Workgroup: OPSAWG
Internet-Draft:
draft-ietf-opsawg-teas-attachment-circuit-05
Published: 22 January 2024
Intended Status: Standards Track
Expires: 25 July 2024
Authors: M. Boucadair, Ed. (Orange)
R. Roberts, Ed. (Juniper)
O. G. D. Dios (Telefonica)
S. B. Giraldo (Nokia)
B. Wu (Huawei Technologies)

YANG Data Models for Bearers and 'Attachment Circuits'-as-a-Service (ACaaS)

Abstract

This document specifies a YANG service data model for Attachment Circuits (ACs). This model can be used for the provisioning of ACs before or during service provisioning (e.g., Network Slice Service). The document also specifies a service model for managing bearers over which ACs are established.

Also, the document specifies a set of reusable groupings. Whether other service models reuse structures defined in the AC models or simply include an AC reference is a design choice of these service models. Utilizing the AC service model to manage ACs over which a service is delivered has the advantage of decoupling service management from upgrading AC components to incorporate recent AC technologies or features.

Discussion Venues

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

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

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

Status of This Memo

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

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

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

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

This Internet-Draft will expire on 25 July 2024.

Table of Contents

1. Introduction

1.1. Scope and Intended Use

Connectivity services are provided by networks to customers via dedicated terminating points, such as Service Functions [RFC7665], customer edges (CEs), peer Autonomous System Border Routers (ASBRs), data centers gateways, or Internet Exchange Points. A connectivity service is basically about ensuring data transfer received from or destined to a given terminating point to or from other terminating points within the same customer/service, an interconnection node, or an ancillary node. 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 terminating points and a provider network, allowing successfully data exchanged over these links. The required setup is referred to in this document as Attachment Circuits (ACs), while the underlying link is referred to as "bearers".

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

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

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

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

Also, because the instantiation of an attachment circuit 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 will greatly simplify the provisioning of value-added services** delivered over an attachment circuits. For example, management systems of adjacent domains that need to connect via an AC will use such means to agree on 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 attachment circuits that are exposed by a network to its customers, such as an enterprise site, a network function, a hosting infrastructure, or a peer network provider. The model can be used for the provisioning of ACs prior or during advanced service provisioning (e.g., Network Slice Service).

The "ietf-ac-svc" module (Section 5.2) includes a set of reusable groupings. Whether a service model 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 to decorrelate the management of the (core) service vs. upgrade the AC components to reflect recent AC technologies or new features (e.g., new encryption scheme, additional routing protocol). **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 Attachment Circuit.**

Since the provisioning of an AC requires a bearer to be in place, this document introduces a new module called "ietf-bearer-svc" that enables customers to manage their bearer requests (Section 5.1). The customers can then retrieve a provider-assigned bearer reference that they will include in their AC service requests. An example to retrieve 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 SAPs. Both schemes are supported in the AC service model.

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 Service Attachment Points (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.

The AC service model **does not make any assumptions about the internal structure or even the nature or the services that will be delivered over an attachment circuit or a set of attachment circuits**. Customers do not have access to that network view other than the ACs that the ordered. For example, the AC service model can be used to provision a set of ACs to connect multiple sites (Site1, Site2, ..., SiteX) for customer who also requested VPN services. If these provisioning of these services require 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 attachment circuit for a known peer SAP (Appendix A.3).

*Instantiate multiple attachment circuits over the same bearer (Appendix A.4).

*Control the precedence over multiple attachment circuits (Appendix A.5).

*Create Multiple ACs bound to Multiple CEs (Appendix A.6).

*Bind a slice service to a set of pre-provisioned attachment circuits (Appendix A.7).

*Connect a Cloud Infrastructure to a service provider network (Appendix A.8). Note that the AC model can be used between service providers for other interconnection purposes.

The examples provided in Appendix A use the IPv4 address blocks reserved for documentation [RFC5737], the IPv6 prefix reserved for documentation [RFC3849], and the Autonomous System (AS) numbers reserved for documentation [RFC5398].

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

1.2. Position 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 augmented SAP AC network model [I-D.ietf-opsawg-ntw-attachment-circuit].

The AC service model **is not a device model**. A network provider may use a variety of device models (e.g., Routing management [RFC8349] or BGP [I-D.ietf-idr-bgp-model]) to provision an AC service in relevant network nodes.

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

The 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 Using the L3SM as Reference Data Model for ACaaS?

Like the L2SM, the 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 [RFC8340].

This document uses the following terms:

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

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

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

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

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

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

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

3. Sample Uses of the Data Models

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

Figure 1 depicts two target topology flavors that involve ACs. These topologies have the following characteristics:

* A Customer Edges (CEs) can be either a physical device or a logical entity. Such logical entity is typically a software component (e.g., a virtual service function that is hosted within the provider's network or a third-party infrastructure). A CE is seen by the network as a peer SAP.

*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 service functions [RFC7665]).

*A network provider may bind a single AC to one or multiple peer SAPs (e.g., CE#1 and CE#2 are tagged as peer SAPs for the same AC). For example, and as discussed in [RFC4364], multiple CEs can be attached to a PE over the same attachment circuit. This scenario is typically implemented when the Layer 2 infrastructure between the CE and the network is a multipoint service.

*A single CE may terminate multiple ACs, which can be associated with the same bearer or distinct bearers.

*Customers may request protection schemes in which the ACs associated with their endpoints are terminated by the same PE (e.g., CE#3), distinct PEs (e.g., CE#34), etc. The network provider uses this request to decide where to terminate the AC in the network provider network and also whether to enable specific capabilities (e.g., Virtual Router Redundancy Protocol (VRRP)).

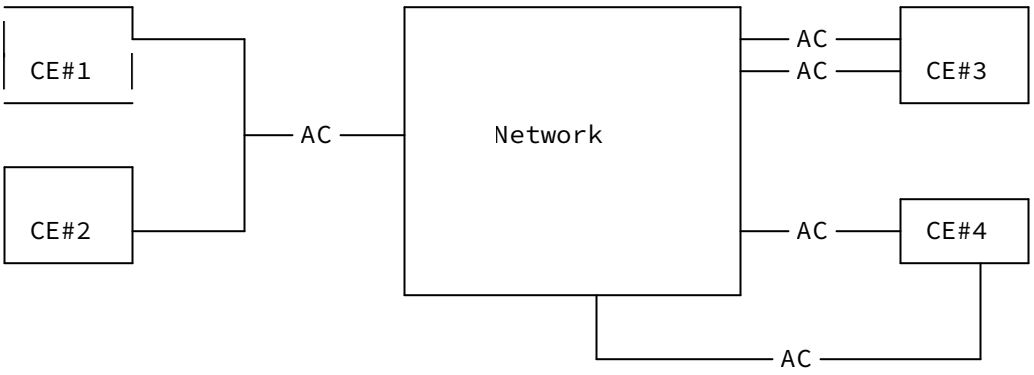


Figure 1: Examples of ACs

3.2. 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 flow put in place for the provisioning of advanced network services and how they are bound to an attachment circuit. For example, a single attachment circuit 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 attachment circuit to be put in place, and then refer to that bearer or AC when requesting services that are bound to the bearer or AC.

