NOKIA

NSP
Network Services Platform

Network Functions Manager - Packet (NFM-P)

Release 17.5

PCMP API Reference

3HE-12004-AAAB-TQZZA

Issue 1

May 2017

Legal notice

Contents

About this document

Purpose

This document describes the REST and Kafka API specifications for the PCMP and use cases for PCMP deployment in a managed network.

Document support

Customer documentation and product support URLs:

- Customer Documentation Welcome Page
- Technical support

How to comment

Documentation feedback

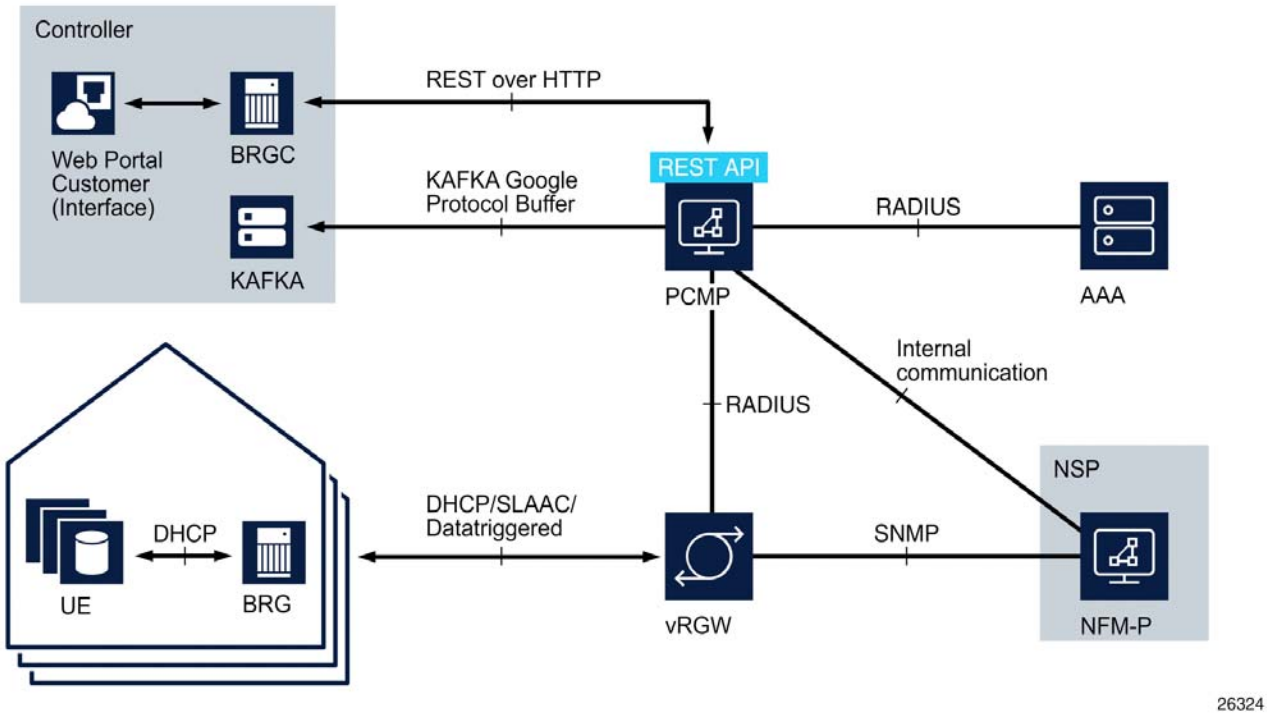
1 Introduction

1.1 Overview of network components

1.1.1 PCMP architecture

The following diagram displays the PCMP architecture. The components are divided between the “PCMP” systems that are designed by Nokia and “External systems” that are maintained by network operators/third parties and must use the API specifications described in this document.

Figure 1 PCMP architecture



1.1.2 Protocol Converter and Mediation Platform (PCMP)

The PMCP is an NFM-P component that interacts with the 7750 SR running as a vRGW. The PCMP translates REST API calls from a BRGC into the configuration required to modify the operational behavior of BRGs connected downstream from the 7750 SR. The

combination of the 7750 SR running as a vRGW and PCMP in a network is sometimes referred to as VRG or vRGW.

