.11.4. NF Failure and Scale-Out

Assuming a failure of "compute-01", the instance "nf-up-1" can be redeployed to "compute-07" by the NF / cloud orchestration. The NFs can be scaled-out thanks to the creation of an extra instance "nf-up7" on "compute-08". Since connectivity is pre-provisioned, these operations happen without any API calls. In other words, this redeployment is transparent from the perspective of the configuration of the provider network.

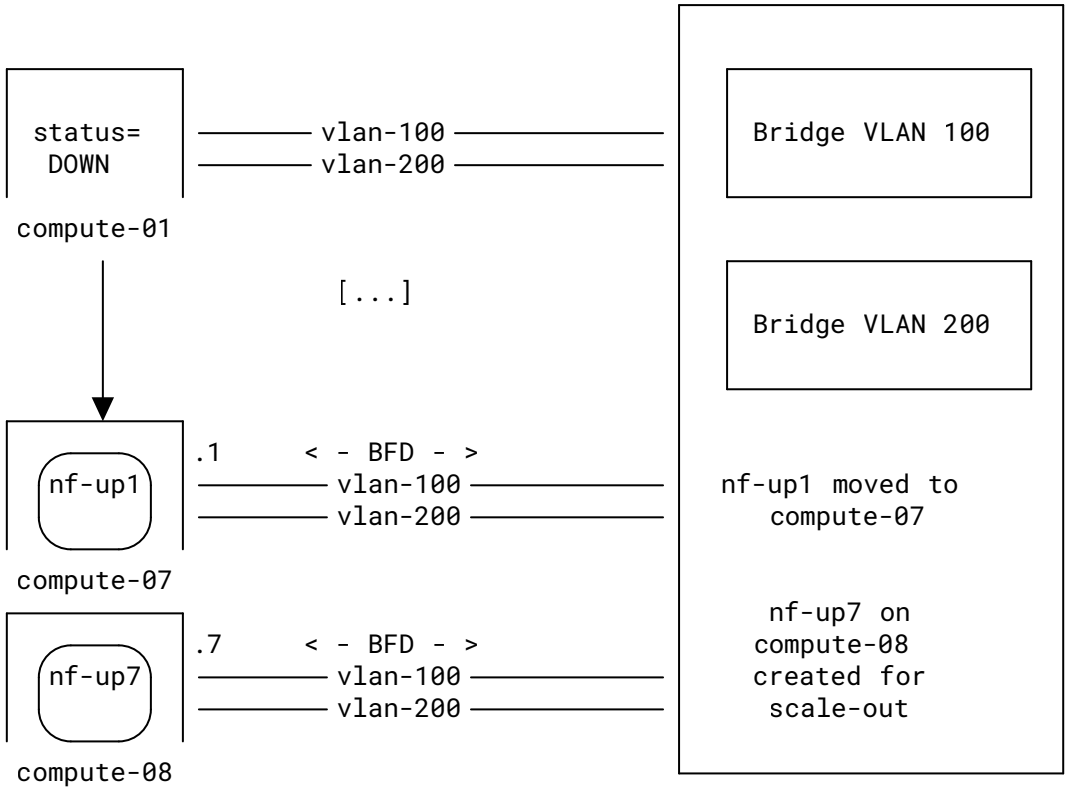


Figure 64: Example of Compute Failure and Scale-Out

Finally, the addition or deletion of compute nodes in the deployment ("compute-11", "compute-12", etc.) involves merely changes on Child ACs and possible routing on the Parent AC. In any case, the Parent AC is a stable identifier, which can be consumed as a reference by end-to-end service models for VPN configuration such as AC Glue [RFC9836], RFC 9543 Network Slice Service [NSSM], etc. This decoupling to a stable identifier provides great benefits in terms of scalability and flexibility since once the reference with the Parent AC is implemented, no API call involving the VPN model is needed for any modification in the cloud.

A.12. BFD and Static Addressing

Figure 65 shows a topology example of a set of CEs connected to a provider network via dedicated bearers. Each of these CEs maintains two BFD sessions with the provider network.