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

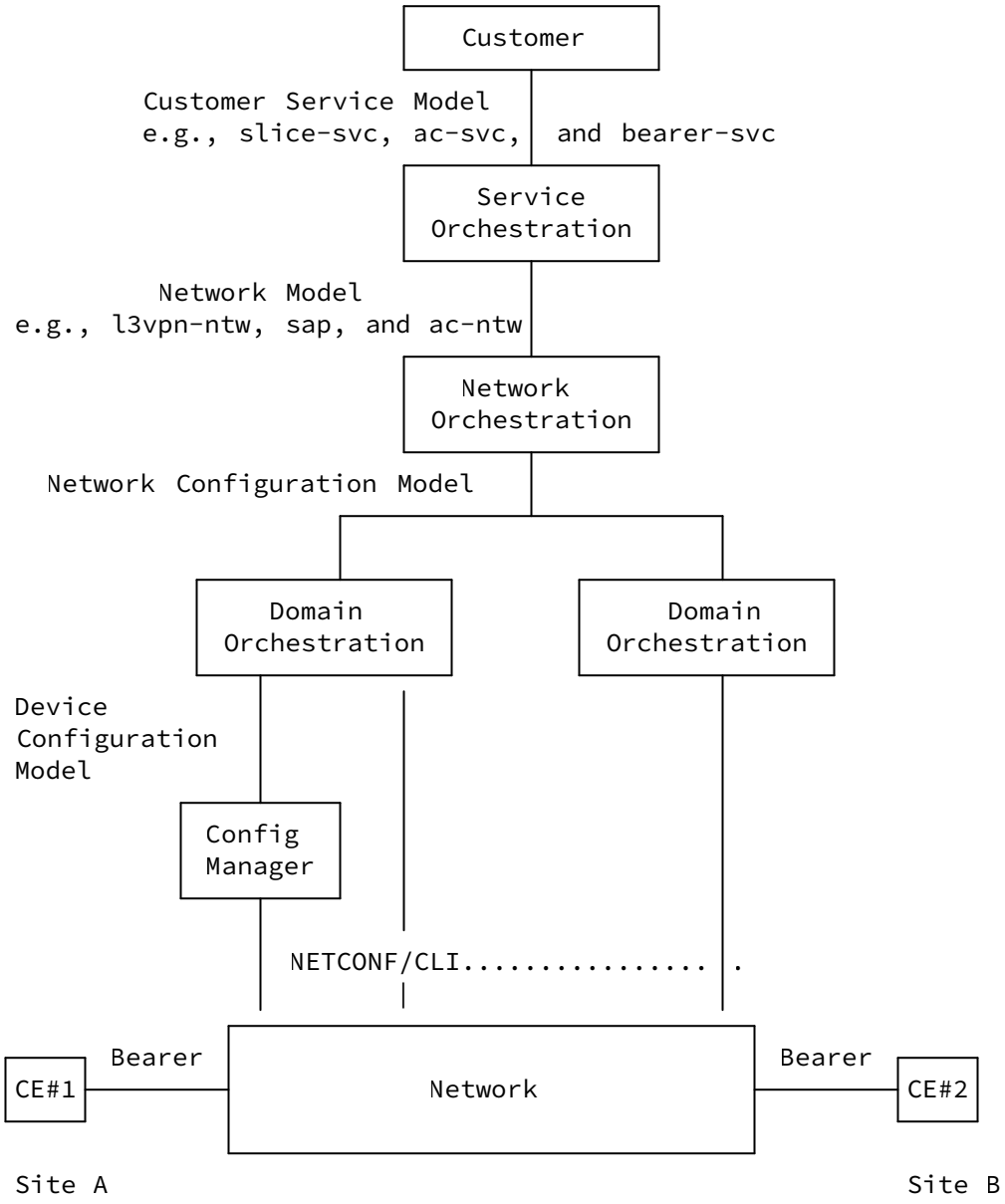


Figure 2: An Example of AC Model Usage

In order to ease the mapping between the service model and underlying network models (e.g., L3NM, 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].

4. Description of the Data Models

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

Figure 3 shows the tree for managing the bearers (that is, the properties of an attachment that are below Layer 3). 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 a Service Function).

```
+--rw bearers
   +--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* [id]
      +--rw id                  string
      +--rw description?        string
      +--rw groups
      |  +--rw group* [group-id]
      |     +--rw group-id    string
      +--rw op-comment?         string
      +--rw customer-point
      |  +--rw identified-by?   identityref
      |  +--rw device
      |  |  +--rw device-id?   string
      |  |  +--rw location
      |  |     +--rw location-name?   string
      |  |     +--rw address?         string
      |  |     +--rw postal-code?     string
      |  |     +--rw state?           string
      |  |     +--rw city?            string
      |  |     +--rw country-code?    string
      |  +--rw site
      |  |  +--rw site-id?    string
      |  |  +--rw location
      |  |     +--rw location-name?   string
      |  |     +--rw address?         string
      |  |     +--rw postal-code?     string
      |  |     +--rw state?           string
      |  |     +--rw city?            string
      |  |     +--rw country-code?    string
      |  +--rw custom-id?       string
      +--rw requested-type?     identityref
      +--rw test-only?          empty
      +--ro bearer-reference?   string
      |       {vpn-common:bearer-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 3: Bearer Service Tree Structure

The same customer site (CE, NF, etc.) can terminate one or multiple bearers; each of them 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., PoP/PE selection).

Auto-discovery using network protocols: Devices can use specific protocols (e.g., Link Layer Discovery Protocol (LLDP)) 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 in the future to accommodate specific deployment contexts.

The descriptions of the bearer data nodes are as follows:

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

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

'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.

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

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

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

'test-only': Indicates that a request is only for test and not for setting, 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 which do not natively support 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 'id' is deployment-specific. The module does not assume nor preclude such schemes.

'ac-svc-ref': Specifies the set of attachment circuits 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 actually was enabled.

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

'status': Used to track the overall status of a given bearer. Both 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': The "oper-status" of a bearer reflects its operational state 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., whether "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. If these status values differ, a trigger to detect an anomaly.

See [RFC9181] for more details.

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

The full tree diagram of the module can be generated using the "pyang" tool [PYANG]. That tree is not included here because it is too long (Section 3.3 of [RFC8340]). Instead, subtrees are provided for the reader's convenience.

4.2.1. Overall Structure

The overall tree structure of the AC service module is shown in Figure 4.

```
+--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
      |  ...
      +--rw ip-connection
      |  ...
      +--rw routing-protocols
      |  ...
      +--rw oam
      |  ...
      +--rw security
      |  ...
      +--rw service
         ...
```

Figure 4: Overall AC Service Tree Structure

The rationale for deciding whether a reusable grouping should be maintained in this document or be moved into the AC common module [I-D.ietf-opsawg-teas-common-ac] is as follows:

- Groupings that are reusable among the AC service module, AC network module, 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 [I-D.ietf-opsawg-teas-common-ac]. Therefore, the description of these nodes are not reiterated in the following subsections.

4.2.2. Service Profiles

4.2.2.1. Description

The 'specific-provisioning-profiles' container (Figure 5) can be used by a service provider to maintain a set of reusable profiles. The profiles definition are similar to those defined in [RFC9181], including: Quality of Service (QoS), Bidirectional Forwarding Detection (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 bfd-profile-identifier* [id]
  |     |  +--rw id    string
  |     +--rw forwarding-profile-identifier* [id]
  |     |  +--rw id    string
  |     +--rw routing-profile-identifier* [id]
  |        +--rw id    string
  +--rw service-provisioning-profiles
  |  +--rw service-profile-identifier* [id]
  |     +--rw id    string
  +--rw attachment-circuits
     +--rw ac-group-profile* [name]
     |  ...
     +--rw placement-constraints
     |  ...
     +--rw ac* [name]
        ...
        +--rw l2-connection
        |  ...
        +--rw ip-connection
        |  ...
        +--rw routing-protocols
        |  ...
        +--rw oam
        |  ...
        +--rw security
        |  ...
        +--rw service
           ...
```

Figure 5: Service Profiles

As shown in Figure 5, 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:

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

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

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

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

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

4.2.2.2. Referencing Service/Specific Profiles

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

4.2.3. 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'. 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 7).

