Subscriber Management Configuration Guide

Version 20.5.1-rc0, 25 May 2020

Registered Address	Support	Sales
26, Kingston Terrace, Princeton, New Jersey 08540, United States		
		+91 80 4850 5445
http://www.rtbrick.com	support@rtbrick.com	sales@rtbrick.com

Table of Contents

1. Introduction to Subscriber Management

The modular, scaleable subscriber management that RtBrick, Inc. calls the next generation access infrastructure (ngaccess) is based on existing protocol implementations from the legacy access daemon accessd. However, this next generation access is based on distributed services instead of a monolithic accessd access daemon.

1.1. Subscriber Management Daemons

There are four main daemons in the RtBrick distributed architecture:

- *subscriberd* is for subscriber management and AAA (which can be local, through RADIUS, or other methods)
- *pppoed* is to handle PPPoE and PPP sessions
- *l2tpd* is for L2TP tunnel and session handling
- *dhcpd* is for IP over Ethertnet (IPoE) session handling (this is for future development)

These distributed ngaccess components share a central database with other processes such as address pooling (poold) and forwarding (fwdd). The relationship between these processes is shown in Figure 1.

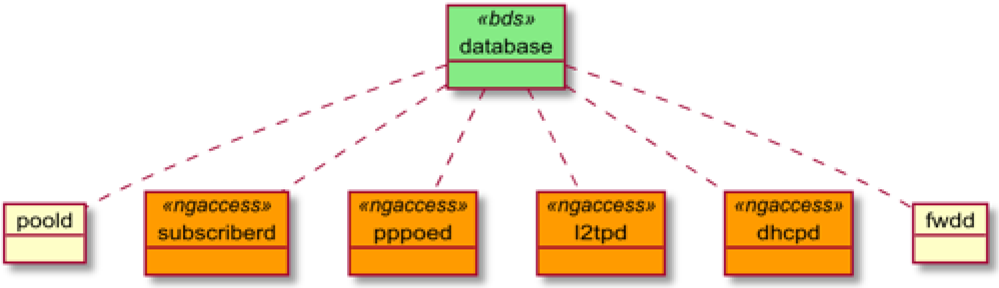

Figure 1. The Next Generation Access (ngaccess) Infrastructure

The result of using this distributed services architecture is increased scaling and stability. A new subscriber interface table serves as an anchor point for QoS, IGMP, MLD, and other aspects of subscriber management.

Legacy access methods based on a monolithic accessd process are compared to the new next generation access infrastructure in Table 1.

Table 1. Legacy and Next Generation Access Compared

Property	Legacy Access with monolithic accessd	Distributed Next Generation Access (ngaccess)
Implementation	Initial prototype	Carrier Grade implementation
Architecture	Monolithic (PPPoE, L2TP, RADIUS and more in single daemon)	Distributed (subscriberd, ppoed, dhcpd, l2tpd, and more if needed)
Scaling	NA	Multiple instance of each daemon possible (sharding)
Configuration engine	NA	Separate daemon for configuration (confd)

1.2. Configuration and Profiles

Configuration is a sophisticated feature of the ngaccess method. The main interface configuration for a physical interface (ifp) and associated VLANs is related to a series of profiles that hold parameters for authentication with AAA, services like IGMP and MLD, access methods like PPPoE and the like, and so on. The overall structure of this configuration and profile system is shown in Figure 2.

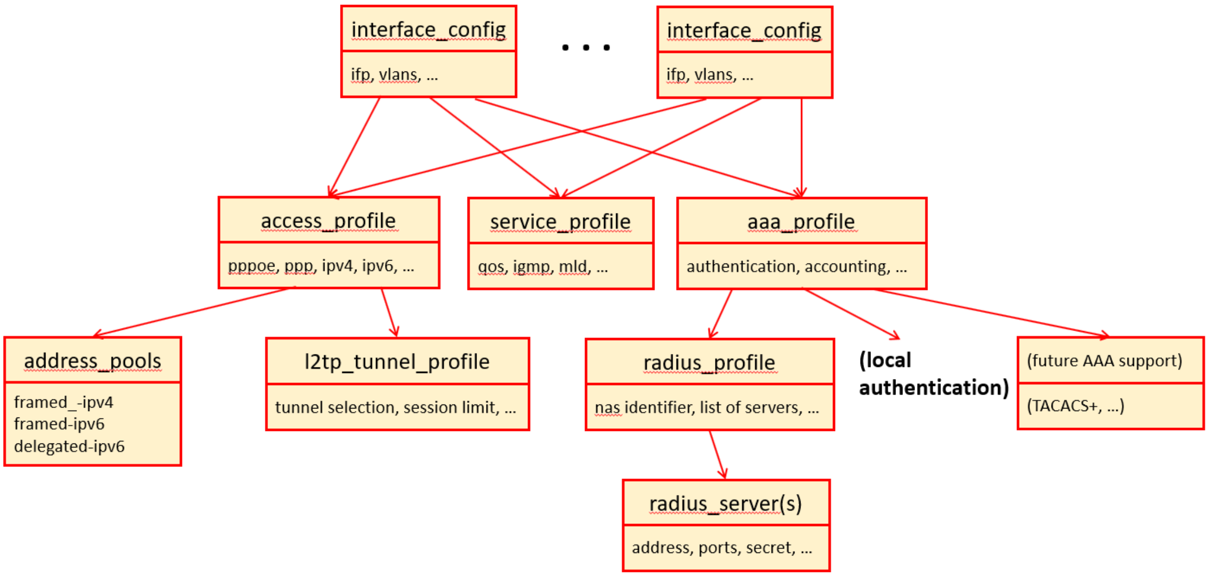

Figure 2. Configuration and Profiles

The major components of this system are:

- interface_config
  - Defines matching ifp and vlans
  - Assigns the access, service, and AAA profiles
  - Supports multiple interface configurations for each ifp and also disjoint VLAN ID ranges

- access_profile
  - Defines access protocols configuration
  - Assigns local IP address pools
  - Assigns L2TP tunnel profile for L2TP LAC
- service_profile
  - assigns service-related configurations for QoS, IGMP, and so on
  - allows dynamic overriding with RADIUS or through API
- aaa_profile
  - Holds profile for Authentication, Authorization, and Accounting
  - Controls selection of AAA type and order
  - Assigns a RADIUS profile to the AAA profile
- radius_profile
  - Holds RADIUS server address and ports
  - Holds RADIUS secret and more
- l2tp_tunnel_profile
  - Holds L2TP LAC configuration
  - Provides tunnel selection algorithm
  - Controls default session limit
  - Other parameters

The rest of this document provides details about this overall architecture. Because many levels of the subscriber management hierarchy use configuration profiles that are assumed to be already done, it makes sense to handle the overall flow in a “bottom-up” fashion, starting with RADIUS configuration and ending with interface configuration. When done in this fashion, no profile has to be named before it is created.

1.3. Subscriber Management and Sharding

RtBrick process daemons support multiple virtual instances of almost any software component. This capability is called *sharding* in RtBrick documentation. Shards are identified by adding a numeral to the designated daemon. The default target for statements that do not include a shard number is the main instance, or “shard.1”.

For example, if the PPPoE daemon has multiple instances, the target of the command or statement can be determined by using pppoed.1, pppoed.2, and so on.

All examples in this document use only the daemon name (for example, pppoed) for simplicity and clarity. Keep in mind that there are circumstances where a shard number is required to reach the desired result.

2. Access Hierarchy

All of the configuration and profile sections are edited under the access component of the confd daemon.

```
root@leaf1:confd> edit access
  aaa-profile                Global radius aaa profile configuration
  access-profile
  interface-name             Interface name
  l2tp-pool                  L2tp pool configuration mode
  l2tp-profile               L2TP profile configuration mode
  pool                       IP pool
  radius-profile             Global radius aaa profile configuration
  radius-server
  service-profile            Service profile
  user-profile               User profile name
  [<Enter>]                  Execute the command
[ ]
```

Each of these configuration modes and profiles are detailed in chapters of this document.

First, this guide configures the three high-level components that are mandatory (the list also has the tables affected):

- interface-config (global.access.interface.config)
- access-profile (global.access.profile.config)
- aaa-profile (global.aaa.interface.config)

Then, this guide configures the optional profiles and configurations (the list also has the tables affected):

- radius-profile (global.access.aaa.radius.config)
- radius-server (global.access.radius.config)
- service-profile (global.access.service.profile.config)
- l2tp-tunnel-profile (global.access.l2tp.pool.config)
- address-pools (global.pools.address-assignment.config)
- user-profile (global.access.user.profile.config)

The user profile is the only component not referenced by name. The key here is the username, which is used if AAA is configured for local authentication.

3. Access Interface Configuration

Although there is no “correct” way to configure subscriber management, it makes most sense to proceed from mandatory configurations and profile to optional ones. First and foremost, among these mandatory configuration items is the interfaces (or interfaces) to use for subscriber management.

Generally, interface configuration involves the following:

- Defining matching ifp and vlans
- Assigning the access, service, and AAA profiles
- Supporting multiple interface configurations for each ifp and also disjoint VLAN ID ranges

The way that the interface configuration relates to all subscriber management configuration tasks is shown in Figure 3. Note that there can be more than one interface configured for subscriber management and each interface can reference the same profiles.

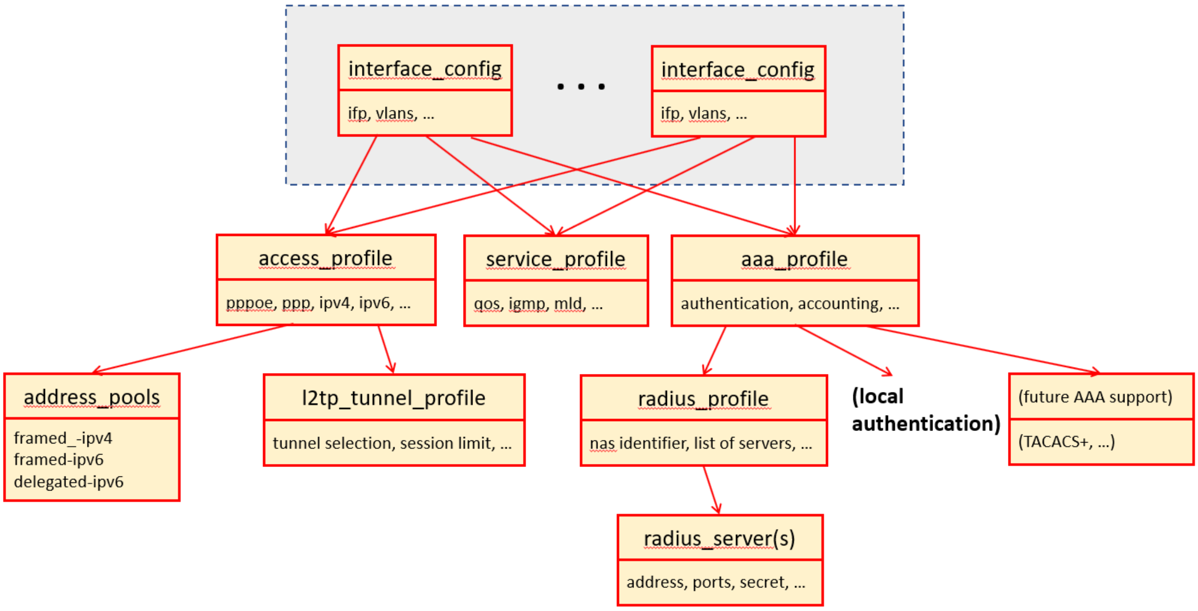

Figure 3. Access Interface Configuration and Overall Subscriber Management

There are four major configuration tasks for the interface, which are best handled in the following order:

1. Configure the interface name
2. Configure the access profile to use
3. Configure the AAA profile to use
4. Configure the service profile to use

3.1. Configuring the Access Interface Name

You can configure an access interface to use VLAN ID tags or not. Both configuration types are shown in this section.

3.1.1. Configuring an Untagged Interface

There are two options for configuring the untagged interface name: at the confd hierarchy level and at the access hierarchy level. Both options are shown below for untagged interface ifp-0/0/1. The running configuration is also shown at the end of the successful commit.

```
#
# Configuring untagged access interface
# Option 1: Configuring untagged access interface name from global mode
[ ]
root@rtbrick:confd> set access interface-name ifp-0/0/1
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring untagged access interface name from global access
mode [ access ]
root@rtbrick:confd> set interface-name ifp-0/0/1
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "interface-name:ifp-0/0/1": {
                 }
            }
         ],
    }
}
```

3.1.2. Configuring a VLAN ID Tagged Interface

There are two options for configuring the VLAN ID tagged interface name: at the confd hierarchy level and at the access hierarchy level. Both options are shown below for tagged interface ifp-0/0/1. The interface is configured to expect inner and outer VLAN ID tags in the range from 100 (min) to 300 (max).

The running configuration is also shown at the end of the successful commit.

```
#
# Configuring tagged access interface
# Option 1: Configuring tagged access interface name from global mode
[ ]
root@rtbrick:confd> set access interface-name ifp-0/0/1 inner-vlan-min 100
inner-vlan-max 100 outer-vlan-min 300 outer-vlan-max 300
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring tagged access interface name from global access mode
[ access ]
root@rtbrick:confd> set interface-name ifp-0/0/1 outer-vlan 300 - 300 inner-
vlan 100 - 100
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "interface-name:ifp-0/0/1 100 100 300 300": {
                    }
                }
            ],
        }
}
```

3.2. Configuring the Interface Access Profile

An access profile is a mandatory subscriber management component. You can configure an access profile for an untagged or VLAN ID tagged interface. Both are shown in this section.

3.2.1. Configuring an Access Profile on an Untagged Interface

There are two options for configuring the access profile: at the confd hierarchy level and at the access hierarchy level. Both options are shown below to configure access-profile1 for untagged interface ifp-0/0/1. The running configuration is also shown at the end of the successful commit.

```
#
# Configuring access profile on untagged interface
# Option 1: Configuring access profile untagged interface name from global
mode
[ ]
root@rtbrick:confd> set access interface-name ifp-0/0/1 access-profile-name
access-profile1
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring access profile on untagged access interface name from
global access mode

[ access ]
root@rtbrick:confd> set interface-name ifp-0/0/1 access-profile-name access-
profile1 root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "interface-name:ifp-0/0/1": {
                    "access-profile-name": "access-profile1"
                    }
                }
        ],
    }
}
```

3.2.2. Configuring an Access Profile on a VLAN ID Tagged Interface

There are two options for configuring the access profile: at the confd hierarchy level and at the access hierarchy level. Both options are shown below to configure access-profile1 for VLAN ID tagged interface ifp-0/0/1. The running configuration is also shown at the end of the successful commit.

```
#
# Configuring access profile on tagged interface
# Option 1: Configuring access profile on tagged interface name from global
mode
[ ]
root@rtbrick:confd> set access interface-name ifp-0/0/1 inner-vlan-min 100
inner-vlan-max 100 outer-vlan-min 300 outer-vlan-max 300 access-profile-name
access-profile1
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring access profile on tagged interface name from global
access mode
[ access ]
root@rtbrick:confd> set interface-name ifp-0/0/1 inner-vlan-min 100 inner-
vlan-max 100 outer-vlan-min 300 outer-vlan-max 300 access-profile-name
access-profile1 root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
"running-configuration": {
    "access": [
        {
            "interface-name:ifp-0/0/1 100 100 300 300": {
                "access-profile-name": "access-profile1"
                }
            }
        ],
    }
}
```

3.3. Configuring the Interface AAA Profile

An AAA profile is a mandatory subscriber management component. You can configure an AAA profile for an untagged or VLAN ID tagged interface. Both are shown in this section.

3.3.1. Configuring an AAA Profile on an Untagged Interface

There are two options for configuring the AAA profile: at the confd hierarchy level and at the access hierarchy level. Both options are shown below to configure aaa-profile1 for untagged interface ifp-0/0/1. The running configuration is also shown at the end of the successful commit.

```
#
# Configuring aaa profile on untagged interface
# Option 1: Configuring aaa profile untagged interface name from global mode
[ ]
root@rtbrick:confd> set access interface-name ifp-0/0/1 aaa-profile-name aaa-
profile1 root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring aaa profile on untagged access interface name from
global access mode
[ access ]
root@rtbrick:confd> set interface-name ifp-0/0/1 aaa-profile-name aaa-
profile1 root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "interface-name:ifp-0/0/1": {
                    "aaa-profile-name": "aaa-profile1"
                    }
            }
        ],
    }
}
```

3.3.2. Configuring an AAA Profile on a VLAN ID Tagged Interface

There are two options for configuring the AAA profile: at the confd hierarchy level and at the access hierarchy level. Both options are shown below to configure aaa-profile1 for VLAN ID tagged interface ifp-0/0/1. The running configuration is also shown at the end of the successful commit.

```
#
# Configuring aaa profile on tagged interface
# Option 1: Configuring aaa profile on tagged interface name from global mode
[ ]
root@rtbrick:confd> set access interface-name ifp-0/0/1 inner-vlan-min 100
inner-vlan-max 100 outer-vlan-min 300 outer-vlan-max 300 aaa-profile-name
aaa-profile1
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring aaa profile on tagged interface name from global
access mode [ access ]
root@rtbrick:confd> set interface-name ifp-0/0/1 inner-vlan-min 100 inner-
vlan-max
100 outer-vlan-min 300 outer-vlan-max 300 aaa-profile-name aaa-profile1
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "interface-name:ifp-0/0/1 100 100 300 300": {
                    "aaa-profile-name": "aaa-profile1"
                    }
              }
        ],
    }
}
```

3.4. Configuring the Interface Service Profile

Unlike the other profile associated with a subscriber management interface, a service profile is an *optional* subscriber management component. If you configure a service profile, you can configure the service profile for an untagged or VLAN ID tagged interface. Both are shown in this section.