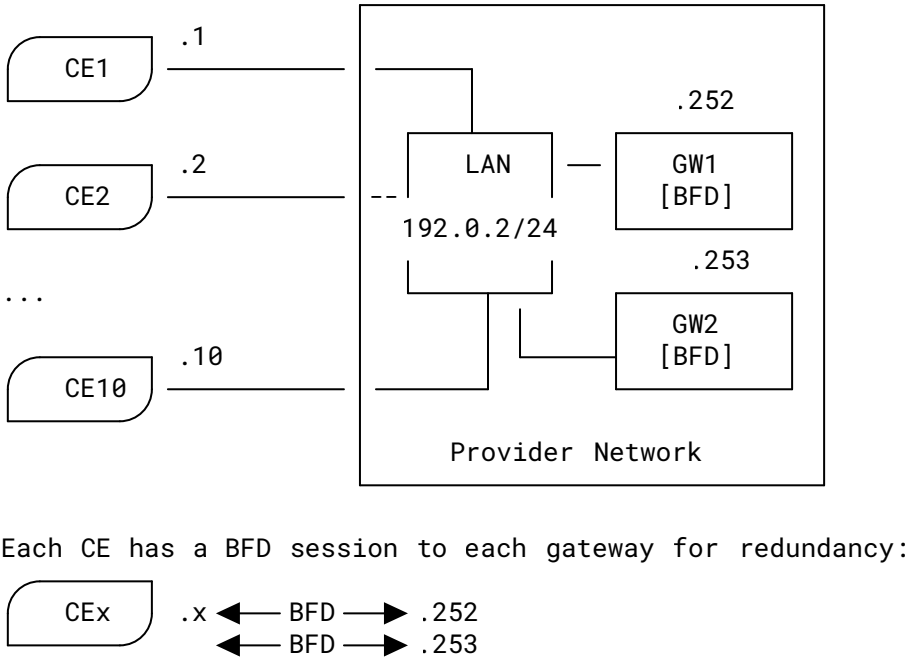


Figure 65: Example of Static Addressing with BFD

Figure 66 shows the message body of the ACaaS configuration to enable the target architecture shown in Figure 65. This example uses an AC group profile to factorize common data between all involved ACs. It also uses Child ACs that inherit the properties of two Parent ACs, each terminating in a separate gateway in the provider network.

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

{
  "ietf-ac-svc:specific-provisioning-profiles": {
    "valid-provider-identifiers": {
      "failure-detection-profile-identifier": [
        {
          "id": "single-hop-bfd"
        }
      ]
    }
  },
  "ietf-ac-svc:attachment-circuits": {
    "ac-group-profile": [
      {
        "name": "profile-vlan-100",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "cvlan-id": 100
            }
          }
        },
        "ip-connection": {
          "ipv4": {
            "virtual-address": "192.0.2.254",
            "prefix-length": 24,
            "address": [
              {
                "address-id": "ce1",
                "customer-address": "192.0.2.1",
                "failure-detection-profile": "single-hop-bfd"
              },
              {
                "address-id": "ce2",
                "customer-address": "192.0.2.2",
                "failure-detection-profile": "single-hop-bfd"
              },
              {
                "_comment": "ce3 to ce9 are not displayed"
              },
              {
                "address-id": "ce10",
                "customer-address": "192.0.2.10",
                "failure-detection-profile": "single-hop-bfd"
              }
            ]
          }
        }
      }
    ],
    "ac": [
      {
        "name": "parent-vlan-100-gw1",
        "description": "This parent represents a bridge with Layer \
                        3 interface (IRB) to connect NFs in VLAN 100",
```