4.2.4. AC Placement Contraints

The 'placement-constraints' specifies the placement constraints of an AC. For example, this container can be used to request avoiding to 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 6.

```
+--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 6: Placement Constraints Subtree Structure

4.2.5. Attachment Circuits

The structure of 'attachment-circuits' 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 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 peer-sap-id*         string
      +--rw ac-group-profile*    ac-group-reference
      +--rw ac-parent-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
      +--rw name                 string
      +--rw service-profile*     service-profile-reference
      +--rw l2-connection
      |  ...
      +--rw ip-connection
      |  ...
      +--rw routing-protocols
      |  ...
      +--rw oam
      |  ...
      +--rw security
      |  ...
      +--rw service
         ...
```

Figure 7: Attachment Circuits Tree Structure

The description of the data nodes is as follows:

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

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

'test-only':
Indicates that a request is only for test and not for setting, 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 which do not natively support such option.

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

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

'actual-start': Reports the actual date and time when the attachment circuit actually was enabled.

'actual-stop': Reports the actual date and time when the attachment circuit actually was disabled.

'peer-sap-id': Includes references to the remote endpoints of an attachment circuit [RFC9408].

'ac-group-profile': Indicates references to one or more profiles that are defined in Section 4.2.3.

'ac-parent-ref': Specifies an AC that is inherited by this attachment circuit.

In contexts where dynamic terminating points are managed for a given AC, a parent AC can be defined with the 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.

Whenever a parent AC is deleted, that all child ACs of that AC **MUST** be deleted.

'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 attachment circuit. The services are indexed by their type.

'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 4.2.5.1.

'ip-connection': See Section 4.2.5.2.

'routing': See Section 4.2.5.3.

'oam': See Section 4.2.5.7.

'security': See Section 4.2.5.8.

'service': See Section 4.2.5.9.

4.2.5.1. Layer 2 Connection Structure

The 'l2-connection' container (Figure 8) 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 [I-D.ietf-opsawg-teas-common-ac].

```
+--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
      |  +--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
      |  ...
      +--rw routing-protocols
      |  ...
      +--rw oam
      |  ...
      +--rw security
```

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

Figure 8: Layer 2 Connection Tree Structure

4.2.5.2. 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 [I-D.ietf-opsawg-teas-common-ac]. Both IPv4 and IPv6 parameters are supported.

Figure 9 shows the structure of the IPv4 connection.

```
| ...
+--rw ip-connection
|  +--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 ipv6 {vpn-common:ipv6}?
|     ...
```

Figure 9: Layer 3 Connection Tree Structure (IPv4)

Figure 10 shows the structure of the IPv6 connection.

```
| ...
+--rw ip-connection
|  +--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
   ...
```

Figure 10: Layer 3 Connection Tree Structure (IPv6)

4.2.5.3. Routing

As shown in the tree depicted in Figure 11, 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 be then enabled between a PE and the customer terminating 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 4.2.5.3.1), the module supports BGP (Section 4.2.5.3.2), OSPF (Section 4.2.5.3.3), IS-IS (Section 4.2.5.4), and RIP (Section 4.2.5.5). It also includes a reference to the 'routing-profile-identifier' defined in Section 4.2.2, so that additional constraints can be applied to a specific instance of each routing protocol.

```
+--rw specific-provisioning-profiles
|  ...
+--rw service-provisioning-profiles
|  ...
+--rw attachment-circuits
   +--rw ac-group-profile* [name]
   |  ...
   +--rw placement-constraints
   |  ...
   +--rw ac* [name]
      +--rw customer-name?       string
      +--rw description?         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 peer-sap-id*         string
      +--rw ac-group-profile*    ac-group-reference
      +--rw group* [group-id]
      |  +--rw group-id      string
      |  +--rw precedence?   identityref
      +--rw name                 string
      +--rw l2-connection
      | ...
      +--rw ip-connection
      |  ...
      +--rw routing-protocols
      |  +--rw routing-protocol* [id]
      |     +--rw id                  string
      |     +--rw type?               identityref
      |     +--rw routing-profiles* [id]
      |     |  +--rw id      routing-profile-reference
      |     |  +--rw type?   identityref
      |     +--rw static
      |     |  ...
      |     +--rw bgp
      |     |  ...
      |     |  ...
      |     +--rw isis
      |     |  ...
      |     +--rw rip
      |     |  ...
      |     +--rw vrrp
      |        ...
      +--rw oam
      |  ...
      +--rw security
      |  ...
```

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

Figure 11: Routing Tree Structure

4.2.5.3.1. Static Routing

The static tree structure is shown in Figure 12.

```
|  ...
+--rw routing-protocols
|  +--rw routing-protocol* [id]
|     +--rw id                  string
|     +--rw type?               identityref
|     +--rw routing-profiles* [id]
|     |  +--rw id      routing-profile-reference
|     |  +--rw type?   identityref
|     +--rw static
|     |  +--rw cascaded-lan-prefixes
|     |     +--rw ipv4-lan-prefixes* [lan next-hop]
|     |     |       {vpn-common:ipv4}?
|     |     |  +--rw lan         inet:ipv4-prefix
|     |     |  +--rw lan-tag?    string
|     |     |  +--rw next-hop    union
|     |     |  +--rw metric?     uint32
|     |     |  +--rw status
|     |     |     +--rw admin-status
|     |     |     |  +--rw status?        identityref
|     |     |     |  +--ro last-change?   yang:date-and-time
|     |     |     +--ro oper-status
|     |     |        +--ro status?        identityref
|     |     |        +--ro last-change?   yang:date-and-time
|     |     +--rw ipv6-lan-prefixes* [lan next-hop]
|     |             {vpn-common:ipv6}?
|     |        +--rw lan         inet:ipv6-prefix
|     |        +--rw lan-tag?    string
|     |        +--rw next-hop    union
|     |        +--rw metric?     uint32
|     |        +--rw status
|     |           +--rw admin-status
|     |           |  +--rw status?        identityref
|     |           |  +--ro last-change?   yang:date-and-time
|     |           +--ro oper-status
|     |              +--ro status?        identityref
|     |              +--ro last-change?   yang:date-and-time
|     +--rw bgp
|     |  ...
|     +--rw ospf
|     |  ...
|     +--rw isis
|     |  ...
|     +--rw rip
|     |  ...
|     +--rw vrrp
|        ...
```

Figure 12: Static Routing Tree Structure

As depicted in Figure 12, the following data nodes can be defined for a given IP prefix:

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

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

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

'bfd-enable': Indicates whether BFD is enabled or disabled for this static route entry.

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

'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.