3.4.1. Configuring a Service Profile on an Untagged Interface

There are two options for configuring the optional service profile: at the confd hierarchy level and at the access hierarchy level. Both options are shown below to configure service-profile1 for untagged interface ifp-0/0/1. The running configuration is also shown at the end of the successful commit.

```
#
# Configuring service profile on untagged interface
# Option 1: Configuring service profile untagged interface name from global
mode
[ ]
root@rtbrick:confd> set access interface-name ifp-0/0/1 service-profile-name
service-profile1
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring service profile on untagged access interface name
from global access mode
[ access ]
root@rtbrick:confd> set interface-name ifp-0/0/1 service-profile-name
service-profile1
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "interface-name:ifp-0/0/1": {
                    "service-profile-name": "service-profile1"
                    }
              }
        ],
    }
}
```

3.4.2. Configuring a Service Profile on a VLAN ID Tagged Interface

There are two options for configuring the optional service profile: at the confd hierarchy level and at the access hierarchy level. Both options are shown below to configure service-profile1 for VLAN ID tagged interface ifp-0/0/1. The running configuration is also shown at the end of the successful commit.

```
#
# Configuring service profile on tagged interface
# Option 1: Configuring service profile on tagged interface name from global
mode
[ ]
root@rtbrick:confd> set access interface-name ifp-0/0/1 inner-vlan-min 100
inner-vlan-max 100 outer-vlan-min 300 outer-vlan-max 300 service-profile-name
service-profile1
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring service profile on tagged interface name from global
access mode
[ access ]
root@rtbrick:confd> set interface-name ifp-0/0/1 inner-vlan-min 100 inner-
vlan-max
100 outer-vlan-min 300 outer-vlan-max 300 service-profile-name service-
profile1 root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
             {
                "interface-name:ifp-0/0/1 100 100 300 300": {
                "service-profile-name": "service-profile1"
                }
            }
        ],
    }
}
```

4. Access Profile Configuration

While it is mandatory to configure an interface with an access profile name , such as access-profile1, it is still necessary to configure the properties and parameters of the access profile itself.

Generally, access profile configuration involves the following:

- Defining the access protocols configuration
- Assigning local IP address pools
- Assigning L2TP tunnel profile for L2TP LAC

The way that the access profile configuration relates to all subscriber management configuration tasks is shown in Figure 4. Note that there can be more than one access interface configured for subscriber management and each interface can reference the same access profile.

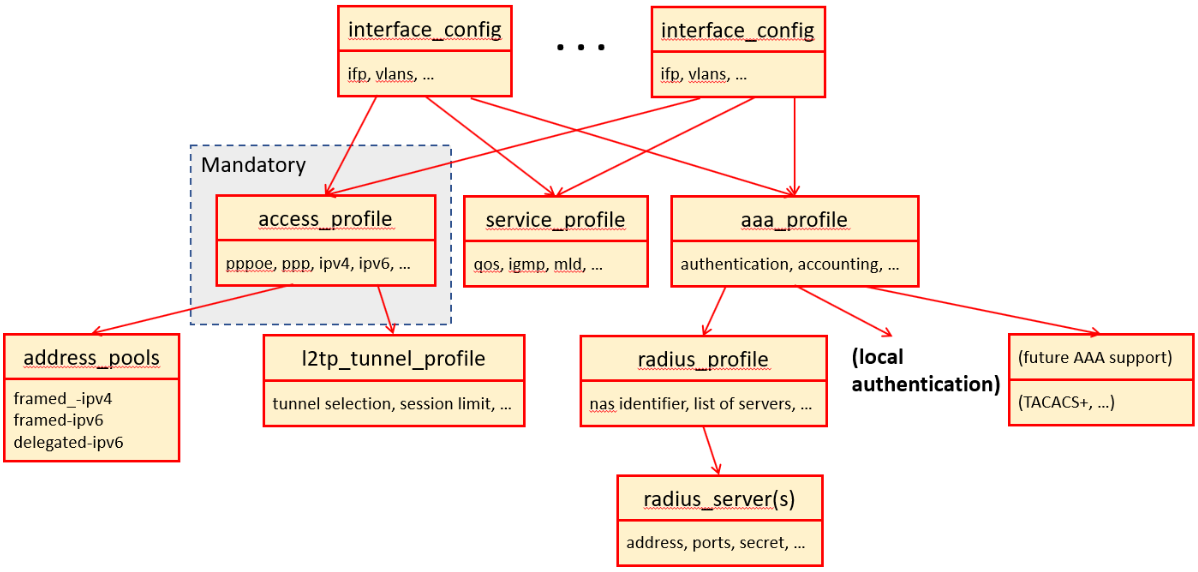

Figure 4. Access Profile Configuration and Overall Subscriber Management

4.1. Configuring the Access Profile Name

There are two options for configuring the access profile name: at the confd hierarchy level and at the access hierarchy level. Both options are shown below to configure access-profile1. User typing in shown in **bold**. The running configuration is also shown at the end of the successful commit.

```
#
# Creating access Profile Name
# Option 1: Creating access profile name from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1
root@rtbrick:confd> commit
Commit succeed
# Option 2: Creating access profile name from global access mode
[ access ]
root@rtbrick:confd> set access-profile access-profile1
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
            }
        ],
    }
}
```

4.2. Configuring the Access Type

There are two options for configuring the access type: at the confd hierarchy level and at the access hierarchy level. Both options are shown below to configure the access type, which can be:

- PPPoE
- L2TP (This means actually L2TP LNS which is currently not supported.)
- DHCP (This means actually IPoE which is currently not supported.)

Setting the access type determines the format of packets sent and received on the access interface. User typing in shown in **bold**. The running configuration is also shown at the end of the successful commit.

4.2.1. Setting the Access Type as PPPoE

Do the following to set the access type to PPPoE.

```
#
# Setting access type as PPPoE
# Option 1: Setting access type as PPPoE from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 access-type
PPPoE root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting access type as PPPoE from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set access-type PPPoE
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "access-type": "PPPoE"
                   }
           ],
      }
}
```

4.2.2. Setting the Access Type as L2TP

Do the following to set the access type to L2TP.

```
#
# Setting access type as L2TP
# Option 1: Setting access type as L2TP from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 access-type
L2TP root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting access type as L2TP from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set access-type L2TP
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "access-type": "L2TP"
                    }
                ],
        }
}
```

4.2.3. Setting the Access Type as DCHP

Do the following to set the access type to DCHP.

```
#
# Setting access type as DHCP
# Option 1: Setting access type as DHCP from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 access-type
DHCP root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting access type as DHCP from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set access-type DHCP
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "access-type": "DHCP"
                }
        ],
    }
}
```

4.3. Configuring IPv4 Address Family-Specific Parameters

There are two options for configuring the IPv4 address family-specific parameters: at the confd hierarchy level and at the access hierarchy level. Both options are shown below to configure the IPv4 address family-specific parameters, which are the:

- IPv4 access service
- Local pool name
- Primary DNS server
- Secondary DNS server
- Static IPv4 address
- IPv4 subscriber instance

Setting the IPv4 address family-specific parameters determines how IPv4 packets are handled on the access interface. User typing in shown in **bold**. The running configuration is also shown at the end of the successful commit.

4.3.1. Enabling the IPv4 Access Service

Do the following to enable the IPv4 access service (this case shows PPPoE, but the same can be done for other access types).

```
#
# Enabling IPv4 Access Service
# Option 1: Enable ipv4 access service from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 address-family
ipv4 enable
root@rtbrick:confd> commit
Commit succeed

# Option 2: Enabling ipv4 access service from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set address-family ipv4 enable
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Enabling ipv4 access service from access-profile address-family
mode [ access access-profile access-profile1 address-family ipv4 ]
root@rtbrick:confd> set enable
[ access access-profile access-profile1 address-family ipv4 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                "access-type": "PPPoE"
                "address-family ipv4": {
                "enable": true
                  }
                }
            }
        ],
    }
}
```

4.3.2. Configuring the Local Pool Name

Do the following to configure local address pool name pool1 (this case shows PPPoE, but the same can be done for other access types). In this case, there are three options for hierarchy level.

```
#
# Configuring Local Pool Name
# Option 1: Configure local pool name from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 address-family
ipv4
pool-name pool1
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configure local pool name from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set address-family ipv4 pool-name pool1
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Enabling ipv4 access service from access-profile address-family
mode [ access access-profile access-profile1 address-family ipv4 ]
root@rtbrick:confd> set pool-name pool1
[ access access-profile access-profile1 address-family ipv4 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                "access-type": "PPPoE"
                "address-family ipv4": {
                "enable": true,
                "pool-name": "pool1"
               }
            }
        ],
    }
}
```

4.3.3. Configuring Primary DNS Server

Do the following to configure a primary DNS server for the local address pool name pool1 (this case shows PPPoE, but the same can be done for other access types). In this case, there are three options for hierarchy level.

```
#
# Configuring Primary DNS Server
# Option 1: Configure primary dns server from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 address-family
ipv4 primary-dns 192.168.1.1
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configure primary dns server from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set address-family ipv4 primary-dns 192.168.1.1
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configure primary dns from access-profile address-family mode
[ access access-profile access-profile1 address-family ipv4 ]
root@rtbrick:confd> set primary-dns 192.168.1.1
[ access access-profile access-profile1 address-family ipv4 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                  "access-type": "PPPoE"
                  "address-family ipv4": {
                    "enable": true,
                    "pool-name": "pool1",
                    "primary-dns": "192.168.1.1"
                     }
                 }
              }
          ],
    }
}
```

4.3.4. Configuring Secondary DNS Server

Do the following to configure a secondary DNS server for the local address pool name pool1 (this case shows PPPoE, but the same can be done for other access types). In this case, there are three options for hierarchy level.

```
#
# Configuring Secondary DNS Server
# Option 1: Configure secondary dns server from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 address-family
ipv4 secondary-dns 192.168.1.2
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configure secondary dns server from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set address-family ipv4 secondary-dns 192.168.1.2
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configure secondary dns from access-profile address-family mode
[ access access-profile access-profile1 address-family ipv4 ]
root@rtbrick:confd> set secondary-dns 192.168.1.2
[ access access-profile access-profile1 address-family ipv4 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "access-type": "PPPoE"
                    "address-family ipv4": {
                        "enable": true,
                        "pool-name": "pool1",
                        "primary-dns": "192.168.1.1",
                        "secondary-dns": "192.168.1.2"
                     }
                }
            ],
      }
}
```

4.3.5. Configuring the IPv4 Subscriber Instance

Do the following to configure an IPv4 subscriber instance for the local address pool name pool1 (this case shows PPPoE, but the same can be done for other access types). In this case, there are three options for hierarchy level.

```
#
# Configuring IPv4 Subscriber Instance
# Option 1: Configure ipv4 subscriber instance from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 address-family
ipv4 instance subscriber
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configure ipv4 subscriber instance from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set address-family ipv4 instance subscriber
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configure ipv4 subscriber instance from access-profile address-
family mode
[ access access-profile access-profile1 address-family ipv4 ]
root@rtbrick:confd> set instance subscriber
[ access access-profile access-profile1 address-family ipv4 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "access-type": "PPPoE"
                    "address-family ipv4": {
                        "enable": true,
                        "pool-name": "pool1",
                        "primary-dns": "192.100.1.1",
                        "secondary-dns": "192.200.1.1",
                        "static-ipv4": "50.1.1.2",
                        "instance": "subscriber"
                       }
                    }
           ],
    }
}
```

4.4. Configuring IPv6 Address Family-Specific Parameters

There are three options for configuring the IPv6 address family-specific parameters: at the confd hierarchy level, at the access hierarchy level, and at the address-family hierarchy level. All options are shown below to configure the IPv6 address family-specific parameters, which are the:

- IPv6 access service
- Local pool name
- Primary DNS server
- Secondary DNS server
- IPv6 subscriber instance

Setting the IPv6 address family-specific parameters determines how IPv6 packets are handled on the access interface. The biggest difference is the lack of static address configuration. User typing in shown in **bold**. The running configuration is also shown at the end of the successful commit.

4.4.1. Enabling the IPv6 Access Service

Do the following to enable the IPv6 access service (this case shows PPPoE, but the same can be done for other access types).

```
#
# Enabling IPv6 Access Service
# Option 1: Enable ipv6 access service from global mode
[ ]root@rtbrick:confd> set access access-profile access-profile1 address-
family ipv6 enable
root@rtbrick:confd> commit
Commit succeed

# Option 2: Enabling ipv6 access service from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set address-family ipv6 enable
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Enabling ipv6 access service from access-profile address-family
mode [ access access-profile access-profile1 address-family ipv6 ]
root@rtbrick:confd> set enable
[ access access-profile access-profile1 address-family ipv6 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "access-type": "PPPoE"
                    "address-family ipv6": {
                        "enable": true
                    }
                }
            ],
        }
}
```

4.4.2. Configuring the Local Pool Name

Do the following to configure local address pool name pool1 (this case shows PPPoE, but the same can be done for other access types).

```
#
# Configuring Local Pool Name
# Option 1: Configure local pool name from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 address-family
ipv6
pool-name pool1
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configure local pool name from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set address-family ipv6 pool-name pool1
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Enabling ipv6 access service from access-profile address-family
mode [ access access-profile access-profile1 address-family ipv6 ]
root@rtbrick:confd> set pool-name pool1
[ access access-profile access-profile1 address-family ipv6 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "access-type": "PPPoE"
                    "address-family ipv6": {
                        "enable": true,
                        "pool-name": "pool1"
                    }
                }
        ],
    }
}
```

4.4.3. Configuring Primary DNS Server

Do the following to configure a primary DNS server for the local address pool name pool1 (this case shows PPPoE, but the same can be done for other access types).

```
#
# Configuring Primary DNS Server
# Option 1: Configure primary dns server from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 address-family
ipv6 primary-dns 2001:db8:100:1:1::2
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configure primary dns server from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set address-family ipv6 primary-dns 2001:db8:100:1:1::2
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configure primary dns server from access-profile address-family
mode [ access access-profile access-profile1 address-family ipv6 ]
root@rtbrick:confd> set primary-dns 2001:db8:100:1:1::2
[ access access-profile access-profile1 address-family ipv6 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{

    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "access-type": "PPPoE"
                    "address-family ipv6": {
                        "enable": true,
                        "pool-name": "pool1",
                        "primary-dns": "2001:db8:100:1:1::2"
                    }
                }
            ],
    }
}
```

4.4.4. Configuring Secondary DNS Server

Do the following to configure a secondary DNS server for the local address pool name pool1 (this case shows PPPoE, but the same can be done for other access types).

```
#
# Configuring Secondary DNS Server
# Option 1: Configure secondary dns server from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 address-family
ipv6 secondary-dns 2001:db8:200:1:1::2
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configure secondary dns server from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set address-family ipv6 secondary-dns 2001:db8:200:1:1::2
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configure secondary dns from access-profile address-family mode
[ access access-profile access-profile1 address-family ipv6 ]
root@rtbrick:confd> set secondary-dns 2011:db8:200:1:1::2
[ access access-profile access-profile1 address-family ipv6 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "access-type": "PPPoE"
                    "address-family ipv6": {
                        "enable": true,
                        "pool-name": "pool1",
                        "primary-dns": "2001:db8:100:1:1::2",
                        "secondary-dns": "2001:db8:200:1:1::2"
                      }
                }
        ],
    }
}
```

4.4.5. Configuring the IPv6 Subscriber Instance

Do the following to configure an IPv6 subscriber instance (this case shows PPPoE, but the same can be done for other access types).

```
#
# Configuring IPv6 Subscriber Instance
# Option 1: Configure ipv6 subscriber instance from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 address-family
ipv6 instance subscriber
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configure ipv6 subscriber instance from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set address-family ipv6 instance subscriber
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configure ipv6 subscriber instance from access-profile address-
family mode
[ access access-profile access-profile1 address-family ipv6 ]
root@rtbrick:confd> set instance subscriber
[ access access-profile access-profile1 address-family ipv6 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "access-type": "PPPoE"
                    "address-family ipv6": {
                        "enable": true,
                        "pool-name": "pool1",
                        "primary-dns": "2001:db8:100:1:1::2",
                        "secondary-dns": "2001:db8:200:1:1::2",
                        "instance": "subscriber"
                      }
                }
         ],
    }
}
```

4.5. Configuring Protocol-Specific Parameters

There are three options for configuring the IPv6 address family-specific parameters: at the confd hierarchy level, at the access hierarchy level, and at the protocol hierarchy level. All options are shown below to configure the protocol-specific parameters, which are the:

- PPPoE parameter (enable)
- PPP parameters (LCP specifics)
- IPCP parameters for PPP
- IP6CP parameters for PPP
- Router advertisement specifics

The PPP protocol in particular has many parameters to set. User typing in shown in **bold**. The running configuration is also shown at the end of the successful commit.