```
        "group-profile-ref": [
          "profile-vlan-100"
        ],
        "ip-connection": {
          "ipv4": {
            "local-address": "192.0.2.252",
            "prefix-length": 24
          }
        }
      },
      {
        "name": "parent-vlan-100-gw2",
        "description": "This parent represents a bridge with Layer \
                        3 interface (IRB) to connect NFs in VLAN 100",
        "group-profile-ref": [
          "profile-vlan-100"
        ],
        "ip-connection": {
          "ipv4": {
            "local-address": "192.0.2.253",
            "prefix-length": 24
          }
        }
      },
      {
        "name": "ac-ce-01-vlan-100",
        "description": "attachment to CE1 in VLAN 100",
        "parent-ref": [
          "parent-vlan-100-gw1",
          "parent-vlan-100-gw2"
        ],
        "l2-connection": {
          "bearer-reference": "bearer--1"
        }
      },
      {
        "name": "ac-ce-02-vlan-100",
        "description": "attachment to CE2 in VLAN 100",
        "parent-ref": [
          "parent-vlan-100-gw1",
          "parent-vlan-100-gw2"
        ],
        "l2-connection": {
          "bearer-reference": "bearer--2"
        }
      },
      {
        "_comment": "ac-ce-03-vlan-100 to ac-ce-09-vlan-100 are \
                                                          hidden"
      },
      {
        "name": "ac-ce-10-vlan-100",
        "description": "attachment to CE10 in VLAN 100",
        "parent-ref": [
          "parent-vlan-100-gw1",
          "parent-vlan-100-gw2"
        ],
        "l2-connection": {
```

```
          "bearer-reference": "bearer--10"
        }
      }
    ]
  }
}
```

Figure 66: Message Body for the Configuration of CEs with Static Addressing and BFD Protection

Appendix B. Full Tree

```
module: ietf-ac-svc
  +--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 service-provisioning-profiles
  |  +--rw service-profile-identifier* [id]
  |     +--rw id    string
  +--rw attachment-circuits
     +--rw ac-group-profile* [name]
     |  +--rw name                 string
     |  +--rw service-profile*     service-profile-reference
     |  +--rw l2-connection {ac-common:layer2-ac}?
     |  |  +--rw encapsulation
     |  |  |  +--rw type?              identityref
     |  |  |  +--rw dot1q
     |  |  |  |  +--rw tag-type?   identityref
     |  |  |  |  +--rw cvlan-id?   uint16
     |  |  |  +--rw priority-tagged
     |  |  |  |  +--rw tag-type?   identityref
     |  |  |  +--rw qinq
     |  |  |     +--rw tag-type?   identityref
     |  |  |     +--rw svlan-id?   uint16
     |  |  |     +--rw cvlan-id?   uint16
     |  |  +--rw (l2-service)?
     |  |  |  +--:(l2-tunnel-service)
     |  |  |  |  +--rw l2-tunnel-service
     |  |  |  |     +--rw type?         identityref
     |  |  |  |     +--rw pseudowire
     |  |  |  |     |  +--rw vcid?      uint32
     |  |  |  |     |  +--rw far-end?   union
     |  |  |  |     +--rw vpls
     |  |  |  |     |  +--rw vcid?      uint32
     |  |  |  |     |  +--rw far-end*   union
     |  |  |  |     +--rw vxlan
     |  |  |  |        +--rw vni-id?           uint32
```