4.2.5.3.2. BGP

The BGP tree structure is shown in Figure 13.

```
|  ...
+--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
|     |  +--rw peer-groups
|     |  |  +--rw peer-group* [name]
|     |  |     +--rw name              string
|     |  |     +--ro local-as?         inet:as-number
|     |  |     +--rw peer-as?          inet:as-number
|     |  |     +--rw address-family?   identityref
|     |  |     +--ro local-address?    inet:ip-address
|     |  |     +--rw authentication
|     |  |        +--rw enable?            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
|     |     +--rw remote-address?   inet:ip-address
|     |     +--ro local-address?    inet:ip-address
|     |     +--rw peer-group?
|     |     |       -> ../../peer-groups/peer-group/name
|     |     +--ro local-as?         inet:as-number
|     |     +--rw peer-as?          inet:as-number
|     |     +--rw address-family?   identityref
|     |     +--rw authentication
|     |     |  +--rw enable?            boolean
|     |     |  +--rw keying-material
|     |     |     +--rw (option)?
|     |     |        +--:(ao)
|     |     |        |  +--rw enable-ao?          boolean
```

```
|     |     |        |  +--rw ao-keychain?
|     |     |        |          key-chain:key-chain-ref
|     |     |        +--:(md5)
|     |     |        |  +--rw md5-keychain?
|     |     |        |          key-chain:key-chain-ref
|     |     |        +--:(explicit)
|     |     |           +--rw key-id?             uint32
|     |     |           +--rw key?                string
|     |     |           +--rw crypto-algorithm?   identityref
|     |     +--rw status
|     |        +--rw admin-status
|     |        |  +--rw status?        identityref
|     |        |  +--ro last-change?   yang:date-and-time
|     |        +--ro oper-status
|     |           +--ro status?        identityref
|     |           +--ro last-change?   yang:date-and-time
|     +--rw ospf
|     |  ...
|     +--rw isis
|     |  ...
|     +--rw rip
|     |  ...
|     +--rw vrrp
|        ...
```

Figure 13: BGP Tree Structure

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

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

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

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

'address-family': Indicates the address family of the peer. It can be set to 'ipv4', 'ipv6', or 'dual-stack'. This address family will be used together with the 'vpn-type' 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]).

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

'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. In addition, the module includes a provision for using IPsec.

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 [RFC8177], mainly SendID and RecvID (Section 3.1 of [RFC5925]).

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

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

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

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

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

4.2.5.3.3. OSPF

The OSPF tree structure is shown in Figure 14.

```
|  ...
+--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
|     |  ...
|     +--rw ospf
|     |  +--rw address-family?   identityref
|     |  +--rw area-id           yang:dotted-quad
|     |  +--rw metric?           uint16
|     |  +--rw authentication
|     |  |  +--rw enable?            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
|     |  ...
|     +--rw rip
|     |  ...
|     +--rw vrrp
|        ...
```

Figure 14: 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 following options are supported: IPsec for OSPFv3 authentication [RFC4552], and the Authentication Trailer for OSPFv2 [RFC5709][RFC7474] and OSPFv3 [RFC7166].

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

4.2.5.4. IS-IS

The IS-IS 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 bgp
|     |  ...
|     +--rw ospf
|     |  ...
|     +--rw isis
|     |  +--rw address-family?   identityref
|     |  +--rw area-address      area-address
|     |  +--rw authentication
|     |  |  +--rw enable?            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
|     |  ...
|     +--rw vrrp
|      ...
```

Figure 15: 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.

4.2.5.5. RIP

The RIP 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
|     |  ...
|     +--rw ospf
|     |  ...
|     +--rw isis
|     |  ...
|     +--rw rip
|     |  +--rw address-family?   identityref
|     |  +--rw authentication
|     |  |  +--rw enable?            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
|      ...
```

Figure 16: 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), or both are to be enabled [RFC2080].

4.2.5.6. VRRP

The model supports the Virtual Router Redundancy Protocol (VRRP) [RFC5798] on an AC (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
|     |  ...
|     +--rw ospf
|     |  ...
|     +--rw isis
|     |  ...
|     +--rw rip
|     |  ...
|     +--rw 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 17: 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 [RFC5798] 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 [RFC5798]).

4.2.5.7. OAM

As shown in the tree depicted in Figure 18, 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
            |  ...
            +--rw ip-connection
            |  ...
            +--rw routing-protocols
            |  ...
            +--rw oam
            |  +--rw bfd {vpn-common:bfd}?
            |     +--rw profile?    bfd-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 18: OAM Tree Structure

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

'profile': Refers to a BFD profile.

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

'status': Indicates the status of the BFD over an AC.

4.2.5.8. Security

As shown in the tree depicted in Figure 19, 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
      |  ...
      +--rw ip-connection
      |  ...
      +--rw routing-protocols
      |  ...
      +--rw oam
      |  ...
      +--rw security
      |  +--rw encryption {vpn-common:encryption}?
      |  |  +--rw enabled?   boolean
      |  |  +--rw layer?     enumeration
      |  +--rw encryption-profile
      |     +--rw (profile)?
      |        +--:(provider-profile)
      |        |  +--rw provider-profile?
      |        |          encryption-profile-reference
      |        +--:(customer-profile)
      |           +--rw customer-key-chain?
      |                   key-chain:key-chain-ref
      +--rw service
         ...
```

Figure 19: Security Tree Structure

The 'security' container specifies the authentication and the encryption to be applied to traffic for a given AC. Tthe 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.

4.2.5.9. Service

The structure of the 'service' container is depicted in Figure 20.

```
+--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
      |  ...
      +--rw ip-connection
      |  ...
      +--rw routing-protocols
      |  ...
      +--rw oam
      |  ...
      +--rw security
      |  ...
      +--rw service
         +--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 20: 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 [I-D.ietf-opsawg-teas-common-ac].

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

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

5. YANG Modules

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

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

```
<CODE BEGINS> file "ietf-bearer-svc@2023-11-13.yang"
module ietf-bearer-svc {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-bearer-svc";
  prefix bearer-svc;

  import ietf-vpn-common {
    prefix vpn-common;
    reference
      "RFC 9181: A Common YANG Data Model for Layer 2 and Layer 3
                 VPNs";
  }
  import ietf-ac-common {
    prefix ac-common;
    reference
      "RFC CCCC: A Common YANG Data Model for Attachment Circuits";
  }
  import ietf-ac-svc {
    prefix ac-svc;
    reference
      "RFC XXXX: YANG Service Data Models for Attachment Circuits";
  }

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

     Editor:   Mohamed Boucadair
               <mailto:mohamed.boucadair@orange.com>
     Author:   Richard Roberts
               <mailto:rroberts@juniper.net>
     Author:   Oscar Gonzalez de Dios
               <mailto:oscar.gonzalezdedios@telefonica.com>
     Author:   Samier Barguil
               <mailto:ssamier.barguil_giraldo@nokia.com>
     Author:   Bo Wu
               <mailto:lana.wubo@huawei.com>";
  description
    "This YANG module defines a generic YANG model for exposing
     network bearers as a service.

     Copyright (c) 2024 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 2023-11-13 {
  description
    "Initial revision.";
  reference
    "RFC xxxx: A YANG Service Data Model for Attachment Circuits";
}

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

grouping location-information {
  description
    "Basic location information";
  container location {
    description
      "Location of the node.";
    leaf location-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 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 city {
```

```
      type string;
      description
        "City of the device/site.";
    }
    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 bearers {
  description
    "Main container for the bearers.";

  container placement-constraints {
    description
      "Diversity constraint type.";
    uses placement-constraints;
  }

  list bearer {
    key "id";
    description
      "Maintains a list of bearers.";
    leaf id {
      type string;
      description
        "An identifier of the bearer.";
    }
    leaf description {
```

```
  type string;
  description
    "A description of this bearer.";
}
uses vpn-common:vpn-components-group;
leaf op-comment {
  type string;
  description
    "Includes comments that can be shared with operational
     teams and which may be useful for the activation of a
     bearer. This may include, for example, information
     about the building, level, etc.";
}
container customer-point {
  description
    "Base container to link the Bearer existence";
  leaf identified-by {
    type identityref {
      base identification-type;
    }
    description
      "Attribute used to identify the bearer";
  }
  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.";
    }
    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.";
    }
    uses location-information;
  }
  leaf custom-id {
    when "derived-from-or-self(../identified-by, "
       + "'bearer-svc:custom')" {
      description
        "Only enabled id identified-by is custom.";
    }
    type string;
    description
      "The semantic of this identifier is shared between the
        customer/provider using out-of-band means.";
  }
}
leaf requested-type {
  type identityref {
    base bearer-type;
  }
  description
    "Type of the requested 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 commited for such bearer
    requests.";
}
leaf bearer-reference {
  if-feature "vpn-common:bearer-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 ACes that are bound to the bearer.";
      }
      uses ac-common:op-instructions;
      uses ac-common:service-status;
    }
  }
}
<CODE ENDS>
```