4.5.1. Enabling the PPPoE Protocol

Do the following to enable the PPPoE protocol for an access profile.

```
#
# Enabling PPPoE protocol
# Option 1: Enable pppoe protocol from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol pppoe
enable root@rtbrick:confd> commit
Commit succeed

# Option 2: Enable pppoe protocol from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol pppoe enable
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Enable pppoe protocol from access-profile protocol mode
[ access access-profile access-profile1 protocol pppoe ]
root@rtbrick:confd> set enable
[ access access-profile access-profile1 protocol pppoe ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                "protocol pppoe": {
                "enable": true
               }
            }
        ],
    }
}
```

4.5.2. PPPoE Dual-Stack IPv4/IPv6 with DHCPv6

This section describes the IPv6 over PPPoE support.

The whole IPv6 control plane of an PPPoE session like ICMPv6 router-solicitation (RS), ICMPv6 router-advertisement (RA) and DHCPv6 will be handled in PPPoED doing the encapsulation and decapsulation of the whole PPPoE packet including IPv6, UDP, checksum validations (UDP or ICMPv6) and carried protocols.

PPPoED handles received router-solicitations by responding with router-advertisements and is sending frequent router-advertisements based on configured interval.

The other-config flag in the router-advertisement is automatically set if DHCPv6 is enabled for this particular subscriber. This flag signals that there are more information available via DHCPv6.

DHCPv6 over PPPoE is differently to DHCPv6 over ethernet because of the special characteristics of point-to-point protocols. DHCPv6 over PPPoE is supporting delegated IPv6 prefixes (IA_PD) and DNS options only. Unsupported IA options (IA_NA and IA_TA) or options which can be served will be rejected with a status code options as defined per RFC.

The delegated IPv6 prefix served by DHCPv6 will be assigned to the PPPoE session via RADIUS or local pool also if client is not requesting it via DHCPv6 protocol. If DHCPv6 is enabled but no delegated-prefix provided, only DNS is served in response if available.

4.5.2.1. DHCPv6 Server DUID

The DHCPv6 server identifier DUID is generated based on IP6CP negotiated interface-identifier as shown below:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |     DUID-Type 3 (DUID-LL)     |    hardware type 27 (EUI64)   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                      interface-identifier                     |
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.5.2.2. Configuring DHCPv6

To configure DHCPv6, enter the following command:

```
ubuntu@leaf1:~/development/libng-access$ rtb confd set access access-profile
access-profile1 protocol dhcpv6
enable              [<Enter>]           lifetime            preferred-
lifetime
```

4.5.3. PPP Protocol Specifics

Configuring the PPP protocol specifics is a lengthy process, due to the many parameters that can be set. The PPP protocol can adjust Link Control Protocol (LCP) specifics such as:

- Authentication protocol type (PAP, CHAP, PAP_CHAP, CHAP_PAP, none)
- Maximum configuration NACK
- Echo on or off
- Echo interval
- Echo maximum retries
- MRU value
- MRU negotiation on or off
- LCP retransmit interval and maximum retries

There are also many parameters associated with the Internet Protocol Control Protocol (IPCP) and IPCP for IPv6 (IP6CP), but these are listed in separate sections to reduce document complexity.

4.5.3.1. Configuring the Authentication Protocol Type

You can configure different types of authentication protocols to use with PPP. These include password authentication protocol (PAP), challenge-handshake authentication protocol (CHAP), combinations of both (PAP_CHAP or CHAP_PAP), and no authentication for PPP sessions (NONE).

This example configures PAP authentication for PPP in the access profile.

```
#
# Configuring Authentication Protocol Type
# Option 1: Configure authentication protocol type from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol ppp
lcp authentication-protocol PAP
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configure authentication protocol type from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol ppp lcp authentication-protocol PAP
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configure authentication protocol type from access-profile
protocol mode
[ access access-profile access-profile1 protocol ppp]
root@rtbrick:confd> set lcp authentication-protocol PAP
[ access access-profile access-profile1 protocol ppp ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "protocol ppp": {
                        "lcp": {
                        "authentication-protocol": "PAP"
                 }
             }
        ],
    }
}
```

4.5.3.2. Configuring the Maximum Configuration NACK

You can configure a maximum number of negative acknowledgements (NACKs) that the access profile will accept before terminating the PPP session. This example sets a value of 10 as the maximum NACKs for the LCP and PPP in the access profile.

```
#
# Setting maximum configuration nack
# Option 1: Setting maximum configuration nack from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol ppp
lcp
config-nack-max 10
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting maximum configuration nack from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol ppp lcp config-nack-max 10
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Setting maximum configuration nack from access-profile protocol
mode [ access access-profile access-profile1 protocol ppp]
root@rtbrick:confd> set lcp config-nack-max 10
[ access access-profile access-profile1 protocol ppp ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "protocol ppp": {
                        "lcp": {
                        "authentication-protocol": "PAP",
                        "config-nack-max": 10
                    }
            }
        ],
    }
}
```

4.5.3.3. Configuring Echo

You can configure LCP to echo requests and replies to form a loopback mechanism which helps in debugging, link quality determination, performance testing, and other functions. This example enablers echoes for LCP and PPP in the access profile.

```
#
# Enabling Echo
# Option 1: Enabling echo from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol ppp
lcp echo-enable
root@rtbrick:confd> commit
Commit succeed

# Option 2: Enabling echo from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol ppp lcp echo-enable
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Enabling echo from access-profile protocol mode
[ access access-profile access-profile1 protocol ppp]
root@rtbrick:confd> set lcp echo-enable
[ access access-profile access-profile1 protocol ppp ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "protocol ppp": {
                        "lcp": {
                        "authentication-protocol": "PAP",
                        "config-nack-max": 10,
                        "echo-enable": true
                    }
                }
        ],
    }
}
```

4.5.3.4. Configuring Echo Interval

You can configure LCP to expect request and reply echoes within a certain interval of time. If the interval is set too high or too low for the link, the PPP session drops. This example sets the echo interval to 10 seconds for LCP and PPP in the access profile.

```
#
# Configuring echo interval
# Option 1: Configuring echo interval from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol ppp
lcp
echo-interval 10
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring echo interval from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol ppp lcp echo-interval 10
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring echo interval from access-profile protocol mode
[ access access-profile access-profile1 protocol ppp]
root@rtbrick:confd> set lcp echo-interval 10
[ access access-profile access-profile1 protocol ppp ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "protocol ppp": {
                        "lcp": {
                        "authentication-protocol": "PAP",
                        "config-nack-max": 10,
                        "echo-enable": true,
                        "echo-interval": 10
                    }
                }
        ],
    }
}
```

4.5.3.5. Configuring Echo Maximum Retries

You can configure LCP to retry an expected request and reply echo. This example sets the echo maximum retries to 20 for LCP and PPP in the access profile.

```
#
# Configuring echo maximum retries
# Option 1: Configuring maximum retries from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol ppp
lcp
echo-max-retransmit 20
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring echo maximum retries from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol ppp lcp echo-max-retransmit 20
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring echo maximum retries from access-profile protocol
mode [ access access-profile access-profile1 protocol ppp]
root@rtbrick:confd> set lcp echo-max-retransmit 20
[ access access-profile access-profile1 protocol ppp ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "protocol ppp": {
                        "lcp": {
                        "authentication-protocol": "PAP",
                        "config-nack-max": 10,
                        "echo-enable": true,
                        "echo-interval": 10,
                        "echo-max-retransmit": 20
                    }
                }
        ],
    }
}
```

4.5.3.6. Configuring LCP Loop Detection

You can configure LCP to detect loops on the link. Note that the running configuration in this example enables loop detection by setting the lcp-loop-detection-disable setting to false.

```
#
# Enabling lcp loop detection
# Option 1: Enabling lcp loop detection from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol ppp
lcp
lcp-loop-detection
root@rtbrick:confd> commit
Commit succeed

# Option 2: Enabling lcp loop detection from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol ppp lcp lcp-loop-detection
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Enabling lcp loop detection from access-profile protocol mode
[ access access-profile access-profile1 protocol ppp ]
root@rtbrick:confd> set lcp lcp-loop-detection
[ access access-profile access-profile1 protocol ppp ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "protocol ppp": {
                        "lcp": {
                        "authentication-protocol": "PAP",
                        "config-nack-max": 10,
                        "echo-enable": true,
                        "echo-interval": 10,
                        "echo-max-retransmit": 20,
                        "lcp-loop-detection-disable": false
                    }
            }
        ],
    }
}
```

4.5.3.7. Configuring the MRU Value

You can configure LCP to accept a certain value for the maximum reception unit (MRU) for the PPP link. This differs from a link MTU size by including the PPP headers (essentially, MRU = PPP fields + MTU size). This example sets the MRU size to 1492 bytes.

```
#
# Configuring mru value
# Option 1: Configuring mru value from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol ppp
lcp mru
1492
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring mru value from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol ppp lcp mru 1492
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring mru value from access-profile protocol mode
[ access access-profile access-profile1 protocol ppp]
root@rtbrick:confd> set lcp mru 1492
[ access access-profile access-profile1 protocol ppp ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "protocol ppp": {
                        "lcp": {
                        "authentication-protocol": "PAP",
                        "config-nack-max": 10,
                        "echo-enable": true,
                        "echo-interval": 10,
                        "echo-max-retransmit": 20,
                        "lcp-loop-detection-disable": false,
                        "mru": 1492
                    }
            }
        ],
    }
}
```

4.5.3.8. Configuring MRU Negotiation

You can configure LCP to negotiate a value for the MRU for the PPP link. This example enables MRU negotiation.

```
#
# Enabling mru negotiation
# Option 1: Enabling mru negotiation from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol ppp
lcp
mru-negotiation
root@rtbrick:confd> commit
Commit succeed

# Option 2: Enabling mru negotiation from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol ppp lcp mru-negotiation
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Enabling mru-negotiation from access-profile protocol mode
[ access access-profile access-profile1 protocol ppp ]
root@rtbrick:confd> set lcp mru-negotiation
[ access access-profile access-profile1 protocol ppp ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "protocol ppp": {
                        "lcp": {
                        "authentication-protocol": "PAP",
                        "config-nack-max": 10,
                        "echo-enable": true,
                        "echo-interval": 10,
                        "echo-max-retransmit": 20,
                        "lcp-loop-detection-disable": false,
                        "mru": 1492,
                        “mru-negotiation”: true
                    }
            }
        ],
    }
}
```

4.5.3.9. Configuring LCP Retransmit Interval

You can configure an LCP retransmit interval for the PPP link. This example sets an LCP retransmit interval to 5 seconds.

```
#
# Configuring lcp retransmit interval
# Option 1: Configuring lcp retransmit interval from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol ppp
lcp retransmit-interval 5
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring lcp retransmit interval from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol ppp lcp retransmit-interval 5
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring lcp retransmit interval from access-profile protocol
mode [ access access-profile access-profile1 protocol ppp ]
root@rtbrick:confd> set lcp retransmit-interval 5
[ access access-profile access-profile1 protocol ppp ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "protocol ppp": {
                        "lcp": {
                        "authentication-protocol": "PAP",
                        "config-nack-max": 10,
                        "echo-enable": true,
                        "echo-interval": 10,
                        "echo-max-retransmit": 20,
                        "lcp-loop-detection-disable": false,
                        "mru": 1492,
                        “mru-negotiation”: true,
                        "retransmit-interval": 5
                    }
            }
        ],
    }
}
```

4.5.3.10. Configuring LCP Retransmit Maximum Retries

You can configure a maximum for LCP retransmit retries for the PPP link. This example sets an LCP retransmit maximum retries to 20.

```
#
# Configuring lcp retransmit max retries
# Option 1: Configuring lcp retransmit max retries from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol ppp
lcp retransmit-max-retry 20
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring lcp retransmit max retries from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol ppp lcp retransmit-max-retry 20
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring lcp retransmit interval from access-profile protocol
mode [ access access-profile access-profile1 protocol ppp ]
root@rtbrick:confd> set lcp retransmit-max-retry 20
[ access access-profile access-profile1 protocol ppp ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "protocol ppp": {
                        "lcp": {
                        "authentication-protocol": "PAP",
                        "config-nack-max": 10,
                        "echo-enable": true,
                        "echo-interval": 10,
                        "echo-max-retransmit": 20,
                        "lcp-loop-detection-disable": false,
                        "mru": 1492,
                        “mru-negotiation”: true,
                        "retransmit-interval": 5,
                        "retransmit-max-retry": 20
                    }
                }
        ],
    }
}
```

4.5.4. IPCP Protocol Specifics

The Internet Protocol Control Protocol (IPCP) specifics are configured for PPP, but are not the same as LCP parameters. IPCP can adjust PPP specifics for IPv4 such as:

- Maximum NACKs
- Retransmit interval
- Maximum retransmit retries
- Source logical interface (ifl) for address negotiation

4.5.4.1. Enabling IPCP for PPP

This example enables IPCP for PPP in the access profile.

```
#
# Enabling IPCP protocol
# Option 1: Enabling ipcp protocol from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol ppp
ipcp enable
root@rtbrick:confd> commit
Commit succeed

# Option 2: Enabling ipcp protocol from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol ppp ipcp enable
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Enabling ipcp protocol from access-profile protocol mode
[ access access-profile access-profile1 protocol ppp]
root@rtbrick:confd> set ipcp enable
[ access access-profile access-profile1 protocol ppp ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "ipcp": {
                        "ipcp-enable": true
                   }
             }
        ],
    }
}
```

4.5.4.2. Configuring the IPCP Maximum Configuration NACK

You can configure a maximum number of IPCP negative acknowledgements (NACKs) that the access profile will accept before terminating the PPP session. This example sets a value of 20 as the maximum NACKs for IPCP in the access profile.

```
#
# Setting max configuration nack
# Option 1: Setting max configuration nack from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol ppp
ipcp config-nack-max 20
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting max configuration nack from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol ppp ipcp config-nack-max 20
[ access access-profile access-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Setting max configuration nack from access-profile protocol mode
[ access access-profile access-profile1 protocol ppp]
root@rtbrick:confd> set ipcp config-nack-max 20
[ access access-profile access-profile1 protocol ppp ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "ipcp": {
                        "config-nack-max": 20,
                        "ipcp-enable": true
                    }
            }
        ],
    }
}
```

4.5.4.3. Configuring the IPCP Retransmit Interval

You can configure an IPCP retransmit interval for the PPP link. This example sets an IPCP retransmit interval to 10 seconds.

```
#
# Configuring Retransmit Interval
# Option 1: Configuring retransmit interval from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol ppp
ipcp retransmit-interval 10
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring retransmit interval from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol ppp ipcp retransmit-interval 10
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring retransmit interval from access-profile protocol mode
[ access access-profile access-profile1 protocol ppp]
root@rtbrick:confd> set ipcp retransmit-interval 10
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "ipcp": {
                        "config-nack-max": 20,
                        "ipcp-enable": true,
                        "retransmit-interval": 10
                    }
              }
        ],
    }
}
```

4.5.4.4. Configuring IPCP Retransmit Maximum Retries

You can configure a maximum for IPCP retransmit retries for the PPP link. This example sets an IPCP retransmit maximum retries to 30.

```
#
# Configuring max retransmit retries
# Option 1: Configuring max retransmit retries from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol ppp
ipcp retransmit-max-retry 30
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring max retransmit retries from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol ppp ipcp retransmit-max-retry 30
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring max retransmit retries from access-profile protocol
mode
[ access access-profile access-profile1 protocol ppp]
root@rtbrick:confd> set ipcp retransmit-max-retry 30
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "ipcp": {
                        "config-nack-max": 20,
                        "ipcp-enable": true,
                        "retransmit-interval": 10, "retransmit-max-retry": 30
                    }
                }
            }
        ],
    }
}
```

4.5.4.5. Configuring IPCP Source Logical Interface (IFL)

You can configure a source IFL for IPCP to negotiate an IPv4 address with the remote end. This example sets the source IFL to lo-0/0/0/0/1.

```
# Configuring Source IFL
#
# Configuring Source IFL
# Option 1: Configuring source ifl from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol ppp
ipcp source-ifl l0-0/0/0/0/1
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring source ifl from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol ppp ipcp source-ifl l0-0/0/0/0/1
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring source ifl from access-profile protocol mode
[ access access-profile access-profile1 protocol ppp]
root@rtbrick:confd> set ipcp source-ifl l0-0/0/0/0/1
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "ipcp": {
                        "config-nack-max": 20,
                        "ipcp-enable": true,
                        "retransmit-interval": 10, "retransmit-max-retry":
30,

                        "source-ifl": "lo-0/0/0/0/1"
                       }
                }
            }
        ],
    }
}
```

4.5.5. IP6CP Protocol Specifics

The IPv6 Control Protocol (IP6CP) specifics are configured for PPP, but are not the same as LCP parameters. IP6CP can adjust PPP specifics for IPv6 such as:

- Maximum NACKs
- Retransmit interval
- Maximum retransmit retries

- Source logical interface (ifl) for address negotiation

4.5.5.1. Enabling IP6CP for PPP

This example enables IP6CP for PPP in the access profile.

```
#
# Enabling IP6CP protocol
# Option 1: Enabling ip6cp protocol from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol ppp
ip6cp enable
root@rtbrick:confd> commit
Commit succeed

# Option 2: Enabling ip6cp protocol from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol ppp ip6cp enable
root@rtbrick:confd> commit
Commit succeed

# Option 3: Enabling ip6cp protocol from access-profile protocol mode
[ access access-profile access-profile1 protocol ppp]
root@rtbrick:confd> set ip6cp enable
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "ip6cp": {
                        "enable": true
                }
            }
        ],
    }
}
```

4.5.5.2. Configuring the IP6CP Maximum Configuration NACK

You can configure a maximum number of IP6CP negative acknowledgements (NACKs) that the access profile will accept before terminating the PPP session. This example sets a value of 20 as the maximum NACKs for IP6CP in the access profile.

```
#
# Setting max configuration nack
# Option 1: Setting max configuration nack from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol ppp
ip6cp config-nack-max 20
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting max configuration nack from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol ppp ip6cp config-nack-max 20
root@rtbrick:confd> commit
Commit succeed

# Option 3: Setting max configuration nack from access-profile protocol mode
[ access access-profile access-profile1 protocol ppp]
root@rtbrick:confd> set ip6cp config-nack-max 20
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "ip6cp": {
                        "config-nack-max": 20,
                        "enable": true
                    }
              }
         ],
    }
}
```

4.5.5.3. Configuring the IP6CP Retransmit Interval

You can configure an IP6CP retransmit interval for the PPP link. This example sets an IP6CP retransmit interval to 10 seconds.

```
#
# Configuring Retransmit Interva
# Option 1: Configuring retransmit interval from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol ppp
ip6cp retransmit-interval 10
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring retransmit interval from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol ppp ip6cp retransmit-interval 10
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring retransmit interval from access-profile protocol mode
[ access access-profile access-profile1 protocol ppp]
root@rtbrick:confd> set ip6cp retransmit-interval 10
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "ip6cp": {
                        "config-nack-max": 20,
                        "enable": true,
                        "retransmit-interval": 10
                    }
            }
        ],
    }
}
```

4.5.5.4. Configuring IP6CP Retransmit Maximum Retries

You can configure a maximum for IP6CP retransmit retries for the PPP link. This example sets an IP6CP retransmit maximum retries to 30.

```
#
# Configuring max retransmit retries
# Option 1: Configuring max retransmit retries from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol ppp
ip6cp retransmit-max-retry 30
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring max retransmit retries from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol ppp ip6cp retransmit-max-retry 30
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring max retransmit retries from access-profile protocol
mode [ access access-profile access-profile1 protocol ppp]
root@rtbrick:confd> set ip6cp retransmit-max-retry 30
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "ip6cp": {
                        "config-nack-max": 20,
                        "enable": true,
                        "retransmit-interval": 10,
                        "retransmit-max-retry": 30
                    }
                }
            }
        ],
    }
}
```

4.5.5.5. Configuring IP6CP Source Logical Interface (IFL)

You can configure a source IFL for IP6CP to negotiate an IPv6 address with the remote end. This example sets the source IFL to lo-0/0/0/0/1.

```
#
# Configuring Source IFL
# Option 1: Configuring source ifl from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol ppp
ip6cp source-ifl l0-0/0/0/0/1
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring source ifl from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol ppp ip6cp source-ifl l0-0/0/0/0/1
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring source ifl from access-profile protocol mode
[ access access-profile access-profile1 protocol ppp]
root@rtbrick:confd> set ip6cp source-ifl l0-0/0/0/0/1
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "ip6cp": {
                        "config-nack-max": 20,
                        "enable": true,
                        "retransmit-interval": 10,
                        "retransmit-max-retry": 30,
                        "source-ifl": "lo-0/0/0/0/1"
                    }
                }
            }
        ],
    }
}
```

4.5.6. Configure Router Advertisement Specifics

You can enable router advertisements on the PPP link and set an interval for the advertisements.

4.5.6.1. Enable Router Advertisements

Do the following to enable router advertisements for the access profile. This example sets the protocol router-advertisement to true.

```
# Enabling router advertisement protocol
# Option 1: Enabling router advertisement protocol from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol
router-advertisement enable
root@rtbrick:confd> commit
Commit succeed

# Option 2: Enabling router advertisement protocol from access-profile mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol router-advertisement enable
root@rtbrick:confd> commit
Commit succeed

# Option 3: Enabling router advertisement protocol from access-profile
protocol mode [ access access-profile access-profile1 protocol ppp]
root@rtbrick:confd> set router-advertisement enable
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "protocol router-advertisement": {
                        "enable": true
                    }
                }
           ],
    }
}
```

4.5.6.2. Configure the Router Advertisement Interval

Do the following to enable the router advertisement interval for the access profile. This example sets the interval to 20 seconds

```
# Configuring router advertisement interval
# Option 1: Configuring router advertisement interval from global mode
[ ]
root@rtbrick:confd> set access access-profile access-profile1 protocol
router-advertisement interval 20
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring router advertisement interval from access-profile
mode
[ access access-profile access-profile1 ]
root@rtbrick:confd> set protocol router-advertisement interval 20
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring router advertisement interval from access-profile
protocol mode
[ access access-profile access-profile1 protocol ppp]
root@rtbrick:confd> set router-advertisement interval 20
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "access-profile:access-profile1": {
                    "protocol router-advertisement": {
                        "enable": true,
                        "interval": 30
                    }
            }
        ],
    }
}
```

5. AAA Profile Configuration

Subscriber management requires the mandatory configuration of an Authentication, Authorization, and Accounting (AAA) profile. Generally, AAA profile configuration involves the following:

- Holds the profile for Authentication, Authorization, and Accounting
- Controls selection of AAA type and order
- Assigns a RADIUS profile to the AAA profile (if RADIUS is used)

The way that the AAA profile configuration relates to all subscriber management configuration tasks is shown in Figure 5. Note that there can be more than one access interface configured for subscriber management and each interface can reference the same AAA profile.

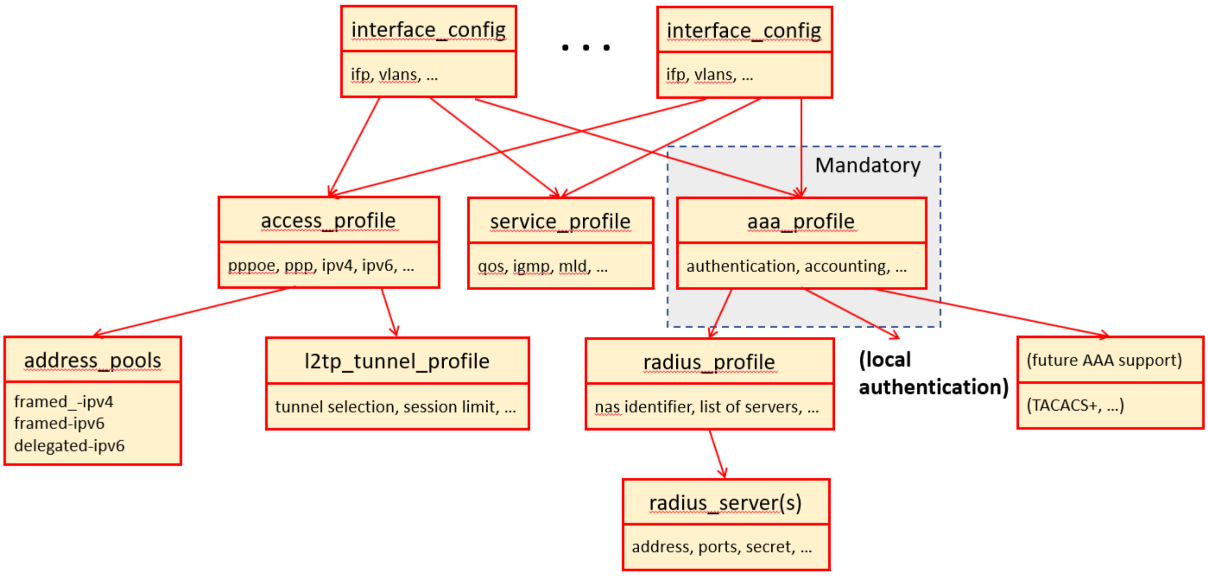

Figure 5. AAA Profile Configuration and Overall Subscriber Management

5.1. Configuring the AAA Profile Name

The basic first step in AAA profile configuration is to give the profile a name. As usual, there is more than one hierarchy level you can use to assign the name. All levels are shown in all examples, including the running configuration that results. User typing is shown in bold. This example assigns the name aaa-profile1 to the AAA profile.

```
#
# Configuring AAA Profile Name
# Option 1: Configuring aaa profile name from global mode
[ ]
root@rtbrick:confd> set access aaa-profile aaa-profile1 root@rtbrick:confd>
commit
Commit succeed

# Option 2: Configuring aaa profile name from access mode
[ access ]
root@rtbrick:confd> set aaa-profile aaa-profile1
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "aaa-profile:aaa-profile1": {
                }
            }
        ],
    }
}
```

5.2. Configuring AAA RADIUS Profile Name

The use of RADIUS is optional, but if you are using RADIUS as an authentication method, you must configure the AAA profile to use a RADIUS profile containing RADIUS server(s) details. This example assigns the name radius-profile1 to the AAA profile.

```
#
# Configuring AAA Radius Profile Name
# Option 1: Configuring aaa radius profile name from global mode
[ ]
root@rtbrick:confd> set access aaa-profile aaa-profile1 aaa-radius-profile
radius-profile1
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring aaa radius profile name from access mode
[ access ]
root@rtbrick:confd> set aaa-profile aaa-profile1 aaa-radius-profile radius-
profile1
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring aaa radius profile name from aaa profile mode
[ access aaa-profile aaa-profile1 ]
root@rtbrick:confd> set aaa-radius-profile radius-profile1
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        “access": [
            {
                "aaa-profile:aaa-profile1": {
                    "aaa-radius-profile": "radius-profile1"
                }
            }
        ],
    }
}
```

5.3. Configuring Session Timeout

You can configure a session timeout to terminate the RADIUS session and require the session to be re-established. This example sets a session timeout value of 3600 seconds (one hour) to the sessions. The default for session and idle timeout is 0 which means infinity.

```
#
# Configuring Session Timeout
# Option 1: Configuring session timeout from global mode
[ ]
root@rtbrick:confd> set access aaa-profile aaa-profile1 session-timeout 3600
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring session timeout from access mode
[ access ]
root@rtbrick:confd> set aaa-profile aaa-profile1 session-timeout 3600
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring session timeout from aaa profile mode
[ access aaa-profile aaa-profile1 ]
root@rtbrick:confd> set session-timeout 3600
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "aaa-profile:aaa-profile1": {
                    "aaa-radius-profile": "radius-profile1",
                    "session-timeout": 3600
                }
            }
        ],
    }
}
```

5.4. Configuring Idle Timeout

You can configure an idle timeout to terminate the RADIUS session when idle for a period and require the session to be re-established. This example sets an idle timeout value of 7200 seconds (two hours) to an idle session.

```
#
# Configuring Idle Timeout
# Option 1: Configuring idle timeout from global mode
[ ]
root@rtbrick:confd> set access aaa-profile aaa-profile1 idle-timeout 7200
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring idle timeout from access mode
[ access ]
root@rtbrick:confd> set aaa-profile aaa-profile1 idle-timeout 7200
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring idle timeout from aaa profile mode
[ access aaa-profile aaa-profile1 ]
root@rtbrick:confd> set idle-timeout 7200
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "aaa-profile:aaa-profile1": {
                    "aaa-radius-profile": "radius-profile1",
                    "session-timeout": 3600,
                    "idle-timeout": 7200
                }
            }
        ],
    }
}
```

5.5. Configuring Authentication Specifics

You can add specifics to authentication messages, such as the authentication order (first use local passwords, then use RADIUS) or a delimiter. Delimiters are required for domain authentication which is currently not supported.

5.5.1. Configurating Authentication Order

If you configure multiple authentication methods, you can configure the order in which they are used. This example configures RADIUS, but uses LOCAL password authentication first.

In general ordering works the way that the next method is only used if current method does not match or respond. If user is not found in local database we go to

RADIUS if configured as LOACL-RADIUS, but if local found but password does not match, we reject without going to RADIUS.

```
#
# Configuring Authentication Order
# Option 1: Configuring authentication order from global mode
[ ]
root@rtbrick:confd> set access aaa-profile aaa-profile1 authentication order
LOCAL
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring authentication order from access mode
[ access ]
root@rtbrick:confd> set aaa-profile aaa-profile1 authentication order LOCAL
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring authentication order from aaa profile mode
[ access aaa-profile aaa-profile1 ]
root@rtbrick:confd> set authentication order LOCAL
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "aaa-profile:aaa-profile1": {
                    "aaa-radius-profile": "radius-profile1",
                    "session-timeout": 3600,
                    "idle-timeout": 7200,
                    "authentication": {
                        "order": "LOCAL"
                    }
                }
            }
        ],
    }
}
```

5.5.2. Configuring Authentication Delimiter

The delimiter is used to get the domain part of a username and authenticate based on this only.

Example: user01@example.de

Delimiter configured as "@"

In this case, the domain part will be "example.de" (part after delimiter).

If domain authentication is supported, you can authenticate all users with same domain with same profile. Typical example is to setup a L2TP tunnel to the specified LNS server for all users with *@example.de.

```
# Configuring Authentication Delimiter
# Option 1: Configuring authentication delimiter from global mode
[ ]
root@rtbrick:confd> set access aaa-profile aaa-profile1 authentication
delimiter a
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring authentication delimiter from access mode
[ access ]
root@rtbrick:confd> set aaa-profile aaa-profile1 authentication delimiter a
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring authentication delimiter from aaa profile mode
[ access aaa-profile aaa-profile1 ]
root@rtbrick:confd> set authentication delimiter a
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "aaa-profile:aaa-profile1": {
                    "aaa-radius-profile": "radius-profile1",
                    "session-timeout": 3600,
                    "idle-timeout": 7200,
                    "authentication": {
                        "order": "LOCAL",
                        "delimiter": "a"
                    }
                }
            }
        ],
    }
}
```

5.6. Configuring Accounting Specifics

You can add specifics to accounting procedures, such as the accounting order (first use local session storage, then use RADIUS). Currently, only RADIUS server is supported. You can also configure an interim interval for the service-related accounting of services created during a login process.

5.6.1. Configuring Accounting Order

You can configure the order in which accounting locations are used. This example configures RADIUS accounting, but uses LOCAL storage of accounting information first.

```
#
# Configuring Accounting Order
# Option 1: Configuring accounting order from global mode
[ ]
root@rtbrick:confd> set access aaa-profile aaa-profile1 accounting order
LOCAL
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring accounting order from access mode
[ access ]
root@rtbrick:confd> set aaa-profile aaa-profile1 accounting order LOCAL
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring accounting order from aaa profile mode
[ access aaa-profile aaa-profile1 ]
root@rtbrick:confd> set accounting order LOCAL
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "aaa-profile:aaa-profile1": {
                    "aaa-radius-profile": "radius-profile1",
                    "session-timeout": 3600,
                    "idle-timeout": 7200,
                    "accounting": {
                        "order": "LOCAL"
                    }
                }
            }
        ],
    }
}
```

5.6.2. Configuring Accounting Interim Interval

You can configure an interim interval for the service-related accounting of services created during a login process. This example sets the interim interval to 3600 seconds.

```
#
# Configuring Accounting Interim Interval
# Option 1: Configuring accounting interim interval from global mode
[ ]
root@rtbrick:confd> set access aaa-profile aaa-profile1 accounting interim-
interval 10
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring accounting interim interval from access mode
[ access ]
root@rtbrick:confd> set aaa-profile aaa-profile1 accounting interim-interval
10
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring accounting interim interval from aaa profile mode
[ access aaa-profile aaa-profile1 ]
root@rtbrick:confd> set accounting interim-interval 10
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "aaa-profile:aaa-profile1": {
                    "aaa-radius-profile": "radius-profile1",
                    "session-timeout": 3600,
                    "idle-timeout": 7200,
                    "accounting": {
                        "order": "LOCAL",
                        "interim-interval": 10
                    }
                }
            }
        ],
    }
}
```

5.6.3. Configuring Session ID Format

You can configure the session identification format to be brief, default, or extensive. The default is more than brief but less than extensive. The more information kept; the more space required for accounting information. This example sets the session format to DEFAULT.

Accounting-Session-Id

The Accounting-Session-Id format is configurable and includes at least the subscriber-id to identify the corresponding subscriber.

The DEFAULT format includes also an unix timestamp to ensure thats the tuple of NAS-Identifier (for example, hostname) and Accounting-Session-Id is global unique to be usable as key in RADIUS databases.