```
      |  |  |  |        +--rw peer-mode?         identityref
      |  |  |  |        +--rw peer-ip-address*   inet:ip-address
      |  |  |  +--:(l2vpn)
      |  |  |     +--rw l2vpn-id?            vpn-common:vpn-id
      |  |  +--rw bearer-reference?          string
      |  |           {ac-common:server-assigned-reference}?
      |  +--rw ip-connection {ac-common:layer3-ac}?
      |  |  +--rw ipv4 {vpn-common:ipv4}?
      |  |  |  +--rw local-address?
      |  |  |  |       inet:ipv4-address
      |  |  |  +--rw virtual-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
      |  |  |     |  +--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?
      |  |  |                   failure-detection-profile-reference
      |  |  |                   {vpn-common:bfd}?
      |  |  +--rw ipv6 {vpn-common:ipv6}?
      |  |  |  +--rw local-address?
      |  |  |  |       inet:ipv6-address
      |  |  |  +--rw virtual-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
      |  |  |     |  +--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?
      |  |  |                   failure-detection-profile-reference
      |  |  |                   {vpn-common:bfd}?
      |  |  +--rw (l3-service)?
      |  |     +--:(l3-tunnel-service)
      |  |        +--rw l3-tunnel-service
      |  |           +--rw type?   identityref
      |  +--rw routing-protocols
      |  |  +--rw routing-protocol* [id]
      |  |     +--rw id                 string
      |  |     +--rw type?              identityref
      |  |     +--rw routing-profiles* [id]
      |  |     |  +--rw id      routing-profile-reference
      |  |     |  +--rw type?   identityref
      |  |     +--rw static
      |  |     |  +--rw cascaded-lan-prefixes
      |  |     |     +--rw ipv4-lan-prefix* [lan next-hop]
      |  |     |     |       {vpn-common:ipv4}?
      |  |     |     |  +--rw lan
      |  |     |     |  |       inet:ipv4-prefix
      |  |     |     |  +--rw lan-tag?                     string
      |  |     |     |  +--rw next-hop                     union
      |  |     |     |  +--rw metric?                      uint32
      |  |     |     |  +--rw failure-detection-profile?
      |  |     |     |  |       failure-detection-profile-reference
      |  |     |     |  |       {vpn-common:bfd}?
      |  |     |     |  +--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 failure-detection-profile?
      |  |     |        |       failure-detection-profile-reference
      |  |     |        |       {vpn-common:bfd}?
      |  |     |        +--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-as?          inet:as-number
      |  |     |  |     +--rw peer-as?           inet:as-number
      |  |     |  |     +--rw address-family?    identityref
      |  |     |  |     +--rw role?              identityref
      |  |     |  |     +--rw local-address?     inet:ip-address
      |  |     |  |     +--rw bgp-max-prefix
      |  |     |  |     |  +--rw max-prefix?   uint32
      |  |     |  |     +--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* [id]
      |  |     |     +--rw id                            string
      |  |     |     +--ro server-reference?             string
      |  |     |     |       {ac-common:server-assigned-reference}?
      |  |     |     +--rw remote-address?          inet:ip-address
      |  |     |     +--rw local-address?           inet:ip-address
      |  |     |     +--rw local-as?                inet:as-number
      |  |     |     +--rw peer-as?                 inet:as-number
      |  |     |     +--rw address-family?          identityref
      |  |     |     +--rw role?                    identityref
      |  |     |     +--rw bgp-max-prefix
      |  |     |     |  +--rw max-prefix?   uint32
      |  |     |     +--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 requested-start?
      |  |     |     |       yang:date-and-time
      |  |     |     +--rw requested-stop?
      |  |     |     |       yang:date-and-time
      |  |     |     +--ro actual-start?
      |  |     |     |       yang:date-and-time
      |  |     |     +--ro actual-stop?
      |  |     |     |       yang:date-and-time
      |  |     |     +--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 peer-group?
      |  |     |     |       -> ../../peer-groups/peer-group/name
      |  |     |     +--rw failure-detection-profile?
      |  |     |             failure-detection-profile-reference
      |  |     |             {vpn-common:bfd}?
      |  |     +--rw ospf {vpn-common:rtg-ospf}?
      |  |     |  +--rw address-family?   identityref
      |  |     |  +--rw area-id           yang:dotted-quad
      |  |     |  +--rw metric?           uint16
      |  |     |  +--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 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 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 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* [id]
      |  |        +--rw id                 string
      |  |        +--rw local-address?     inet:ip-address
      |  |        +--rw remote-address?    inet:ip-address
      |  |        +--rw profile?
      |  |        |       failure-detection-profile-reference
      |  |        +--rw holdtime?          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 security
|  |  +--rw encryption {vpn-common:encryption}?
|  |  |  +--rw enabled?   boolean
|  |  |  +--rw layer?     enumeration
|  |  +--rw encryption-profile
|  |     +--rw (profile)?
|  |        +--:(provider-profile)
|  |        |  +--rw provider-profile?
|  |        |          encryption-profile-reference
|  |        +--:(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* [profile]
```