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

This module uses types defined in [RFC6991], [RFC9181], [RFC8177], and [I-D.ietf-opsawg-teas-common-ac].

```
<CODE BEGINS> file "ietf-ac-svc@2023-11-13.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 CCCC: 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>
     Author:   Richard Roberts
               <mailto:rroberts@juniper.net>
     Author:   Oscar Gonzalez de Dios
               <mailto:oscar.gonzalezdedios@telefonica.com>
     Author:   Samier Barguil
               <mailto:ssamier.barguil_giraldo@nokia.com>
     Author:   Bo Wu
               <mailto:lana.wubo@huawei.com>";
```

```
description
  "This YANG module defines a YANG model for exposing
   attachment circuits as a service (ACaaS).

   Copyright (c) 2024 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 XXXX; see the
   RFC itself for full legal notices.";

revision 2023-11-13 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: YANG Service Data Models for Attachment Circuits";
}

/* 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 attachment circuit 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 attachment circuit 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 type to an encryption profile for referencing
     purposes.";
}

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 type to a QoS profile for referencing purposes.";
}

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

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 type to a forwarding profile for referencing
     purposes.";
}

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 type to a routing profile for referencing
```

```
     purposes.";
}

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

/******************** 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 "vpn-common:bearer-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;
  }
  container ipv6 {
    if-feature "vpn-common:ipv6";
    description
      "IPv6-specific parameters.";
    uses ac-common:ipv6-connection;
  }
}

// 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.";
    }
  }
}

//  BGP Service

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 local IP address that will be used to establish
           the BGP session.";
      }
      uses ac-common:bgp-authentication;
    }
  }
  list neighbor {
    key "id";
    description
      "List of BGP neighbors.";
    leaf id {
      type string;
      description
        "A neighbor identifier.";
    }
    leaf remote-address {
      type inet:ip-address;
      description
        "The remote IP address of this entry's BGP peer.

         If this leaf is not present, this means that the primary
         customer IP address is used as remote IP address.";
    }
    leaf local-address {
      type inet:ip-address;
```

```
      description
        "The local IP address that will be used to establish
         the BGP session.";
    }
    leaf peer-group {
      type leafref {
        path "../../peer-groups/peer-group/name";
      }
      description
        "The peer-group with which this neighbor is associated.";
    }
    uses ac-common:bgp-peer-group-without-name;
    uses ac-common:bgp-authentication;
    uses ac-common:service-status;
  }
}

//  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 5798: Virtual Router Redundancy Protocol (VRRP)
               Version 3 for IPv4 and IPv6";
  leaf address-family {
    type identityref {
      base vpn-common:address-family;
    }
    description
      "Indicates whether IPv4, IPv6, or both
       address families are to be enabled.";
  }
  uses 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.";
      }
      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.";
  }
  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.";
  }
  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.";
  }
  description
    "Configuration specific to RIP routing.";
  leaf address-family {
    type identityref {
      base vpn-common:address-family;
    }
    description
      "Indicates whether IPv4, IPv6, or both
       address families are to be activated.";
  }
```

```
    }
    container vrrp {
      when "derived-from-or-self(../type, "
         + "'vpn-common:vrrp-routing')" {
        description
          "Only applies when the protocol is the
           Virtual Router Redundancy Protocol (VRRP).";
      }
      description
        "Configuration specific to VRRP.";
      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 static routing
           protocol.";
      }
      description
        "Configuration specific to static routing.";
      container cascaded-lan-prefixes {
        description
          "LAN prefixes from the customer.";
```

```
    uses ac-common:ipv4-static-rtg;
    uses ac-common:ipv6-static-rtg;
  }
}
container bgp {
  when "derived-from-or-self(../type, "
     + "'vpn-common:bgp-routing')" {
    description
      "Only applies when the protocol is BGP.";
  }
  description
    "Configuration specific to BGP.";
  uses bgp-svc {
    refine "peer-groups/peer-group/local-address" {
      config false;
    }
    refine "neighbor/local-address" {
      config false;
    }
  }
}
container ospf {
  when "derived-from-or-self(../type, "
     + "'vpn-common:ospf-routing')" {
    description
      "Only applies when the protocol is OSPF.";
  }
  description
    "Configuration specific to OSPF.";
  uses ospf-svc;
}
container isis {
  when "derived-from-or-self(../type, "
     + "'vpn-common:isis-routing')" {
    description
      "Only applies when the protocol is IS-IS.";
  }
  description
    "Configuration specific to IS-IS.";
  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.";
  }
```

```
      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).";
      }
      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;
  }
}

// Bandwith 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 attachment circuit.";
  leaf id {
    type string;
    description
      "An identifier of the AC.";
  }
  container l2-connection {
    description
      "Defines Layer 2 protocols and parameters that are required
       to enable AC connectivity.";
    uses l2-connection-basic;
  }
  container ip-connection {
    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 bandwith parameters.";
    leaf mtu {
      type uint32;
      units "bytes";
      description
        "Layer 2 MTU.";
    }
    uses bandwidth;
  }
}


// Full AC parameters

grouping ac {
  description
    "Grouping for an attachment circuit.";
  leaf name {
    type string;
    description
      "A name of the AC. Data models that need to reference
       an attachment circuit should use
       attachment-circuit-reference.";
  }
  leaf-list service-profile {
    type service-profile-reference;
    description
      "A reference to a service profile.";
  }
  container l2-connection {
    description
      "Defines Layer 2 protocols and parameters that are required
       to enable AC connectivity.";
    uses l2-connection;
  }
  container ip-connection {
    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.";
    leaf profile {
      type bfd-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 bandwith parameters.";
  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.";
          }
        }
      }
    }
  }
}

/******************** 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 attachment circuits.";
    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;
    }
    list ac {
      key "name";
      description
        "Global provisioning of attachment circuits.";
      leaf customer-name {
        type string;
        description
          "Indicates the name of the customer that requested this
           AC.";
      }
      leaf description {
        type string;
        description
          "Associates a description with an AC.";
      }
      leaf test-only {
        type empty;
        description
         "When present, this indicates that this is a feasibility
          check request. No resources are commited for such AC
          requests.";
      }
      uses ac-common:op-instructions;
      leaf-list peer-sap-id {
        type string;
        description
          "One or more peer SAPs can be indicated.";
      }
      leaf-list ac-group-profile {
```

```
  type ac-group-reference;
  description
    "A reference to an AC profile.";
}
leaf ac-parent-ref {
  type ac-svc:attachment-circuit-reference;
  description
    "Specifies the parent AC that is inherited by an AC.
     In contexts where dynamic terminating points are
     bound to the same AC, a parent AC with stable
     inforamtion is created with a set of child AC
     that trackes dynamic informaiton.";
}
list group {
  key "group-id";
  description
    "List of group-ids.";
  leaf group-id {
    type string;
    description
      "Indicates the group-id to which the network access
       belongs.";
  }
  leaf precedence {
    type identityref {
      base ac-common:precedence-type;
    }
    description
      "Defines redundancy of an AC.";
  }
}
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, 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.";
        }
      }
      uses ac;
    }
  }
}
<CODE ENDS>
```