Name	Format	Example
DEFAULT	<subscriber-id>.<timestamp>	72339069014639577.1551943760
BRIEF	<subscriber-id>	72339069014639577
EXTENSIVE	<subscriber-id>.<ifp>.<outervlan>.< inner-vlan>.<clientmac>.< session-id>.<timestamp>	72339069014639577.ifp-0/0/0.128.7.01:02:03:04:05:05.1.1551943 760

⚠ Currently only DEFAULT is supported.

```
#
# Configuring Accounting Session ID Format
# Option 1: Configuring accounting session id format from global mode
[ ]
root@rtbrick:confd> set access aaa-profile aaa-profile1 accounting session-
id-format DEFAULT
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring accounting session id format from access mode
[ access ]
root@rtbrick:confd> set aaa-profile aaa-profile1 accounting session-id-format
DEFAULT
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring accounting session id format from aaa profile mode
[ access aaa-profile aaa-profile1 ]
root@rtbrick:confd> set accounting session-id-format DEFAULT
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "aaa-profile:aaa-profile1": {
                    "aaa-radius-profile": "radius-profile1",
                    "session-timeout": 3600,
                    "idle-timeout": 7200,
                    "accounting": {
                        "order": "LOCAL",
                        "interim-interval": 10,
                        "session-id-format": "DEFAULT"
                    }
                }
            }
        ],
    }
}
```

6. RADIUS Profile Configuration

Subscriber management allows the optional configuration of a RADIUS profile.

Generally, RADIUS profile configuration involves the following:

- Holds the RADIUS server address and ports
- Holds the RADIUS secret and more

The way that the optional RADIUS profile configuration relates to all subscriber management configuration tasks is shown in Figure 6.

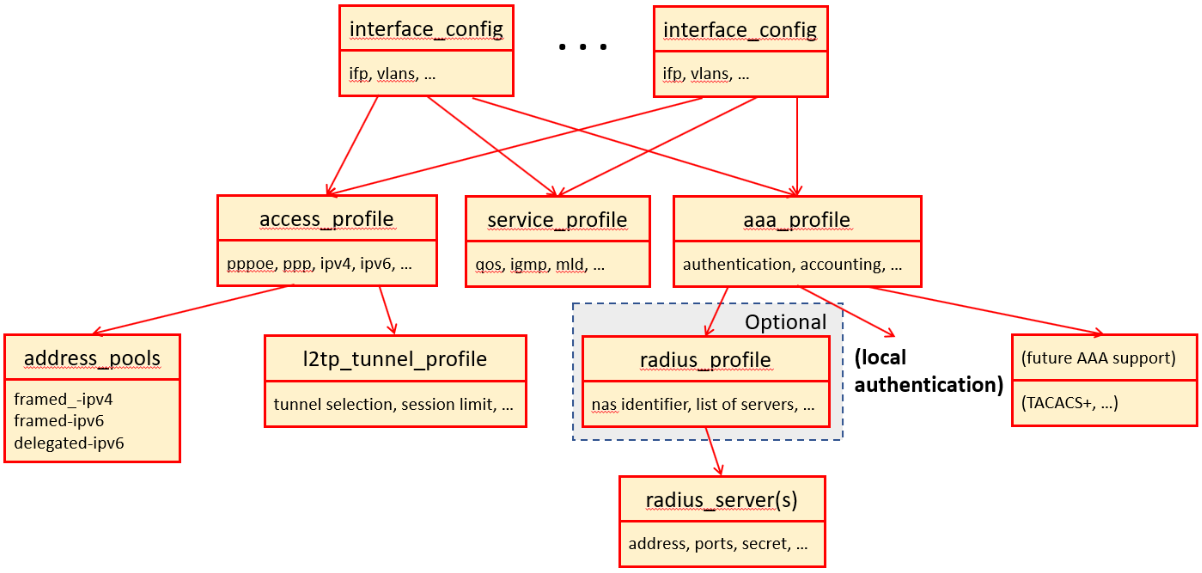

Figure 6. Optional RADIUS Profile Configuration and Overall Subscriber Management

6.1. Configuring the RADIUS Profile Name

The basic first step in RADIUS profile configuration is to give the profile a name. As usual, there is more than one hierarchy level you can use to assign the name. All levels are shown in all examples, including the running configuration that results. User typing is shown in bold. This example assigns the name radius-profile1 to the RADIUS profile.

```
#
# Creating Radius Profile Name
# Option 1: Creating radius profile name from global mode
[ ]
root@rtbrick:confd> set access radius-profile radius-profile1
root@rtbrick:confd> commit
Commit succeed

# Option 2: Creating radius profile name from global access mode
[ access ]
root@rtbrick:confd> set radius-profile radius-profile1
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "radius-profile:radius-profile1": {
            }
        ],
    }
}
```

6.2. Configuring the NAS Identifier

You configure the Network Access Server (NAS) identifier so that the source of the RADIUS access request can choose the correct policy for that request. This example sets the NAS identifier to BNG.

```
#
# Configuring NAS identifier
# Option 1: Configuring nas identifier from global mode
[ ]
root@rtbrick:confd> set access radius-profile radius-profile1 nas-identifier
BNG
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring nas identifier from global access mode
[ access ]
root@rtbrick:confd> set radius-profile radius-profile1 nas-identifier BNG
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring nas identifier from radius profile mode
[ access radius-profile radius-profile1 ]
root@rtbrick:confd> set nas-identifier BNG
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "radius-profile:radius-profile1": {
                    "nas-identifier": "BNG"
                }
            ],
        }
}
```

6.3. Configuring the NAS Port Format

The format of the 32 bit NAS-Port (attribute type 5) is configurable in the RADIUS profile using the configuration attribute nas_port_format.

Name	Bits	Values
DEFAULT	1:1:6:12:12	slot:subslot:port:vlan:vlan
SLOTS	6:2:6:12:6	slot:subslot:port:vlan:vlan

```
# Configuring NAS port format
# Option 1: Configuring nas port format from global mode
[ ]
root@rtbrick:confd> set access radius-profile radius-profile1 nas-port-format
DEFAULT root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring nas port format from global access mode
[ access ]
root@rtbrick:confd> set radius-profile radius-profile1 nas-port-format
DEFAULT root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring nas port format from radius profile mode
[ access radius-profile radius-profile1 ]
root@rtbrick:confd> set nas-port-format DEFAULT
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "radius-profile:radius-profile1": {
                "nas-identifier": "192.168.1.2",
                "nas-port-format": "DEFAULT"
            }
        ],
    }
}
```

6.4. Configuring the NAS Port Type

You can also configure the NAS port type, most commonly for Ethernet. This example sets the NAS port type to Ethernet.

```
#
# Configuring NAS port type
# Option 1: Configuring nas port type from global mode
[ ]
root@rtbrick:confd> set access radius-profile radius-profile1 nas-port-type
Ethernet root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring nas port type from global access mode
[ access ]
root@rtbrick:confd> set radius-profile radius-profile1 nas-port-type Ethernet
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring nas port type from radius profile mode
[ access radius-profile radius-profile1 ]
root@rtbrick:confd> set nas-port-type Ethernet
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "radius-profile:radius-profile1": {
                    "nas-identifier": "192.168.1.2",
                    "nas-port-format": "DEFAULT",
                    "nas-port-type": "Ethernet"
                }
          ],
    }
}
```

6.5. Configuring Authentication Specifics

If you configure an optional RADIUS profile, you can also configure the specifics of the authentication process. These specifics include the name of the RADIUS server and algorithm type used for authentication.

6.5.1. Configuring the RADIUS Server Name for Authentication

If you configure an optional RADIUS profile, then you must configure at least one RADIUS server name. Note that this name does not define any of the RADIUS server properties: the name simply provides a link between the RADIUS profile and the actual RADIUS server. This example assigns the name radius-server1 to the RADIUS profile radius-profile1 for authentication.

```
#
# Configuring radius server profile name
# Option 1: Configuring radius server profile name from global mode
[ ]
root@rtbrick:confd> set access radius-profile radius-profile1 authentication
radius-server-profile-name radius-server1
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring radius server profile name from global access mode
[ access ]
root@rtbrick:confd> set radius-profile radius-profile1 authentication
radius-server-profile-name radius-server1
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring nas port type from radius profile mode
[ access radius-profile radius-profile1 ]
root@rtbrick:confd> set authentication radius-server-profile-name radius-
server1 root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "radius-profile:radius-profile1": {
                    "nas-identifier": "192.168.1.2",
                    "nas-port-format": "DEFAULT",
                    "nas-port-type": "Ethernet",
                    "authentication": {
                        "radius-server-profile-name": [
                        "radius-server1"
                        ]
                }
            }
        ],
    }
}
```

6.5.2. Configuring the Authentication Algorithm Type

You can configure the algorithmic method used to determine which RADIUS server to use for authentication. The choices are:

- DIRECT (default): Requests are sent to the server following the one where the last request was sent. If the subscriber daemon receives no response from the server, requests are sent to the next server and so on.
- ROUND-ROBIN: Requests are sent to the server following the one where the last request was sent. If the subscriber daemon router receives no response

from the server, requests are sent to the next server and so on.

```
#
# Configuring algorithm type
# Option 1: Configuring algorithm type from global mode
[ ]
root@rtbrick:confd> set access radius-profile radius-profile1 authentication
algorithm-type DIRECT
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring algorithm type from global access mode
[ access ]
root@rtbrick:confd> set radius-profile radius-profile1 authentication
algorithm-type DIRECT
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring algorithm type from radius profile mode
[ access radius-profile radius-profile1 ]
root@rtbrick:confd> set authentication algorithm-type DIRECT
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "radius-profile:radius-profile1": {
                    "nas-identifier": "192.168.1.2",
                    "nas-port-format": "DEFAULT",
                    "nas-port-type": "Ethernet",
                    "authentication": {
                        "radius-server-profile-name": [
                        "radius-server1"
                        ],
                        "algorithm-type": "DIRECT"
                    }
                }
        ],
    }
}
```

6.6. Configuring Accounting Specifics

If you configure an optional RADIUS profile, you can also configure the specifics of the accounting process. These specifics include the name of the RADIUS server, the algorithm type used for accounting, and what should happen to the accounting process when receiving a reject or failure message.

6.6.1. Configuring the RADIUS Server Name for Accounting

If you configure an optional RADIUS profile for accounting, then you must configure *at least one* RADIUS server name. Note that this name does not define any of the RADIUS server properties: the name simply provides a link between the RADIUS profile and the actual accounting RADIUS server.

This example assigns the name radius-server1 to the RADIUS profile radius-profile1 for accounting.

```
#
# Configuring radius server profile name
# Option 1: Configuring radius server profile name from global mode
[ ]
root@rtbrick:confd> set access radius-profile radius-profile1 accounting
radius-server-profile-name radius-server1
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring radius server profile name from global access mode [
access ]
root@rtbrick:confd> set radius-profile radius-profile1 accounting
radius-server-profile-name radius-server1
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring nas port type from radius profile mode
[ access radius-profile radius-profile1 ]
root@rtbrick:confd> set accounting radius-server-profile-name radius-server1
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "radius-profile:radius-profile1": {
                "nas-identifier": "192.168.1.2",
                "nas-port-format": "DEFAULT",
                "nas-port-type": "Ethernet",
                "accounting": {
                    "radius-server-profile-name": [
                    "radius-server1"
                    ]
                }
            }
        ],
    }
}
```

6.6.2. Configuring the Accounting Algorithm Type

You can configure the algorithmic method used to determine which RADIUS server to use for accounting. The choices are direct and round-robin. This example configures the direct method (there is only one RADIUS server configured in the profile).

```
#
# Configuring radius server profile name
# Option 1: Configuring radius server profile name from global mode
[ ]
root@rtbrick:confd> set access radius-profile radius-profile1 accounting
radius-server-profile-name radius-server1
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring radius server profile name from global access mode [
access ]
root@rtbrick:confd> set radius-profile radius-profile1 accounting
radius-server-profile-name radius-server1
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring nas port type from radius profile mode
[ access radius-profile radius-profile1 ]
root@rtbrick:confd> set accounting radius-server-profile-name radius-server1
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "radius-profile:radius-profile1": {
                    "nas-identifier": "192.168.1.2",
                    "nas-port-format": "DEFAULT",
                    "nas-port-type": "Ethernet",
                    "accounting": {
                        "radius-server-profile-name": [
                        "radius-server1"
                        ],
                        "algorithm-type": "DIRECT"
                    }
               }
         ],
    }
}
```

6.6.3. Configuring Accounting Stop on Failure

You can configure RADIUS accounting to send an stop-on-failure message when the AAA server refuses a client request for access.

```
#
# Configuring accounting stop on failure
# Option 1: Configuring accounting stop on failure from global mode
[ ]
root@rtbrick:confd> set access radius-profile radius-profile1 accounting
stop-on-failure
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring accounting stop on failure from global access mode
[ access ]
root@rtbrick:confd> set radius-profile radius-profile1 accounting stop-on-
failure root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring accounting stop on failure from radius profile mode
[ access radius-profile radius-profile1 ]
root@rtbrick:confd> set accounting accounting stop on failure
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "radius-profile:radius-profile1": {
                    "nas-identifier": "192.168.1.2",
                    "nas-port-format": "DEFAULT",
                    "nas-port-type": "Ethernet",
                    "accounting": {
                        "radius-server-profile-name": [
                        "radius-server1"
                        ],
                        "algorithm-type": "DIRECT",
                        "stop-on-failure": true
                        }
                    }
            ],
    }
}
```

6.6.4. Configuring Accounting Stop on Reject

You can configure what should happen when an accounting server rejects accounting information. In most cases, accounting should stop to prevent accumulating information that the local device has no way of off-loading. This example configures radius-profile1 to stop gathering accounting information if a reject occurs.

```
#
# Configuring accounting stop on reject
# Option 1: Configuring accounting stop on reject from global mode
[ ]
root@rtbrick:confd> set access radius-profile radius-profile1 accounting
stop-on-reject
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring accounting stop on reject from global access mode
[ access ]
root@rtbrick:confd> set radius-profile radius-profile1 accounting stop-on-
reject root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring accounting stop on reject from radius profile mode
[ access radius-profile radius-profile1 ]
root@rtbrick:confd> set accounting accounting stop on reject
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "radius-profile:radius-profile1": {
                "nas-identifier": "192.168.1.2",
                "nas-port-format": "DEFAULT",
                "nas-port-type": "Ethernet",
                "accounting": {
                    "radius-server-profile-name": [
                    "radius-server1"
                    ],
                    "algorithm-type": "DIRECT",
                    "stop-on-failure": true,
                    "stop-on-reject": true
                }
            }
        ],
    }
}
```

6.6.5. Enabling Accounting Status-On-Off

A RADIUS Acct-Status-Type attribute is used by the RADIUS client (subscriber daemon) to mark the start of accounting (for example, upon booting) by specifying Accounting-On and to mark the end of accounting (for example, just before a scheduled reboot) by specifying Accounting-Off. This message is often used by RADIUS servers to automatically close/terminate all open accounting records/sessions for the corresponding client and therefore must not be sent to servers belonging to a profile which was already used/started for accounting.

By default, the assumption is that all servers referenced by a RADIUS profile share the same states and therefore accounting-on must be only sent to one of those before first accounting-start is sent.

RADIUS Accounting-On/Off messages are optional enabled in the RADIUS profile configuration using the accounting_on_off attribute. The additional attribute accounting_on_wait prevents any new session until accounting hast started meaning that Accounting-On response received.

Accounting-Off is currently not implemented.

Syntax

```
set access radius-profile <profile> accounting status-on-off
```

Example

```
set access radius-profile radius-profile-1 accounting status-on-off
```

6.6.6. Enabling Accounting Status-On-Wait

You can configure RADIUS accounting to send an accounting Stop message when a subscriber session has been successfully authenticated and authorized, but then fails before an accounting start message is sent. By default, an accounting stop message is sent only if an accounting start message has been exchanged with the accounting server.

Syntax

```
set access radius-profile <profile> accounting status-on-wait
```

Example

```
set access radius-profile radius-profile-1 accounting status-on-wait
```

7. RADIUS Server Configuration

Successful subscriber management AAA methods are often supplied by a RADIUS server, although there are cases where other forms of AAA, including local methods independent of networks availability, are appropriate.

RADIUS server configuration is a *dependent* step in subscriber management configuration. In other words, if you configure an optional RADIUS profile for AAA, then you must configure a RADIUS server to go along with it. So, RADIUS server configuration is dependent on RADIUS profile configuration.

Generally, RADIUS profile configuration involves the following:

- Server names, IP addresses, and ports
- Server RADIUS secret key
- Authentication or accounting details (modes)
- Retry counters and timers

The way that the dependent RADIUS server configuration relates to all subscriber management configuration tasks is shown in Figure 7.