1.1.3 Virtual Residential Gateway (vRGW)

The vRGW is a designation of a 7750 SR BNG that hosts the L3 functions of a routed residential gateway. L3 functionality, such as address management, routing, IP connectivity, NAT, UPnP, firewalls, and parental controls are moved to the network and handled by the vRGW leaving the residential gateway to operate in bridge mode. This mode of operation allows for operator visibility of the connected devices on the home LAN. In a certain context, vRGW can refer to both the 7750 SR and PCMP as an integrated solution.

1.1.4 Bridged Residential Gateway (BRG)

The BRG is a residential gateway for which L3 functions are handled by a vRGW in the network. The BRG performs local switching of intra-home traffic that originates and terminates on devices within the home. The BRG is the logical representation of the home/residential subscriber in this mode of operation.

1.1.5 Bridged Residential Gateway Controller (BRGC)

The BRGC is the third party component that maintains a view of the active configuration for each BRG in the network (such as bandwidth, static IP address, and UPnP). The BRGC issues and responds to REST calls to/from the PCMP to apply the configuration to the vRGW in the network.

1.2 PCMP and BRGC REST API URLs

1.2.1 Standard URL format

The base PCMP REST API URL is the following:

http://*pcmp_server_ip*:*port*/api/v1/brgc/

Contact Nokia support for information about changing the default URL.

1.3 To specify the BRGC IP and provisioning URL

1.3.1 Purpose

Perform the following steps to specify the IP address of the BRGC server and the URL for the BRGC REST API.

1.3.2 Steps

1

Locate this section in the pcmpConverter.xml file:

<brgc serverIp="192.168.0.1"

serverPort="8081"

provBasePath="/api/v1/brgc/brgcprov/"/>

2

Set the serverIp to the BRGC IP address and set provBasePath to the required value for the BRGC REST API.

END OF STEPS

1.4 REST API methods used by PCMP

1.4.1 GET

The GET method retrieves information about the requested resource.

1.4.2 POST

The POST method creates a new resource entry or replaces the resource data with the data contained in the request. Fields not specified in the request are set to their default values (unless specified otherwise).

1.4.3 DELETE

The DELETE method deletes the resource specified in the request and all subordinate entries.

1.5 Response codes

1.5.1 Success response example and parameters

The following is an example of a successful GET operation:

```
{
  "status": 200,
  "message": "Success Message",
  "data": {
    "default_async_report_interval_in_sec": 300,
    "hardware_model": "sr_shelf_12Slot",
```

```
    "vendor": "Nokia",
    "software_version": "TiMOS-C-14.0.R4"
  }
}
```

Parameters:

status (integer): HTTP status code as defined in 1.5.3 “HTTP status codes” (p. 9).

message (string): A message stating the result of the operation.

data (object): The data contained in and returned by the request.

1.5.2 Failure response parameters

status (integer): HTTP status code as defined in 1.5.3 “HTTP status codes” (p. 9).

message (string): A message stating why the request failed and some information about what was invalid.

error_code: A predefined error code that corresponds to the specific failure. See 1.5.4 “Error codes” (p. 11).

1.5.3 HTTP status codes

Responses to HTTP requests contain the following status codes indicating the success or failure of the request.

Code	Description	Meaning
200	Success	GET, PUT, or DELETE method operation success.
201	Created	POST or PUT method operation success.
207	Partial	Success for some parameters on POST or PUT method operation.
400	Bad Request	The request is invalid and cannot be completed. The message states what is invalid about the request.

Code	Description	Meaning
401	Unauthorized	Authentication failure (missing or incorrect credentials).
403	Forbidden	Request refused. The error message states why the request was refused.
404	Not Found	The URI is invalid, the requested resource does not exist, or the request does not support the method type used.
500	Internal Server Error	The request cannot be completed due to an operational error. The message explains what went wrong.

1.5.4 Error codes

Error codes correspond to a specific failure type. Client application error codes range from 4000 to 4999. Server error codes range from 5000 to 5999.

Code	Error name and description
4000	ERROR_CODE_VALIDATION_FAILURE — The request has a validation error.
4101	ERROR_CODE_VALIDATION_MULTIPLE — The request has multiple validation errors.
4102	ERROR_CODE_VALIDATION_MANDATORY — A mandatory field was not specified.
4103	ERROR_CODE_VALIDATION_OUT_OF_RANGE — The parameter is out of range.
4104	ERROR_CODE_VALIDATION_WRONG_FORMAT — The parameter is in the wrong format.
5101	ERROR_CODE_SERVER_GENERAL — General exception from the server.
5102	ERROR_CODE_SERVER_RUNTIME — Runtime exception from the server, may indicate missing configuration in pcmpConverter.xml.