6. Security Considerations

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

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

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

These are the subtrees and data nodes and their sensitivity/ vulnerability in the "ietf-ac-svc" module:

'specific-provisioning-profiles': This container includes a set of sensitive data that influence 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 data nodes are defined with "nacm:default-deny-write" tagging [I-D.ietf-opsawg-teas-common-ac].

'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.

These data nodes are 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. These are the subtrees and data nodes and their sensitivity/vulnerability in the "ietf-bearer-svc" module:

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

These are the subtrees and data nodes and their sensitivity/ vulnerability 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.

Several data nodes ('bgp', 'ospf', 'isis', and 'rip') rely upon [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]).

7. IANA Considerations

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

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

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

8. References

8.1. Normative References

[I-D.ietf-opsawg-teas-common-ac] Boucadair, M., Roberts, R., de Dios, O. G., Barguil, S., and B. Wu, "A Common YANG Data Model for Attachment Circuits", Work in Progress, Internet-Draft, draft-ietf-opsawg-teas-common-ac-03, 14 January 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-teas-common-ac-03>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/

RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

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

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

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

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

[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/rfc/rfc5709>.

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

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

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

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

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

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

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

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

[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/rfc/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/rfc/rfc7474>.

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

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

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

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

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

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

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

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

[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/rfc/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/rfc/rfc9408>.

8.2. Informative References

[I-D.ietf-idr-bgp-model] 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-17, 5 July 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-bgp-model-17>.

[I-D.ietf-opsawg-ntw-attachment-circuit] Boucadair, M., Roberts, R., de Dios, O. G., Barguil, S., and B. Wu, "A Network YANG Data Model for Attachment Circuits", Work in Progress, Internet-Draft, draft-ietf-opsawg-ntw-attachment-circuit-04, 14 December 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-ntw-attachment-circuit-04>.

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

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

[PYANG] "pyang", 2023, <https://github.com/mbj4668/pyang>.

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

[RFC2453] Malkin, G., "RIP Version 2", STD 56, RFC 2453, DOI 10.17487/RFC2453, November 1998, <https://www.rfc-editor.org/rfc/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/rfc/rfc3644>.

[RFC3849] Huston, G., Lord, A., and P. Smith, "IPv6 Address Prefix Reserved for Documentation", RFC 3849, DOI 10.17487/RFC3849, July 2004, <https://www.rfc-editor.org/rfc/rfc3849>.

[RFC5398] Huston, G., "Autonomous System (AS) Number Reservation for Documentation Use", RFC 5398, DOI 10.17487/RFC5398, December 2008, <https://www.rfc-editor.org/rfc/rfc5398>.

[RFC5737] Arkko, J., Cotton, M., and L. Vegoda, "IPv4 Address Blocks Reserved for Documentation", RFC 5737, DOI 10.17487/RFC5737, January 2010, <https://www.rfc-editor.org/rfc/rfc5737>.

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

[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/rfc/rfc6151>.

[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/rfc/rfc6952>.

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

[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/rfc/rfc8299>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/rfc/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/rfc/rfc8349>.

[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/rfc/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/rfc/rfc8695>.

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

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 instance data can also be found at [Instance-Data].

A.1. Create A New Bearer

An example of a request message body to create a bearer is shown in Figure 21.

```
{
  "ietf-bearer-svc:bearers": {
    "bearer": [
      {
        "id": "an-identifier",
        "description": "A bearer example",
        "customer-point": {
          "device": {
            "device-id": "CE_X_SITE_Y"
          }
        },
        "requested-type": "ietf-bearer-svc:ethernet"
      }
    ]
  }
}
```

Figure 21: Example of a Message Body to Create A New Bearer

A bearer-reference is then generated by the controller for this bearer. Figure 22 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": [
      {
        "id": "an-identifier",
        "description": "A bearer example",
        "customer-point": {
          "device": {
            "device-id": "CE_X_SITE_Y"
          }
        },
        "requested-type": "ietf-bearer-svc:ethernet",
        "bearer-reference": "line-156"
      }
    ]
  }
}
```

Figure 22: Example of a Response Message Body with the Bearer Reference

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 23. 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-ac-start": "2023-12-12T05:00:00.00Z",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q"
          },
          "bearer-reference": "line-156"
        }
      }
    ]
  }
}
```


Figure 23: Example of a Message Body to Request an AC over an Existing Bearer

Figure 24 shows the message body of a response received from the controller and which 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",
        "requested-ac-start": "2023-12-12T05:00:00.00Z",
        "l2-connection": {
          "encapsulation": {
            "type": "ietf-vpn-common:dot1q",
            "dot1q": {
              "tag-type": "ietf-vpn-common:c-vlan",
              "cvlan-id": 550
            }
          },
          "bearer-reference": "line-156"
        }
      }
    ]
  }
}
```


Figure 24: Example of a Message Body of a Response to Assign a CVLAN 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 25. In this example, the peer SAP identifier points to an identifier of a service function. The (topological) location of that service function is assumed to be known to the network controller. For example, this can be determined as part of an on-demand procedure to instantiate a service function in a cloud. That instantiated service function can be granted a connectivity service via the provider network.

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


Figure 25: Example of a Message Body to Request an AC with a Peer SAP

A.4. One CE, Two ACs

Let’s consider the example of an eNodeB (CE) that is directly connected to the access routers of the mobile backhaul (see Figure 26). In this example, two ACs are needed to service the eNodeB (e.g., distinct VLANs for Control and User Planes).

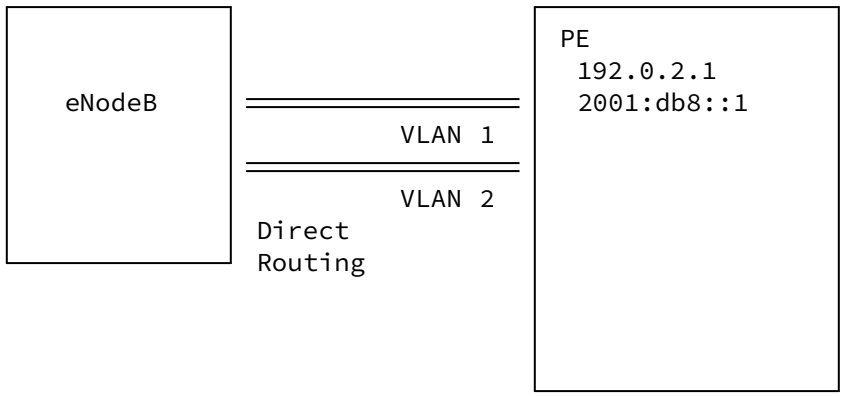


Figure 26: Example of a CE-PE ACs

An example of a request to create the ACs to service the eNodeB is shown in Figure 27. 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 27: Example of a Message Body to Request Two ACs on The Same Link (Not Recommended)

Figure 28 shows the message body of a 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 28: Example of a Message Body of a Response to Create Two ACs on The Same Link (Not Recommended)

The example shown Figure 28 is not optimal as it includes many redundant data. Figure 29 shows a more compact request that factorizes all the redundant data.

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

Figure 29: 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 30. 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": [
      {
        "name": "ac3",
        "description": "a third AC with a same peer node",
        "ac-group-profile": [
          "simple-node-profile"
        ],
        "l2-connection": {
          "encapsulation": {
            "dot1q": {
              "cvlan-id": 3
            }
          },
          "bearer-reference": "line-156"
        }
      }
    ]
  }
}
```

Figure 30: 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 32. 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).