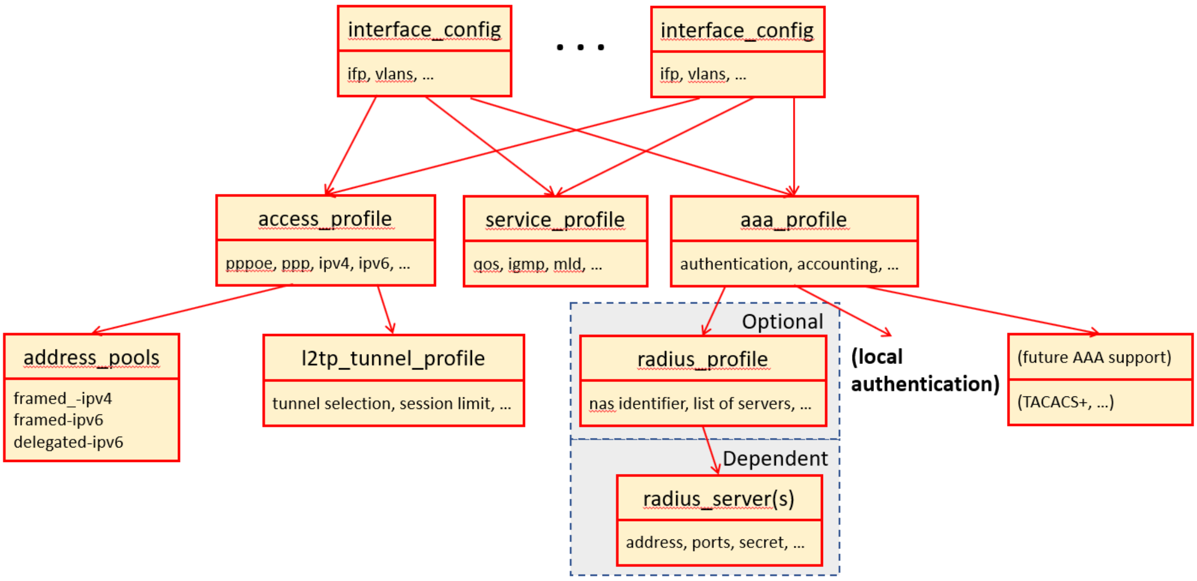

Figure 7. Dependent RADIUS Server Configuration and Overall Subscriber Management

7.1. Configuring the RADIUS Server Name

The first step is to give the RADIUS server a name. This name must match the name of the RADIUS server configured in the RADIUS profile if the RADIUS server is used for that profile. That is, the name forms the link between the RADIUS profile and a particular server. This example assigns the name rad-server1 to the RADIUS server.

```
#
# Setting Radius Server Name
# Option 1: Setting radius server name from global mode
[ ]
root@rtbrick:confd> set access radius-server rad-server1
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting radius server name from global access mode
[ access ]
root@rtbrick:confd> set radius-server rad-server1
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "radius-server:rad-server1": {
            }
        ],
    }
}
```

7.2. Configuring the RADIUS Server IPv4 Address

Do the following to assign an IPv4 address to the RADIUS server. This example assigns the IPv4 address 198.51.100.1 to the RADIUS server.

```
#
# Setting the Radius Server Ipv4 Address
# Option 1: Setting radius server ipv4 address from global mode
[ ]
root@rtbrick:confd> set access radius-server rad-server1 address
198.51.100.10 root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting radius server ipv4 address from access radius-server mode
[ access radius-server rad-server1 ]
root@rtbrick:confd> set address 198.51.100.10
[ access radius-server rad-server1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "radius-server:rad-server1": {
                    "address": "198.51.100.10",}
                }
            ],
        }
    }
}
```

7.3. Configuring the Source RADIUS IPv4 Address

Do the following to assign a source IPv4 address for packets sent by RADIUS server. This example assigns the source IPv4 address 198.51.100.1 to the RADIUS server.

```
#
# Setting Source IPv4 address for Radius packets
# Option 1: Setting source ipv4 address from global mode
[ ]
root@rtbrick:confd> set access radius-server rad-server1 source-address
198.51.1.2 root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting source ipv4 address from access radius-server mode
[ access radius-server rad-server1 ]
root@rtbrick:confd> set source-address 198.51.1.2
[ access radius-server rad-server1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "radius-server:rad-server1": {
                    "address": "198.51.100.1",
                    "source-address": "198.51.100.2"
                }
            }
        ],
    }
}
```

7.4. Configuring the RADIUS Secret Key

Do the following to assign a secret key to the RADIUS server. This example assigns the secret key test123 to the RADIUS server (this is not a good secret key).

```
#
# Setting the Secret Key for Radius Interactions
# Option 1: Setting secret key for radius from global mode
[ ]
root@rtbrick:confd> set access radius-server rad-server1 secret test123
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting radius server ipv4 address from access radius-server mode
[ access radius-server rad-server1 ]
root@rtbrick:confd> set secret test123
[ access radius-server rad-server1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "radius-server:rad-server1": {
                    "address": "198.51.100.1",
                    "source-address": "198.51.100.2",
                    "secret": "test123"
                }
            }
        ],
    }
}
```

7.5. Configuring the Source RADIUS IPv6 Address

Do the following to assign a source IPv6 address for packets sent by RADIUS server. This example assigns the source IPv6 address 2001:db8:200:1:1::2 to the RADIUS server.

```
#
# Setting Source IPv6 address for Radius packets
# Option 1: Setting source ipv6 address from global mode
[ ]
root@rtbrick:confd> set access radius-server rad-server1 source-ipv6-address
2001:db8:200:1:1::2
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting source ipv6 address from access radius-server mode
[ access radius-server rad-server1 ]

root@rtbrick:confd> set source-ipv6-address 200:1:1::2
[ access radius-server rad-server1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "radius-server:rad-server1": {
                    "address": "198.51.100.1",
                    "source-address": "198.51.100.2",
                    "ipv6-address": "2001:db8:100:1:1::2",
                    "source-ipv6-address": "2001:db8:200:1:1::2",
                    "secret": "test123"
                }
            }
        ],
    }
}
```

7.6. Configuring the RADIUS Authentication Mode

Do the following to use this RADIUS server for authentication. Basically, you set the enable condition for authentication to true.

```
#
# Enabling the radius authentication mode
# Option 1: Enable radius authentication from global mode
[ ]
root@rtbrick:confd> set access radius-server rad-server1 authentication
enable root@rtbrick:confd> commit
Commit succeed

# Option 2: Enable radius authentication from access radius-server mode
[ access radius-server rad-server1 ]
root@rtbrick:confd> set authentication enable
[ access radius-server rad-server1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Enable radius authentication from access radius-server
authentication mode
[ access radius-server rad-server1 authentication]
root@rtbrick:confd> set enable
[ access radius-server rad-server1 authentication ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "radius-server:rad-server1": {
                    "address": "198.51.100.1",
                    "source-address": "198.51.100.2",
                    "ipv6-address": "2001:db8:100:1:1::2",
                    "source-ipv6-address": "2001:db8:200:1:1::2",
                    "secret": "test123"
                    "authentication": {
                        "enable": true
                    }
                }
            }
        ],
    }
}
```

7.7. Configuring the RADIUS Authentication Port

Do the following to configure the RADIUS server port used for authentication. The default RADIUS port is 1812.

This example sets the RADIUS port to 1001.

```
#
# Setting radius server authentication port
# Option 1: set radius server authentication port from global mode
[ ]
root@rtbrick:confd> set access radius-server rad-server1 authentication port
1001 root@rtbrick:confd> commit
Commit succeed

# Option 2: Set radius authentication port from access radius-server mode
[ access radius-server rad-server1 ]
root@rtbrick:confd> set authentication port 1001
[ access radius-server rad-server1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Set radius server authentication port from access radius-server
authentication mode
[ access radius-server rad-server1 authentication]
root@rtbrick:confd> set port 1001
[ access radius-server rad-server1 authentication ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "radius-server:rad-server1": {
                    "address": "198.51.100.1",
                    "source-address": "198.51.100.2",
                    "ipv6-address": "2001:db8:100:1:1::2",
                    "source-ipv6-address": "2001:db8:200:1:1::2",
                    "secret": "test123",
                    "authentication": {
                        "enable": true,
                        "port": 1001
                    }
                }
            }
        ],
    }
}
```

7.8. Configuring the RADIUS Server Maximum Authentication Request Retries

Do the following to configure the maximum number of retries the RADIUS server sends authentication requests. This example sets maximum request retries to 10.

```
#
# Setting max retries for authentication request packets
# Option 1: Set authentication max retry count from global mode
[ ]
root@rtbrick:confd> set access radius-server rad-server1 authentication retry
10 root@rtbrick:confd> commit
Commit succeed

# Option 2: Set authentication max retry count from access radius-server mode
[ access radius-server rad-server1 ]
root@rtbrick:confd> set authentication retry 10
[ access radius-server rad-server1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed
# Option 3: Set authentication max retry count from access radius-server
authentication mode
[ access radius-server rad-server1 authentication]
root@rtbrick:confd> set retry 10
[ access radius-server rad-server1 authentication ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "radius-server:rad-server1": {
                    "address": "198.51.100.1",
                    "source-address": "198.51.100.2",
                    "ipv6-address": "2001:db8:100:1:1::2",
                    "source-ipv6-address": "2001:db8:200:1:1::2",
                    "secret": "test123",
                    "authentication": {
                        "enable": true,
                        "port": 1001,
                        "retry": 10
                    }
                }
            }
        ],
    }
}
```

7.9. Configuring the RADIUS Server Authentication Request Timeout Interval

Do the following to configure the amount of time the RADIUS server waits for a response to a request before timing out. This example sets request timeout interval to 30 seconds.

```
#
# Setting authentication request timeout period
# Option 1: Set authentication request timeout period from global mode
[ ]
root@rtbrick:confd> set access radius-server rad-server1 authentication
timeout 30 root@rtbrick:confd> commit
Commit succeed

# Option 2: Set authentication request timeout period from access radius-
server mode [ access radius-server rad-server1 ]
root@rtbrick:confd> set authentication timeout 30
[ access radius-server rad-server1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Set authentication request timeout period from access radius-
server authentication mode
[ access radius-server rad-server1 authentication]
root@rtbrick:confd> set timeout 30
[ access radius-server rad-server1 authentication ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "radius-server:rad-server1": {
                    "address": "198.51.100.1",
                    "source-address": "198.51.100.2",
                    "ipv6-address": "2001:db8:100:1:1::2",
                    "source-ipv6-address": "2001:db8:200:1:1::2",
                    "secret": "test123",
                    "authentication": {
                        "enable": true,
                        "port": 1001,
                        "retry": 10,
                        "timeout": 30
                    }
                }
            }
        ],
    }
}
```

7.10. Configuring the RADIUS Accounting Mode

Do the following to use this RADIUS server for accounting. Basically, you set the enable condition for accounting to true.

```
#
# Enabling global radius accounting mode
# Option 1: Enable global radius accounting from global mode
[ ]
root@rtbrick:confd> set access radius-server rad-server1 accounting enable
root@rtbrick:confd> commit
Commit succeed

# Option 2: Enable global radius accounting from access radius-server mode [
access radius-server rad-server1 ]
root@rtbrick:confd> set accounting enable
[ access radius-server rad-server1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "radius-server:rad-server1": {
                    "address": "198.51.100.1",
                    "source-address": "198.51.100.2",
                    "ipv6-address": "2001:db8:100:1:1::2",
                    "source-ipv6-address": "2001:db8:200:1:1::2",
                    "secret": "test123",
                    "accounting": {
                        "enable": true,
                    }
                }
            }
        ],
    }
}
```

7.11. Configuring the RADIUS Accounting Port

Do the following to configure the RADIUS server port used for accounting. This example sets the RADIUS port to 1500.

```
#
# Setting radius server accounting port
# Option 1: Set radius server accounting port from global mode
[ ]
root@rtbrick:confd> set access radius-server rad-server1 accounting port 1500
root@rtbrick:confd> commit
Commit succeed

# Option 2: Set radius server accounting port from access radius-server mode
[ access radius-server rad-server1 ]
root@rtbrick:confd> set accounting port 1500
[ access radius-server rad-server1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Set radius server accounting port from access radius-server
accounting mode
[ access radius-server rad-server1 accounting ]
root@rtbrick:confd> set port 1500
[ access radius-server rad-server1 accouting ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "radius-server:rad-server1": {
                    "address": "198.51.100.1",
                    "source-address": "198.51.100.2",
                    "ipv6-address": "2001:db8:100:1:1::2",
                    "source-ipv6-address": "2001:db8:200:1:1::2",
                    "secret": "test123",
                    "accounting": {
                        "enable": true,
                        "port": 1500,
                    }
                }
            }
        ],
    }
}
```

7.12. Configuring the RADIUS Server Maximum Accounting Request Retries

Do the following to configure the maximum number of retries the RADIUS server sends accounting requests. This example sets the maximum request retries to 10.

```
#
# Setting maximum retries for accounting request packets
# Option 1: Set max retries for accounting request packets from global mode
[ ]
root@rtbrick:confd> set access radius-server rad-server1 accounting retry 10
root@rtbrick:confd> commit
Commit succeed

# Option 2: Set max retries for accounting request packets from access
radius-server mode
[ access radius-server rad-server1 ]
root@rtbrick:confd> set accounting retry 10
[ access radius-server rad-server1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed
# Option 3: Set max retries for accounting request packets from access
radius-server accounting mode
[ access radius-server rad-server1 accounting ]
root@rtbrick:confd> set retry 10
[ access radius-server rad-server1 accounting ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "radius-server:rad-server1": {
                    "address": "198.51.100.1",
                    "source-address": "198.51.100.2",
                    "ipv6-address": "2001:db8:100:1:1::2",
                    "source-ipv6-address": "2001:db8:200:1:1::2",
                    "secret": "test123",
                    "accounting": {
                        "enable": true,
                        "port": 1500,
                        "retry": 10
                    }
                }
            }
        ],
    }
}
```

7.13. Configuring the RADIUS Server Accounting Request Timeout Interval

Do the following to configure the amount of time the RADIUS server waits for a response to an accounting request before timing out. This example sets the request timeout interval to 30 seconds.

```
#
# Setting accounting request timeout period
# Option 1: Set accounting request timeout period from global mode
[ ]
root@rtbrick:confd> set access radius-server rad-server1 accounting timeout
30 root@rtbrick:confd> commit
Commit succeed

# Option 2: Set accounting request timeout period from access radius-server
mode [ access radius-server rad-server1 ]
root@rtbrick:confd> set accounting timeout 30
[ access radius-server rad-server1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Set accounting request timeout period from access radius-server
accounting mode
[ access radius-server rad-server1 accounting ]
root@rtbrick:confd> set timeout 30
[ access radius-server rad-server1 accounting ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "radius-server:rad-server1": {
                    "address": "198.51.100.1",
                    "source-address": "198.51.100.2",
                    "ipv6-address": "2001:db8:100:1:1::2",
                    "source-ipv6-address": "2001:db8:200:1:1::2",
                    "secret": "test123",
                    "accounting": {
                        "enable": true,
                        "port": 1500,
                        "retry": 10,
                        "timeout": 30
                    }
                }
             }
        ],
    }
}
```

8. Service Profile Configuration

Service profile configuration is an optional step in subscriber management configuration. If you configure features such as QoS ingress or egress shapers, you must configure a service profile to hold these parameters. Generally, the service profile configuration involves the following:

- assigns service-related configurations for QoS, IGMP, and MLD
- allows dynamic overriding with RADIUS or through API

The way that the optional service profile configuration relates to all subscriber management configuration tasks is shown in Figure 8.