```
      |     |        +--rw profile      qos-profile-reference
      |     |        +--rw direction?   identityref
      |     +--rw access-control-list
      |        +--rw acl-profiles
      |           +--rw acl-profile* [profile]
      |              +--rw profile    forwarding-profile-reference
      +--rw placement-constraints
      |  +--rw constraint* [constraint-type]
      |     +--rw constraint-type    identityref
      |     +--rw target
      |        +--rw (target-flavor)?
      |           +--:(id)
      |           |  +--rw group* [group-id]
      |           |     +--rw group-id    string
      |           +--:(all-accesses)
      |           |  +--rw all-other-accesses?   empty
      |           +--:(all-groups)
      |              +--rw all-other-groups?     empty
      +--rw customer-name?           string
      +--rw requested-start?         yang:date-and-time
      +--rw requested-stop?          yang:date-and-time
      +--ro actual-start?            yang:date-and-time
      +--ro actual-stop?             yang:date-and-time
      +--rw ac* [name]
         +--rw customer-name?       string
         +--rw description?         string
         +--rw test-only?           empty
         +--rw requested-start?     yang:date-and-time
         +--rw requested-stop?      yang:date-and-time
         +--ro actual-start?        yang:date-and-time
         +--ro actual-stop?         yang:date-and-time
         +--rw role?                identityref
         +--rw peer-sap-id*         string
         +--rw group-profile-ref*   ac-group-reference
         +--rw parent-ref*
         |         ac-svc:attachment-circuit-reference
         +--ro child-ref*
         |         ac-svc:attachment-circuit-reference
         +--rw group* [group-id]
         |  +--rw group-id      string
         |  +--rw precedence?   identityref
         +--ro service-ref* [service-type service-id]
         |  +--ro service-type    identityref
         |  +--ro service-id      string
         +--ro server-reference?    string
         |       {ac-common:server-assigned-reference}?
         +--rw name                 string
         +--rw service-profile*     service-profile-reference
         +--rw l2-connection {ac-common:layer2-ac}?
         |  +--rw encapsulation
         |  |  +--rw type?              identityref
         |  |  +--rw dot1q
         |  |  |  +--rw tag-type?   identityref
         |  |  |  +--rw cvlan-id?   uint16
         |  |  +--rw priority-tagged
         |  |  |  +--rw tag-type?   identityref
         |  |  +--rw qinq
         |  |     +--rw tag-type?   identityref
```

```
|  |     +--rw svlan-id?   uint16
|  |     +--rw cvlan-id?   uint16
|  +--rw (l2-service)?
|  |  +--:(l2-tunnel-service)
|  |  |  +--rw l2-tunnel-service
|  |  |     +--rw type?         identityref
|  |  |     +--rw pseudowire
|  |  |     |  +--rw vcid?      uint32
|  |  |     |  +--rw far-end?   union
|  |  |     +--rw vpls
|  |  |     |  +--rw vcid?      uint32
|  |  |     |  +--rw far-end*   union
|  |  |     +--rw vxlan
|  |  |        +--rw vni-id?            uint32
|  |  |        +--rw peer-mode?         identityref
|  |  |        +--rw peer-ip-address*   inet:ip-address
|  |  +--:(l2vpn)
|  |     +--rw l2vpn-id?            vpn-common:vpn-id
|  +--rw bearer-reference?          string
|           {ac-common:server-assigned-reference}?
+--rw ip-connection {ac-common:layer3-ac}?
|  +--rw ipv4 {vpn-common:ipv4}?
|  |  +--rw local-address?
|  |  |       inet:ipv4-address
|  |  +--rw virtual-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
|  |     |  +--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?
|  |                   failure-detection-profile-reference
```

```
|  |                    {vpn-common:bfd}?
|  +--rw ipv6 {vpn-common:ipv6}?
|  |  +--rw local-address?
|  |  |       inet:ipv6-address
|  |  +--rw virtual-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
|  |     |  +--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?
|  |                   failure-detection-profile-reference
|  |                   {vpn-common:bfd}?
|  +--rw (l3-service)?
|     +--:(l3-tunnel-service)
|        +--rw l3-tunnel-service
|           +--rw type?   identityref
+--rw routing-protocols
|  +--rw routing-protocol* [id]
|     +--rw id                  string
|     +--rw type?               identityref
|     +--rw routing-profiles* [id]
|     |  +--rw id      routing-profile-reference
|     |  +--rw type?   identityref
|     +--rw static
|     |  +--rw cascaded-lan-prefixes
|     |     +--rw ipv4-lan-prefix* [lan next-hop]
|     |        |       {vpn-common:ipv4}?
|     |        |  +--rw lan
|     |        |  |       inet:ipv4-prefix
|     |        |  +--rw lan-tag?                   string
|     |        |  +--rw next-hop                   union
```