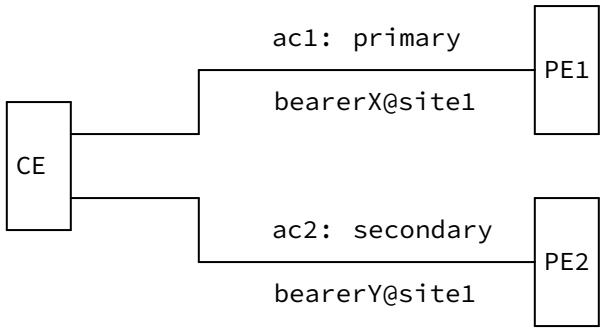


Figure 31: An Example Topology for AC Precedence Enforcement

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

Figure 32: Example of a Message Body to Associate a Precedence Level with ACs

A.6. Create Multiple ACs Bound to Multiple CEs

Figure 33 shows an example of CEs that are interconnected by a service provider network.

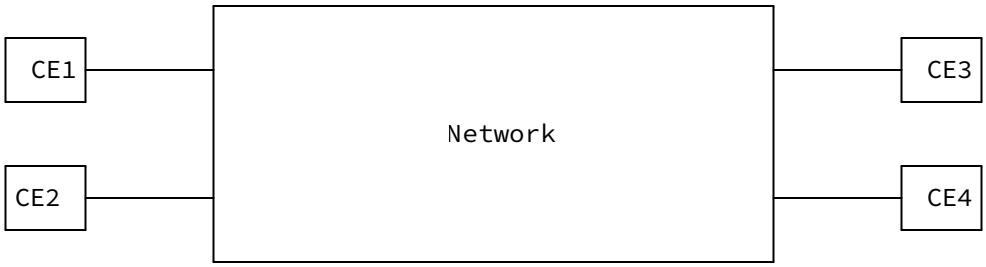


Figure 33: Network Topology Example

Figure 34 depicts an example of the message body of a response to a request to instantiate the various ACs that are shown in Figure 33.

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

```
        "ac-group-profile": [
          "simple-profile"
        ],
        "l2-connection": {
          "bearer-reference": "ce4-network"
        }
      }
    ]
  }
}
```

Figure 34: 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 [I-D.ietf-teas-ietf-network-slice-nbi-yang] to connect a site to a slice service.

First, Figure 35 describes the end-to-end network topology as well the orchestration scopes:

- The topology is made up of two sites (site1 and site2), interconnected via a Transport Network (e.g. IP/MPLS Network). A Network Function is deployed within each site in a dedicated IP Subnet.

- A 5G SMO is responsible for the deployment Network Functions and the indirect management of a local Gateway (i.e., CE device).

- An IETF Network Slice Controller is responsible for the deployment of IETF Network Slices across the TN.

Network Functions are deployed within each site.

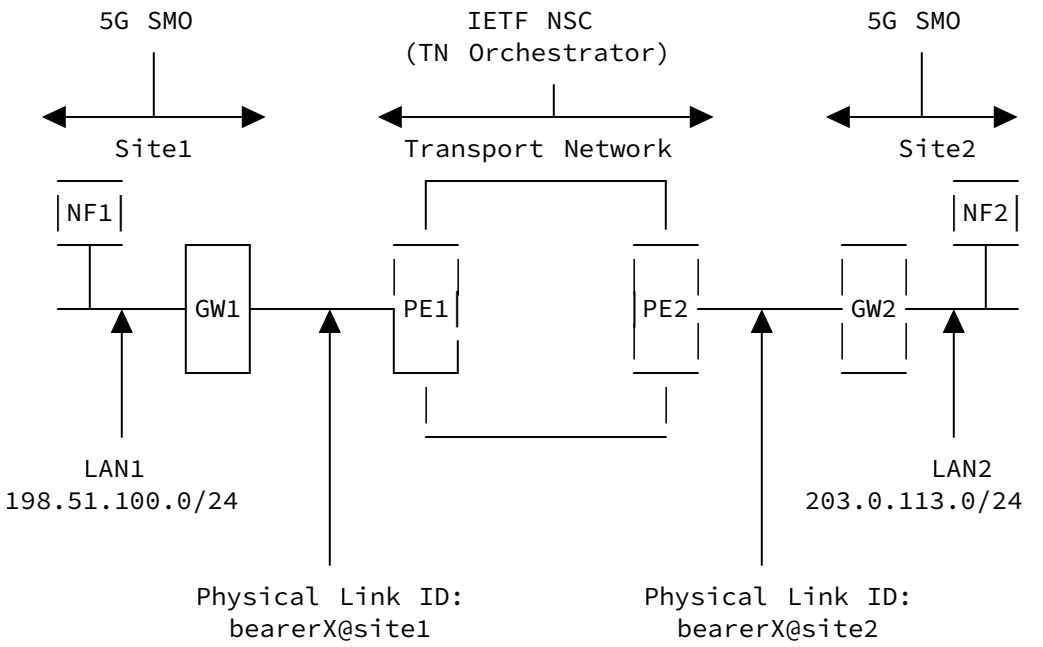
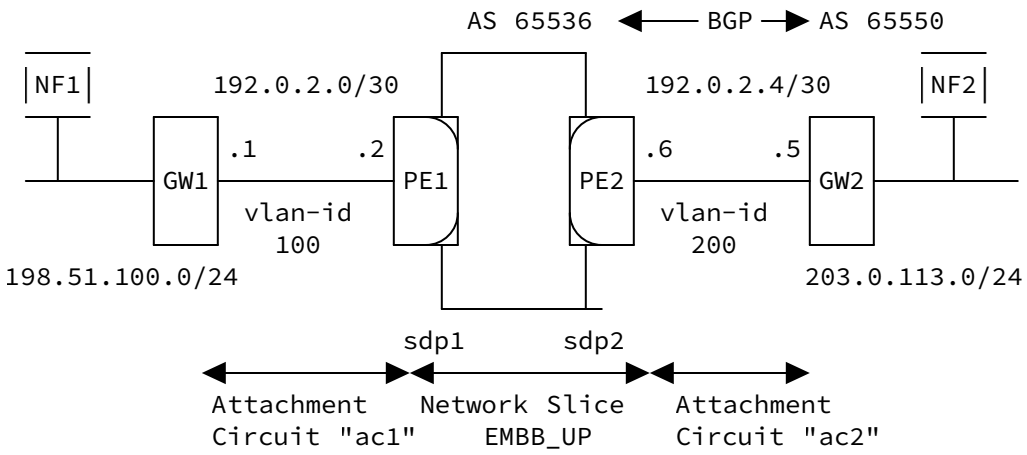