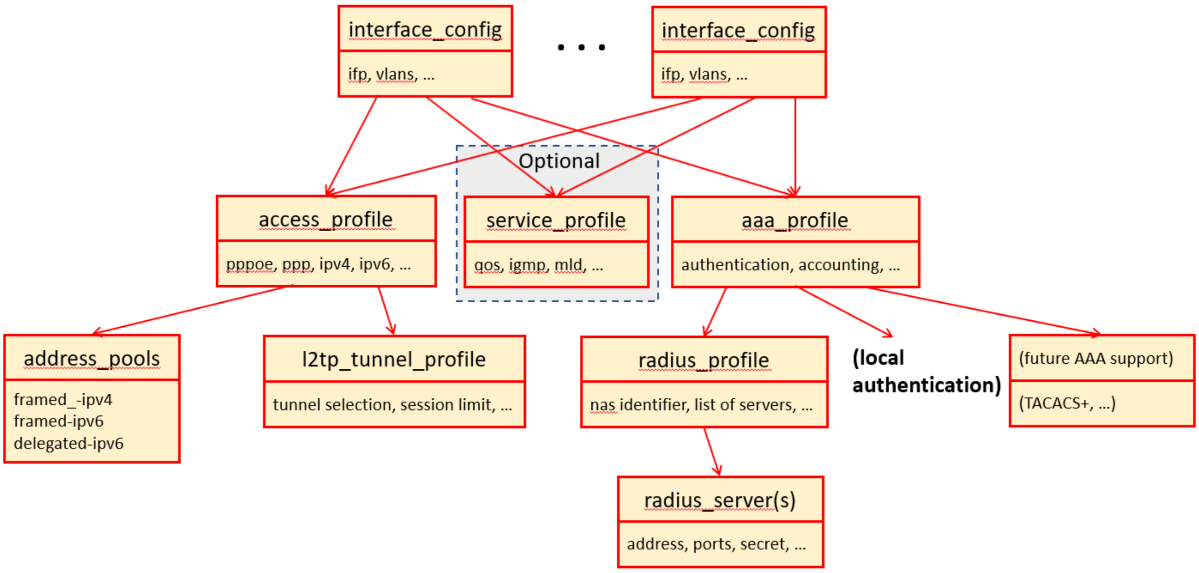

Figure 8. Optional Service Profile Configuration and Overall Subscriber Management

8.1. Configuring the Service Profile Name

The first step is to give the service profile a name. This example assigns the name service-profile1 to the service profile.

```
#
# Setting Service Profile Name
# Option 1: Setting service profile name from global mode
[ ]
root@rtbrick:confd> set access service-profile service-profile1
root@rtbrick:confd> commit
Commit succeed
# Option 2: Setting service profile name from global access mode [ access ]
root@rtbrick:confd> set service-profile service-profile1
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "service-profile:service-profile1": {
            }
        ],
    }
}
```

8.2. Configuring the QoS Ingress Shaper

Do the following to configure the shaper applied to ingress (inbound) traffic to guarantee a given QoS. This example sets the QoS ingress shaper to 5.

```
#
# Setting QoS Ingress Shaper
# Option 1: Setting qos ingress shaper from global mode
[ ]
root@rtbrick:confd> set access service-profile service-profile1 qos ingress-shaper 5 root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting qos ingress shaper from service-profile mode
[ access service-profile service-profile1 ]
root@rtbrick:confd> set qos ingress-shaper 5
[ access service-profile service-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Setting qos ingress shaper from service-profile qos mode
[ access service-profile service-profile1 qos ]
root@rtbrick:confd> set ingress-shaper 5
[ access service-profile service-profile1 qos ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "service-profile:service-profile1": {
                    "qos": {
                        "Ingress-shaper": 5
                       }
                   }
               }
          ],
    }
}
```

8.3. Configuring the QoS Egress Shaper

Do the following to configure the shaper applied to egress (outbound) traffic to guarantee a given QoS. This example sets the QoS egress shaper to 10.

```
#
# Setting QoS Egress Shaper
# Option 1: Setting qos Egress shaper from global mode
[ ]
root@rtbrick:confd> set access service-profile service-profile1 qos Egress-
shaper 10 root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting qos Egress shaper from service-profile mode
[ access service-profile service-profile1 ]
root@rtbrick:confd> set qos Egress-shaper 10
[ access service-profile service-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Setting qos Egress shaper from service-profile qos mode
[ access service-profile service-profile1 qos ]
root@rtbrick:confd> set Egress-shaper 10
[ access service-profile service-profile1 qos ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "service-profile:service-profile1": {
                "qos": {
                    "Ingress-shaper": 5,
                    "Egress-shaper": 10
                    }
                }
            }
        ],
    }
}
```

8.4. Enabling the IGMP Service

Do the following to enable the service profile to use the Internet Group Membership Protocol version 2 (IGMPv2) for multicast traffic. This example sets IGMPv2 indicator to true.

```
#
# Enabling IGMP Service
# Option 1: Enabling IGMP protocol from global mode
[ ]
root@rtbrick:confd> set access service-profile service-profile1 protocol
igmpv2 enable
root@rtbrick:confd> commit
Commit succeed

# Option 2: Enabling IGMP protocol from service-profile mode
[ access service-profile service-profile1 ]
root@rtbrick:confd> set protocol igmpv2 enable
[ access service-profile service-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Enabling IGMP protocol from service-profile protocol igmpv2 mode
[ access service-profile service-profile1 protocol igmpv2 ]
root@rtbrick:confd> set enable
[ access service-profile service-profile1 protocol igmpv2 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "service-profile:service-profile1": {
                    "protocol igmpv2": {
                        "enable": true
                    },
                    "qos": {
                        "Ingress-shaper": 5,
                        "Egress-shaper": 10
                    }
                }
            }
        ],
    }
}
```

8.5. Configuring the IGMP Query Interval

Do the following to configure the service profile’s IGMP query interval. This example sets IGMPv2 query interval to 10 seconds.

```
#
# Enabling IGMP Service
# Option 1: Setting IGMP query interval from global mode
[ ]
root@rtbrick:confd> set access service-profile service-profile1 protocol
igmpv2
query-interval 10
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting IGMP query interval from service-profile mode
[ access service-profile service-profile1 ]
root@rtbrick:confd> set protocol igmpv2 query-interval 10
[ access service-profile service-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Setting IGMP query interval from service-profile protocol igmpv2
mode [ access service-profile service-profile1 protocol igmpv2 ]
root@rtbrick:confd> set query-interval 10
[ access service-profile service-profile1 protocol igmpv2 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "service-profile:service-profile1": {
                    "protocol igmpv2": {
                        "enable": true,
                        "query-interval": 10
                    },
                "qos": {
                    "Ingress-shaper": 5,
                    "Egress-shaper": 10
                    }
                }
            }
        ],
    }
}
```

8.6. Enabling the MLD Service

Do the following to enable the service profile to use the Multicast Listener

Discovery (MLD) protocol for IPv6 multicast traffic. This example sets MLD indicator to true.

```
#
# Enabling MLD Service
# Option 1: Enabling mld protocol from global mode
[ ]
root@rtbrick:confd> set access service-profile service-profile1 protocol mld
enable root@rtbrick:confd> commit
Commit succeed

# Option 2: Enabling mld protocol from service-profile mode
[ access service-profile service-profile1 ]
root@rtbrick:confd> set protocol mld enable
[ access service-profile service-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Enabling IGMP protocol from service-profile protocol mld mode
[ access service-profile service-profile1 protocol mld ]
root@rtbrick:confd> set enable
[ access service-profile service-profile1 protocol mld ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
            "access": [
                {
                    "service-profile:service-profile1": {
                        "protocol igmpv2": {
                            "enable": true,
                            "query-interval": 10
                        },
                        "protocol mld": {
                            "enable": true
                        },
                        "qos": {
                            "Ingress-shaper": 5,
                            "Egress-shaper": 10
                        }
                    }
                }
            ],
    }
}
```

8.7. Configuring the MLD Query Interval

Do the following to configure the service profile’s MLD query interval. This example sets MLD query interval to 10 seconds.

```
#
#
# Enabling MLD query interval
# Option 1: Setting mld query interval from global mode
[ ]
root@rtbrick:confd> set access service-profile service-profile1 protocol mld
query-interval 10
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting mld query interval from service-profile mode
[ access service-profile service-profile1 ]
root@rtbrick:confd> set protocol mld query-interval 10
[ access service-profile service-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Setting mld query interval from service-profile protocol mld mode
[ access service-profile service-profile1 protocol mld ]
root@rtbrick:confd> set query-interval 10
[ access service-profile service-profile1 protocol mld ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "service-profile:service-profile1": {
                    "protocol igmpv2": {
                        "enable": true,
                        "query-interval": 10
                    },
                    "protocol mld": {
                        "enable": true,
                        "query-interval": 10
                    },
                    "qos": {
                        "Ingress-shaper": 5,
                        "Egress-shaper": 10
                    }
                }
            }
        ],
    }
}
```

9. L2TP Tunnel Profile Configuration

The Layer 2 Tunnel Protocol (L2TP) profile configuration is an optional step in subscriber management configuration. If you configure features using L2TP tunnels, you must configure a service profile to hold these parameters.

Generally, the L2TP profile configuration involves the following:

- Holding the L2TP LAC configuration
- Providing the tunnel selection algorithm
- Controlling the default session limit
- Establishing other parameters

The way that the optional L2TP tunnel profile configuration relates to all subscriber management configuration tasks is shown in Figure 9.

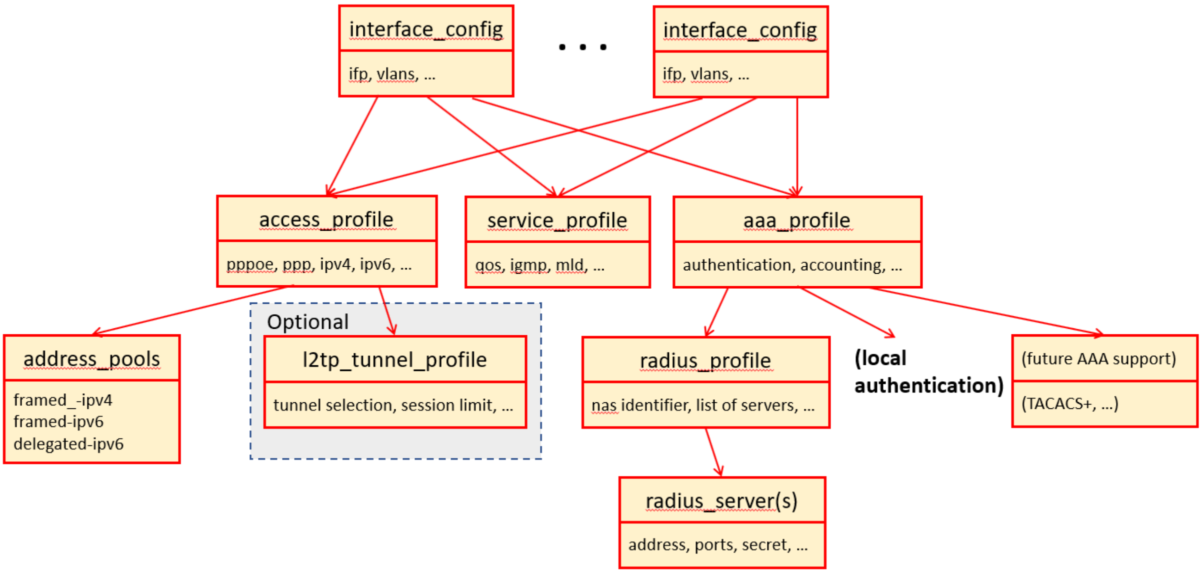

Figure 9. Optional L2TP Tunnel Profile Configuration and Overall Subscriber Management

9.1. Configuring the L2TP Tunnel Profile Name

The first step is to give the L2TP tunnel profile a name. This example assigns the name l2tp-profile1 to the service profile.

```
#
# Setting the L2TP profile name
# Option 1: Setting L2TP profile name from global mode
#
[ ]
root@rtbrick:confd> set access l2tp-profile l2tp-profile1
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting l2tp-profile name from global access mode
root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> set l2tp-profile l2tp-profile1
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration

{
    "running-configuration": {
        "access": [
            {
                "l2tp-profile:l2tp-profile1": {
                }
            }
        ]
    }
}
```

9.2. Configuring the L2TP VRF Instance Profile

The L2TP profile is associated with a VRF instance. Do the following to configure the VRF instance for the L2TP tunnel profile. This example sets the VRF instance to subscriber.

```
#
# Setting the L2TP vrf instance of the profile
# Option 1: Setting L2TP vrf instance of the profile from global mode
#
[ ]
root@rtbrick:confd> set access l2tp-profile l2tp-profile1 instance subscriber
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting l2tp vrf instance of the profile from global access mode
root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> set l2tp-profile l2tp-profile1 instance subscriber
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed
# Option 3: Setting l2tp vrf instance of the profile from l2tp-profile mode
root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> edit l2tp-profile l2tp-profile1
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> set instance subscriber
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "l2tp-profile:l2tp-profile1": {
                    "instance": "subscriber"
                }
            }
        ]
    }
}
```

9.3. Configuring the L2TP Profile Address Pool Name

The L2TP profile is associated with an address pool. Do the following to configure the address pool name for the L2TP tunnel profile. This example sets the pool name to l2tp-pool.

```
#
# Setting the L2TP pool name of the profile
# Option 1: Setting L2TP pool name of the profile from global mode
#
[ ]
root@rtbrick:confd> set access l2tp-profile l2tp-profile1 pool-name l2tp_pool
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting l2tp pool name of the profile from global access mode
root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> set l2tp-profile l2tp-profile1 pool-name l2tp_pool
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Setting l2tp pool name of the profile from l2tp-profile mode
root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> edit l2tp-profile l2tp-profile1
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> set pool-name l2tp_pool
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "l2tp-profile:l2tp-profile1": {
                    "pool-name": "l2tp_pool"
                }
            }
        ]
    }
}
```

9.4. Configuring the L2TP Profile Hello Timer

The L2TP profile is associated with a hello timer. Do the following to configure the hello interval for the L2TP tunnel profile. This example sets the timer interval to 60 seconds.

```
#
# Setting the L2TP hello timer of the profile
# Option 1: Setting L2TP hello timer of the profile from global mode
#
[ ]
root@rtbrick:confd> set access l2tp-profile l2tp-profile1 hello-interval 60
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting l2tp hello timer of the profile from global access mode
root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> set l2tp-profile l2tp-profile1 hello-interval 60
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Setting l2tp hello timer of the profile from l2tp-profile mode
root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> edit l2tp-profile l2tp-profile1
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> set hello-interval 60
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "l2tp-profile:l2tp-profile1": {
                    "hello-interval": "60"
                }
            }
        ]
    }
}
```

9.5. Configuring the L2TP Profile Congestion Window Size

The L2TP profile is associated with a window size to prevent buffer overflow. Do the following to configure the congestion window size for the L2TP tunnel profile. This example sets the congestion window size to 60.

```
#
# Setting the L2TP congestion window of the profile
# Option 1: Setting L2TP congestion window of the profile from global mode
#
[ ]
root@rtbrick:confd> set access l2tp-profile l2tp-profile1 congestion-window
60
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting l2tp congestion window of the profile from global access
mode root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> set l2tp-profile l2tp-profile1 congestion-window 60
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Setting l2tp congestion window of the profile from l2tp-profile
mode root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> edit l2tp-profile l2tp-profile1
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> set congestion-window 60
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "l2tp-profile:l2tp-profile1": {
                    "congestion-window": 60
                }
            }
        ]
    }
}
```

9.6. Configuring the L2TP Profile Hide Authentication Indicator

The L2TP profile is associated with an option to hide the authentication details. Do the following to configure the hiding of the authentication procedure for the L2TP tunnel profile. This example sets the hide authentication indicator to true.

```
#
# Setting the L2TP hide authentication of the profile
# Option 1: Setting L2TP hide authentication of the profile from global mode
#
[ ]
root@rtbrick:confd> set access l2tp-profile l2tp-profile1 hide-authentication
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting l2tp hide authentication of the profile from global
access mode root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> set l2tp-profile l2tp-profile1 hide-authentication
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Setting l2tp hide authentication of the profile from l2tp-profile
mode root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> edit l2tp-profile l2tp-profile1
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> set hide-authentication
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "l2tp-profile:l2tp-profile1": {
                    "hide-authentication": true
                }
            }
        ]
    }
}
```

9.7. Configuring the L2TP Profile Idle Timeout Interval

The L2TP profile is associated with an idle timeout for idle sessions. Do the following to configure the idle timeout interval for the L2TP tunnel profile. This example sets the idle timeout interval to 60 seconds.

```
#
# Setting the L2TP idle timeout of the profile
# Option 1: Setting L2TP idle timeout of the profile from global mode
#
[ ]
root@rtbrick:confd> set access l2tp-profile l2tp-profile1 idle-timeout-
interval 60
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting l2tp idle timeout of the profile from global access mode
root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> set l2tp-profile l2tp-profile1 idle-timeout-interval 60
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed
# Option 3: Setting l2tp idle timeout of the profile from l2tp-profile mode
root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> edit l2tp-profile l2tp-profile1
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> set idle-timeout-interval 60
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "l2tp-profile:l2tp-profile1": {
                    "idle-timeout-interval": "60"
                }
            }
        ]
    }
}
```

9.8. Configuring the L2TP Profile Receive Window Size

The L2TP profile is associated with a receive window size. Do the following to configure the receive window size for the L2TP tunnel profile. This example sets the receive window size to 60.

```
#
# Setting the L2TP receive window of the profile
# Option 1: Setting L2TP receive window of the profile from global mode
#
[ ]
root@rtbrick:confd> set access l2tp-profile l2tp-profile1 receive-window 60
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting l2tp receive window of the profile from global access
mode root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> set l2tp-profile l2tp-profile1 receive-window 60
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Setting l2tp receive window of the profile from l2tp-profile mode
root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> edit l2tp-profile l2tp-profile1
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> set receive-window 60
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
            "l2tp-profile:l2tp-profile1": {
                "receive-window": 60
             }
          }
      ]
   }
}
```

9.9. Configuring the L2TP Profile Request Retries

The L2TP profile is associated with a number of request retries. Do the following to configure the request retires number for the L2TP tunnel profile. This example sets the request retries number to 60.

```
#
# Setting the L2TP request retries of the profileprofile
# Option 1: Setting L2TP request retries of the profile from global mode
#
[ ]
root@rtbrick:confd> set access l2tp-profile l2tp-profile1 request-retries 60
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting l2tp request retries of the profile from global access
mode root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> set l2tp-profile l2tp-profile1 request-retries 60
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Setting l2tp request retries of the profile from l2tp-profile
mode root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> edit l2tp-profile l2tp-profile1
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> set request-retries 60
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "l2tp-profile:l2tp-profile1": {
                    "request-retries": 60
                  }
            }
        ]
    }
}
```

9.10. Configuring the L2TP Profile Retransmit Interval

The L2TP profile is associated with a retransmit interval. Do the following to configure the retransmit interval for the L2TP tunnel profile. This example sets the retransmit interval to 60 seconds.

```
#
# Setting the L2TP retransmit interval of the profile
# Option 1: Setting L2TP retransmit interval of the profile from global mode
#
[ ]
root@rtbrick:confd> set access l2tp-profile l2tp-profile1 retransmit-interval
60
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting l2tp retransmit interval of the profile from global
access mode root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> set l2tp-profile l2tp-profile1 retransmit-interval 60
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Setting l2tp retransmit interval of the profile from l2tp-profile
mode root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> edit l2tp-profile l2tp-profile1
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> set retransmit-interval 60
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "l2tp-profile:l2tp-profile1": {
                    "retransmit-interval": "60"
                }
            }
        ]
    }
}
```

9.11. Configuring the L2TP Profile Session Limit

The L2TP profile is associated with a session limit. Do the following to configure the session limit number for the L2TP tunnel profile. This example sets the session limit number to 60.

```
#
# Setting the L2TP session limit of the profile
# Option 1: Setting L2TP session limit of the profile from global mode
#
[ ]
root@rtbrick:confd> set access l2tp-profile l2tp-profile1 session-limit 60
[ ]
root@rtbrick:confd> commit
Commit succeed
# Option 2: Setting l2tp session limit of the profile from global access mode
root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> set l2tp-profile l2tp-profile1 session-limit 60
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed
# Option 3: Setting l2tp session limit of the profile from l2tp-profile mode
root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> edit l2tp-profile l2tp-profile1
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> set session-limit 60
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "l2tp-profile:l2tp-profile1": {
                    "session-limit": 60
                }
            }
        ]
    }
}
```

9.12. Configuring the L2TP Profile Selection Algorithm

The L2TP profile is associated with an algorithm used to select L2TP tunnels. This algorithm can be configured as BALANCED (choose tunnels in order to balance the numbers) or RANDOM (choose tunnels at random, regardless of how many are already allocated). This section shows how to configure both algorithms.