```
|     |     |  +--rw metric?                      uint32
|     |     |  +--rw failure-detection-profile?
|     |     |  |       failure-detection-profile-reference
|     |     |  |       {vpn-common:bfd}?
|     |     |  +--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 failure-detection-profile?
|     |        |       failure-detection-profile-reference
|     |        |       {vpn-common:bfd}?
|     |        +--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-as?         inet:as-number
|     |  |     +--rw peer-as?          inet:as-number
|     |  |     +--rw address-family?   identityref
|     |  |     +--rw role?             identityref
|     |  |     +--rw local-address?    inet:ip-address
|     |  |     +--rw bgp-max-prefix
|     |  |     |  +--rw max-prefix?   uint32
|     |  |     +--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* [id]
|     |     +--rw id                            string
|     |     +--ro server-reference?             string
```

```
          |     |     |       {ac-common:server-assigned-reference}?
          |     |     +--rw remote-address?          inet:ip-address
          |     |     +--rw local-address?           inet:ip-address
          |     |     +--rw local-as?                inet:as-number
          |     |     +--rw peer-as?                 inet:as-number
          |     |     +--rw address-family?          identityref
          |     |     +--rw role?                    identityref
          |     |     +--rw bgp-max-prefix
          |     |     |  +--rw max-prefix?   uint32
          |     |     +--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 requested-start?
          |     |     |       yang:date-and-time
          |     |     +--rw requested-stop?
          |     |     |       yang:date-and-time
          |     |     +--ro actual-start?
          |     |     |       yang:date-and-time
          |     |     +--ro actual-stop?
          |     |     |       yang:date-and-time
          |     |     +--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 peer-group?
          |     |     |       -> ../../peer-groups/peer-group/name
          |     |     +--rw failure-detection-profile?
          |     |             failure-detection-profile-reference
          |     |             {vpn-common:bfd}?
          |     +--rw ospf {vpn-common:rtg-ospf}?
          |     |  +--rw address-family?   identityref
          |     |  +--rw area-id           yang:dotted-quad
          |     |  +--rw metric?           uint16
          |     |  +--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 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 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 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* [id]
         |        +--rw id                string
         |        +--rw local-address?    inet:ip-address
         |        +--rw remote-address?   inet:ip-address
         |        +--rw profile?
         |        |       failure-detection-profile-reference
         |        +--rw holdtime?         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 security
         |  +--rw encryption {vpn-common:encryption}?
         |  |  +--rw enabled?   boolean
         |  |  +--rw layer?     enumeration
         |  +--rw encryption-profile
         |     +--rw (profile)?
         |        +--:(provider-profile)
         |        |  +--rw provider-profile?
         |        |          encryption-profile-reference
         |        +--:(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* [profile]
|        +--rw profile      qos-profile-reference
|        +--rw direction?   identityref
+--rw access-control-list
   +--rw acl-profiles
      +--rw acl-profile* [profile]
         +--rw profile    forwarding-profile-reference
```

Acknowledgments

This document leverages [RFC9182] and [RFC9291]. Thanks to Gyan Mishra for the review.

Thanks to Ebben Aries for the YANG Doctors review and for providing [Instance-Data].

Thanks to Donald Eastlake for the careful RTGDIR review, Tero Kivinen for the SECDIR review, Gyan Mishra for the GENART review, and Adrian Farrel for the OPSDIR review.

Thanks to Luis Miguel Contreras Murillo for the careful shepherd review.

Thanks to Mahesh Jethanandani for the AD review.

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

Contributors

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

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

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

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

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

Authors' Addresses

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

Richard Roberts (EDITOR)
Juniper
Email: rroberts@juniper.net

Oscar Gonzalez de Dios
Telefonica
Email: oscar.gonzalezdedios@telefonica.com

Samier Barguil
Nokia
Email: samier.barguil_giraldo@nokia.com

Bo Wu
Huawei Technologies
Email: lana.wubo@huawei.com