Figure 35: An Example of a Network Topology Used to Deploy Slices

Figure 36 describes the logical connectivity enforced thanks to both IETF Network Slice and Attachment Circuit 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
    customer-as: 65550
    customer-address: 192.0.2.5

Figure 36: Logical Overview

Figure 37 shows the message body of the request to create the required ACs using the Attachment Circuit 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-prefixes":[
                                    {
                                       "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 37: Message Body of a Request to Create Required ACs

Figure 38 shows the message body of a reponse received from the controller.

```
{
  "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",
              "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-prefixes": [
                    {
                      "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",
              "cvlan-id": 200
            }
          },
          "bearer-reference": "beareY@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 38: Example of a Message Body of a Response Indicating the Creation of the ACs

Figure 39 shows the message body of the request to create the a slice service bound to the ACs created using Figure 37. Only references to these ACs are included in the Slice Service request. This example assumes that the module that "glues" the service/AC is also supported by the NSC.

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

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

Figure 39: Message Body of a Request to Create a Slice Service Referring to the ACs

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

This example (Figure 40) 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 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).

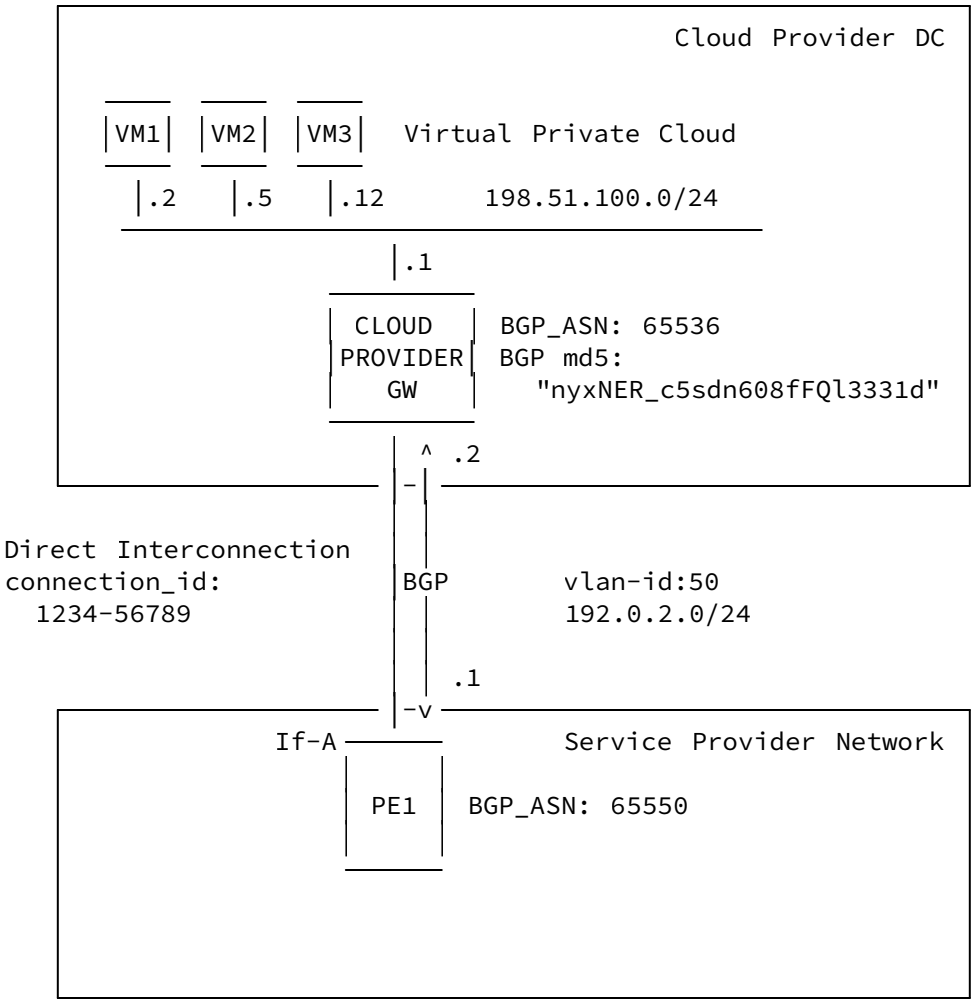


Figure 40: An Example of Realization for Connecting a Cloud Site

Figure 41 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".

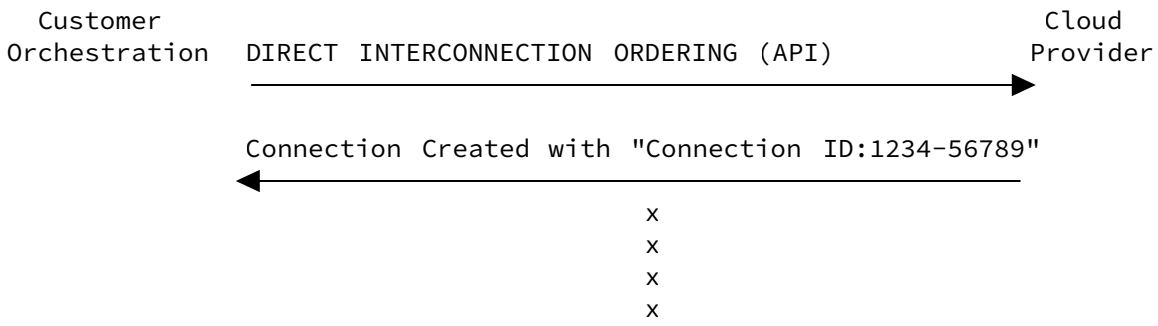


Figure 41: Illustration of Pre-provisioning

Next, API workflows can be initiated:

*Cloud Provider for the configuration as per (3) above.

*Service provider network via the Attachment Circuit model. This request can be used in conjunction with additional requests based on L3SM (VPN provisioning) or Network Slice Service model (5G hybrid Cloud deployment).

Figure 42 shows the message body of the request to create the required ACs to connect the Cloud Provider Virtualized (VM) using the Attachment Circuit 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 42: Message Body of a Request to Create the ACs for Connecting to the Cloud Provider

Figure 43 shows the message body of the response received from the provider. 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",
        "requested-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": {
                      "keying-material": {
                        "md5-keychain": "nyxNER_c5sdn608fFQl3331d"
                      }
                    }
                  }
```

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

Figure 43: Message Body of a Response to the Request to Create ACs for Connecting to the Cloud Provider

Acknowledgments

The document leverages [RFC9182] and [RFC9291].

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

Contributors

Victor Lopez
Nokia

Email: victor.lopez@nokia.com

Ivan Bykov
Ribbon Communications

Email: Ivan.Bykov@rbbn.com

Qin Wu
Huawei

Email: bill.wu@huawei.com

Kenichi Ogaki
KDDI

Email: ke-oogaki@kddi.com

Luis Angel Munoz
Vodafone

Email: luis-angel.munoz@vodafone.com

Authors' Addresses

Mohamed Boucadair (editor)

Orange

Email: mohamed.boucadair@orange.com

Richard Roberts (editor)
Juniper

Email: rroberts@juniper.net

Oscar Gonzalez de Dios
Telefonica

Email: oscar.gonzalezdedios@telefonica.com

Samier Barguil Giraldo
Nokia

Email: samier.barguil_giraldo@nokia.com

Bo Wu
Huawei Technologies

Email: lana.wubo@huawei.com