9.12.1. Configuring the BALANCED L2TP Profile Selection Algorithm

The L2TP profile is associated with an algorithm used to select L2TP tunnels. Do the following to configure the BALANCED selection algorithm for the L2TP tunnel profile.

```
#
# Setting the L2TP selection algorithm type balanced of the profile
# Option 1: Setting L2TP selection algorithm type balanced of the profile
from global mode

#
[ ]
root@rtbrick:confd> set access l2tp-profile l2tp-profile1 selection-algorithm
BALANCED
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting l2tp selection algorithm type balanced of the profile
from global access mode
root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> set l2tp-profile l2tp-profile1 selection-algorithm
BALANCED
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Setting l2tp selection algorithm type balanced of the profile
from
l2tp-profile mode
root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> edit l2tp-profile l2tp-profile1
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> set selection-algorithm BALANCED
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "l2tp-profile:l2tp-profile1": {
                    "selection-algorithm": "BALANCED"
                }
            }
        ]
    }
}
```

9.12.2. Configuring the RANDOM L2TP Profile Selection Algorithm

The L2TP profile is associated with an algorithm used to select L2TP tunnels. Do the following to configure the RANDOM selection algorithm for the L2TP tunnel profile.

```
#
# Setting the L2TP selection algorithm type random of the profile
# Option 1: Setting L2TP selection algorithm type random of the profile from
global mode
#
[ ]
root@rtbrick:confd> set access l2tp-profile l2tp-profile1 selection-algorithm
RANDOM
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting l2tp selection algorithm type random of the profile from
global access mode
root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> set l2tp-profile l2tp-profile1 selection-algorithm RANDOM
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Setting l2tp selection algorithm type random of the profile from
l2tp-profile mode
root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> edit l2tp-profile l2tp-profile1
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> set selection-algorithm RANDOM
[ access l2tp-profile l2tp-profile1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "l2tp-profile:l2tp-profile1": {
                    "selection-algorithm": "RANDOM"
                }
            }
        ]
    }
}
```

10. L2TP Address Pool Configuration

The Layer 2 Tunnel Protocol (L2TP) address pool configuration is an optional step in subscriber management configuration. If you configure features using L2TP tunnels, you must configure address pool parameters.

Generally, the L2TP address pool configuration involves the following:

- Holding the L2TP LAC and server configuration
- Providing the shared secret password
- Controlling the default session limit
- Establishing other address pool parameters

The way that the optional L2TP address pool configuration relates to all subscriber management configuration tasks is shown in Figure 10.

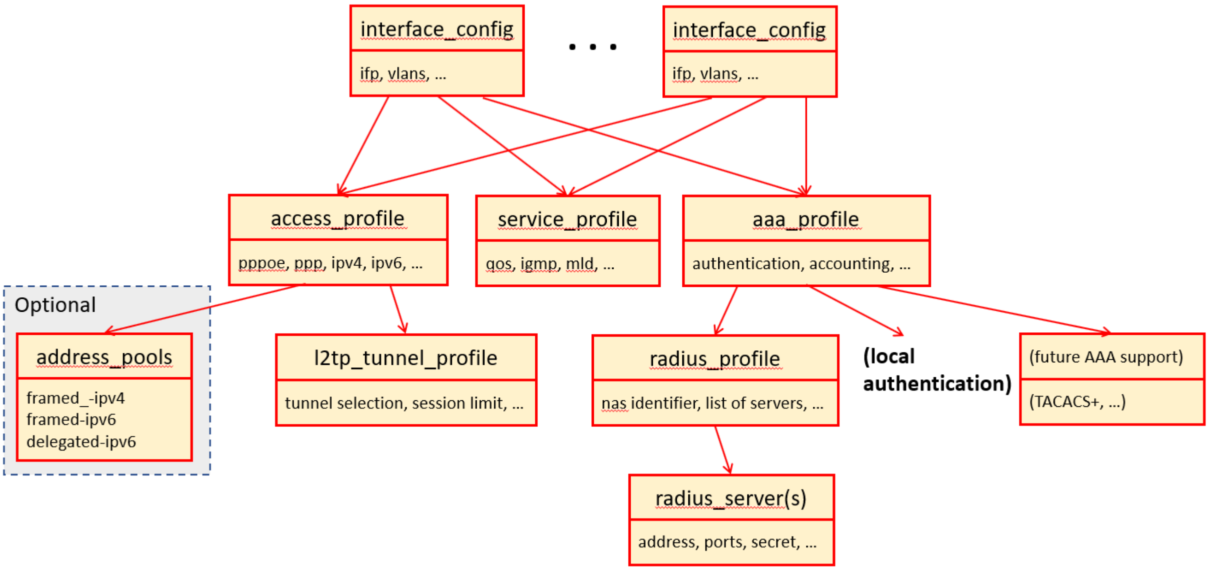

Figure 10. Optional L2TP Address Pool Configuration and Overall Subscriber Management

10.1. Configuring the L2TP Pool Client and Server Names

The first step is to give the L2TP pool client and server a name. This example assigns the name lac1 to the client and server1 to the server.

```
#
# Setting the L2TP pool configuration of client and server
# Option 1: Setting L2TP pool configuration of client and server from global
mode
#
[ ]
root@rtbrick:confd> set access l2tp-pool l2tp_pool client-name lac1 server-
name server1
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting l2tp pool configuration of client and server from global
access mode
root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> set l2tp-pool l2tp_pool client-name lac1 server-name
server1
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "l2tp-pool:l2tp_pool lac1 server1": {
                }
            }
        ]
    }
}
```

10.2. Configuring the L2TP Pool Client IPv4 Address

Do the following to assign an IPv4 address to the L2TP pool client. This example assigns the IPv4 address 10.1.1.6 to the L2TP pool client.

```
#
# Setting the L2TP pool client ipv4 address configuration
# Option 1: Setting L2TP pool client ipv4 address configuration from global
mode#
[ ]
root@rtbrick:confd> set access l2tp-pool l2tp_pool client-name lac1 server-
name server1 client-ipv4 11.1.1.6
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting l2tp pool client ipv4 address configuration from global
access mode
root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> set l2tp-pool l2tp_pool client-name lac1 server-name
server1 client-ipv4 11.1.1.6
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Setting l2tp pool client ipv4 address configuration from l2tp-
pool mode root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> edit l2tp-pool l2tp_pool client-name lac1 server-name
server1
[ access l2tp-pool l2tp_pool client-name lac1 server-name server1 ]
root@rtbrick:confd> set client-ipv4 11.1.1.6
[ access l2tp-pool l2tp_pool client-name lac1 server-name server1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "l2tp-pool:l2tp_pool lac1 server1": {
                    "client-ipv4": "10.1.1.6"
                }
            }
        ]
    }
}
```

10.3. Configuring the L2TP Pool Preference

Do the following to assign a pool preference value to the L2TP pool. This example assigns a preference value of 7 to the L2TP pool.

```
#
# Setting the L2TP pool preference value configuration
# Option 1: Setting L2TP pool preference value configuration from global mode
#
[ ]
root@rtbrick:confd> set access l2tp-pool l2tp_pool client-name lac1 server-
name server1 preference 7
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting l2tp pool preference value configuration from global
access mode root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> set l2tp-pool l2tp_pool client-name lac1 server-name
server1 preference 7
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Setting l2tp pool preference value configuration from l2tp-pool
mode root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> edit l2tp-pool l2tp_pool client-name lac1 server-name
server1
[ access l2tp-pool l2tp_pool client-name lac1 server-name server1 ]
root@rtbrick:confd> set preference 7
[ access l2tp-pool l2tp_pool client-name lac1 server-name server1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "l2tp-pool:l2tp_pool lac1 server1": {
                    "preference": 7
                }
            }
        ]
    }
}
```

10.4. Configuring the L2TP Pool Shared Secret Password

Do the following to assign a shared secret password to the L2TP pool. This example assigns a shared secret password of lac1 to the L2TP pool. (This is not a good shared secret password.)

```
#
# Setting the L2TP pool secret password configuration
# Option 1: Setting L2TP pool secret password configuration from global mode
#
[ ]
root@rtbrick:confd> set access l2tp-pool l2tp_pool client-name lac1 server-
name server1 secret lac1
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting l2tp pool secret password configuration from global
access mode root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> set l2tp-pool l2tp_pool client-name lac1 server-name
server1 secret lac1
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Setting l2tp pool secret password configuration from l2tp-pool
mode root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> edit l2tp-pool l2tp_pool client-name lac1 server-name
server1
[ access l2tp-pool l2tp_pool client-name lac1 server-name server1 ]
root@rtbrick:confd> set secret lac1
[ access l2tp-pool l2tp_pool client-name lac1 server-name server1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "l2tp-pool:l2tp_pool lac1 server1": {
                    "secret": "lac1"
                }
            }
        ]
    }
}
```

10.5. Configuring the L2TP Pool Server IPv4 Address

Do the following to assign an IPv4 address to the L2TP pool server. This example assigns the IPv4 address 10.0.0.3 to the L2TP pool server.

```
#
# Setting the L2TP pool server ipv4 address configuration
# Option 1: Setting L2TP pool server ipv4 address configuration from global
mode
#
[ ]
root@rtbrick:confd> set access l2tp-pool l2tp_pool client-name lac1 server-
name server1 server-ipv4 10.0.0.3
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting l2tp pool server ipv4 address configuration from global
access mode
root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> set l2tp-pool l2tp_pool client-name lac1 server-name
server1 server-ipv4 10.0.0.3
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Setting l2tp pool server ipv4 address configuration from l2tp-
pool mode root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> edit l2tp-pool l2tp_pool client-name lac1 server-name
server1
[ access l2tp-pool l2tp_pool client-name lac1 server-name server1 ]
root@rtbrick:confd> set server-ipv4 10.0.0.3
[ access l2tp-pool l2tp_pool client-name lac1 server-name server1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "l2tp-pool:l2tp_pool lac1 server1": {
                    "server-ipv4": "10.0.0.3"
                }
            }
        ]
    }
}
```

10.6. Configuring the L2TP Pool Session Limit

The L2TP pool is associated with a session limit. Do the following to configure the session limit number for the L2TP pool. This example sets the session limit number to 10.

```
#
# Setting the L2TP pool session limit value configuration
# Option 1: Setting L2TP pool session limit value configuration from global
mode
#
[ ]
root@rtbrick:confd> set access l2tp-pool l2tp_pool client-name lac1 server-
name server1 session-limit 10
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 2: Setting l2tp pool session limit value configuration from global
access mode
root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> set l2tp-pool l2tp_pool client-name lac1 server-name
server1 session-limit 10
[ access ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Option 3: Setting l2tp pool session limit value configuration from l2tp-
pool mode root@rtbrick:confd> edit access
[ access ]
root@rtbrick:confd> edit l2tp-pool l2tp_pool client-name lac1 server-name
server1
[ access l2tp-pool l2tp_pool client-name lac1 server-name server1 ]
root@rtbrick:confd> set session-limit 10
[ access l2tp-pool l2tp_pool client-name lac1 server-name server1 ]
root@rtbrick:confd> end
[ ]
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "l2tp-pool:l2tp_pool lac1 server1": {
                    "session-limit": 10
                }
            }
        ]
    }
}
```

11. User Profile Configuration

The user profile configuration is an optional step in subscriber management configuration. If you do not configure features using a RADIUS server, you must configure a user profile for local authentication.

Generally, the user profile configuration involves the following:

- Giving the user profile a name
- Providing the secret password for users

The way that the optional user profile (local authentication) configuration relates to all subscriber management configuration tasks is shown in Figure 11.

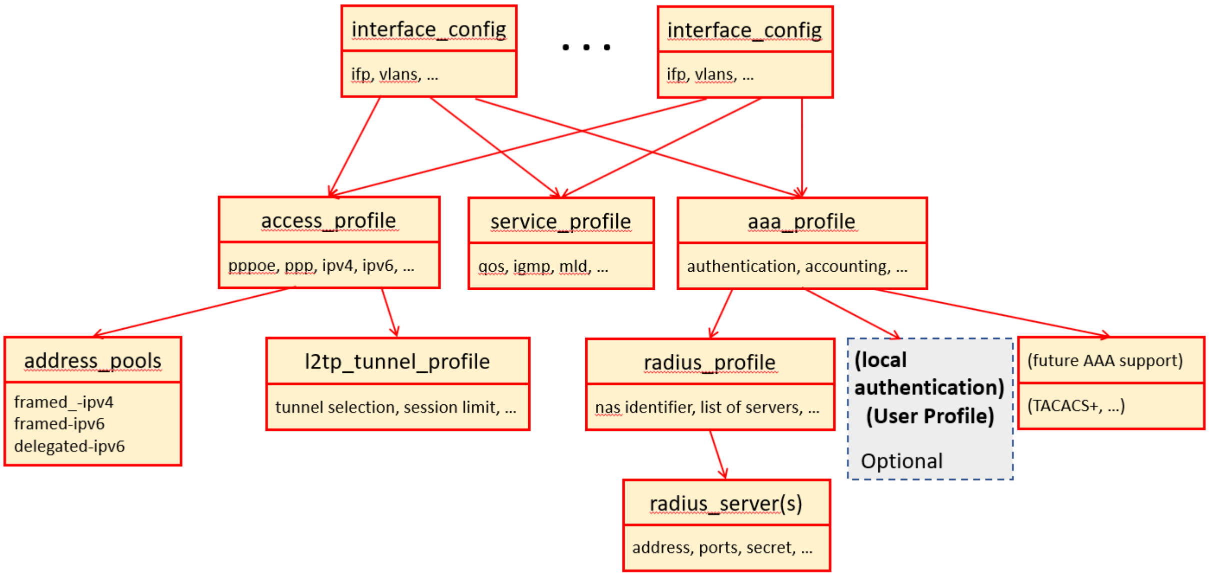

Figure 11. Optional User Profile Configuration and Overall Subscriber Management

11.1. Configuring the User Profile Name

The first step is to give the user profile a name. This example assigns the name user1@rtbrick.com to the user profile.

```
#
# Configuring User Profile Name
# Option 1: Configuring user profile name from global mode
[ ]
root@rtbrick:confd> set access user-profile user1@rtbrick.com
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring user profile name from access mode
[ access ]
root@rtbrick:confd> set user-profile user1@rtbrick.com
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "user-profile:user1@rtbrick.com": {
                }
            }
        ],
    }
}
```

11.2. Configuring the User Profile Password

Do the following to assign a password to the user profile. This example assigns a secret password of test123 to the user profile. (This is not a good password.)

```
#
# Configuring Password for the User Profile
# Option 1: Configuring password from global mode
[ ]
root@rtbrick:confd> set access user-profile user1@rtbrick.com password
test123
root@rtbrick:confd> commit
Commit succeed

# Option 2: Configuring password from access mode
[ access ]
root@rtbrick:confd> set user-profile user1@rtbrick.com password test123
root@rtbrick:confd> commit
Commit succeed

# Option 3: Configuring password from access user profile mode
[ access user-profile user1@rtbrick.com]
root@rtbrick:confd> set password test123
root@rtbrick:confd> commit
Commit succeed

# Running Configurations:
[ ]
root@rtbrick:confd> show running-configuration
{
    "running-configuration": {
        "access": [
            {
                "user-profile:user1@rtbrick.com": {
                    "password": "test123"
                }
            }
        ],
    }
}
```