Code	Error name and description
5103	ERROR_CODE_SERVER_OBJ_NOT_FOUND — Cannot find object for REST GET.
5104	ERROR_CODE_AAA_SERVER_CONNECTION_FAILED — Cannot connect to AAA server on web portal application.
5105	Portal authentication is NOT enabled on PCMP.

1.6 Sequence flow diagrams

1.6.1 First device discovery

The following diagrams describe the first device discovery sequence flow.

Figure 2 BRG implicit authentication - first device in home

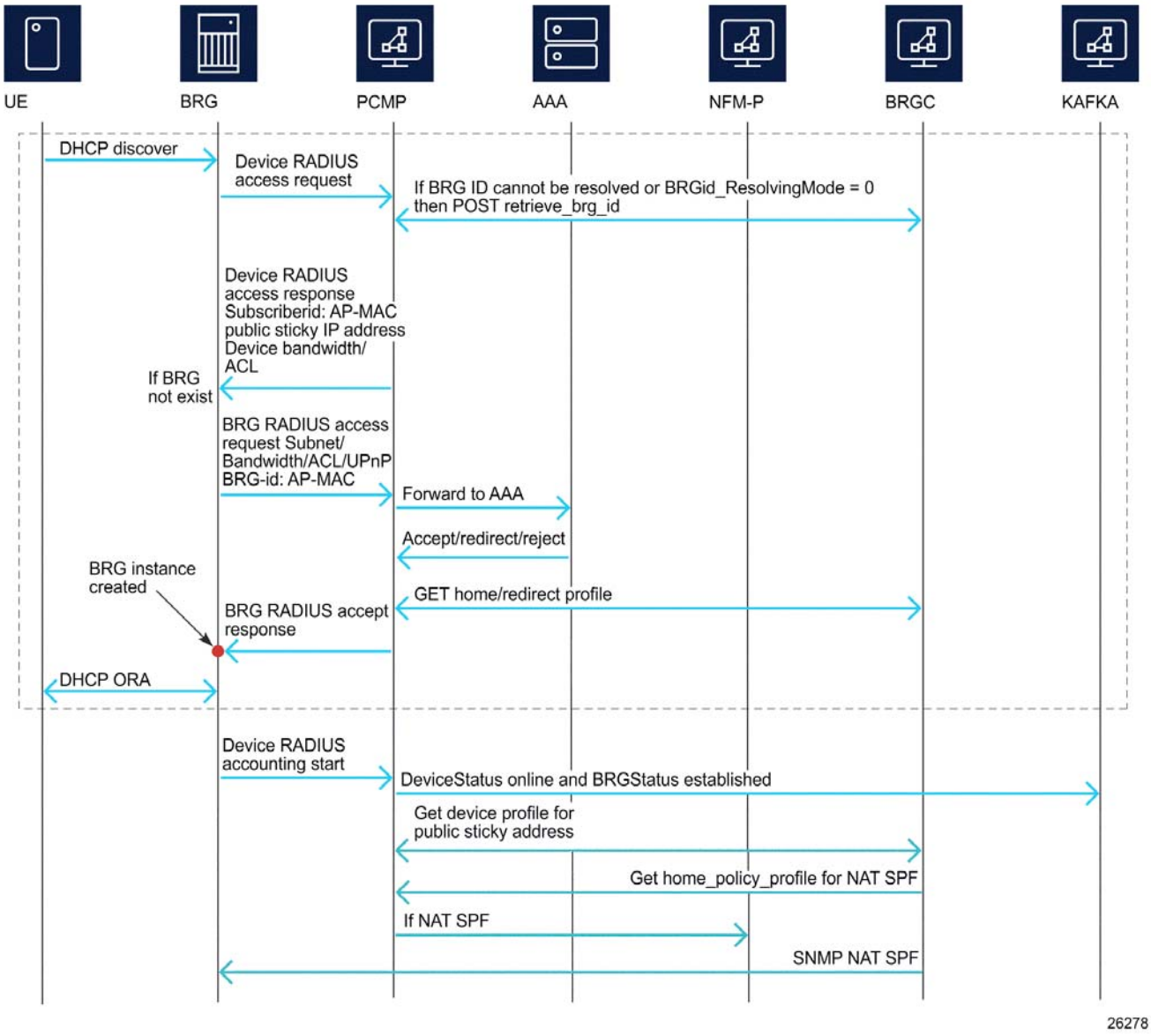
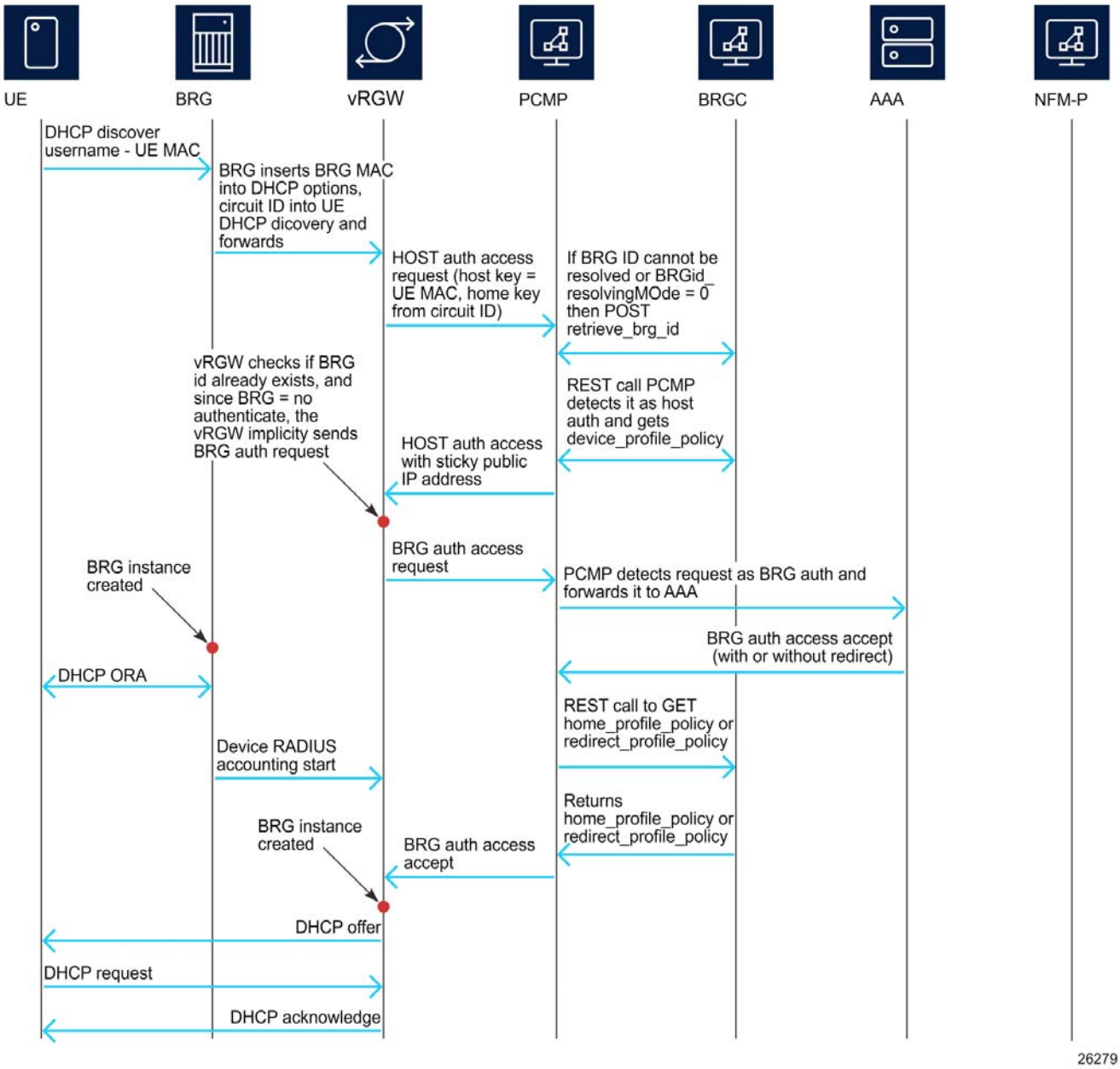


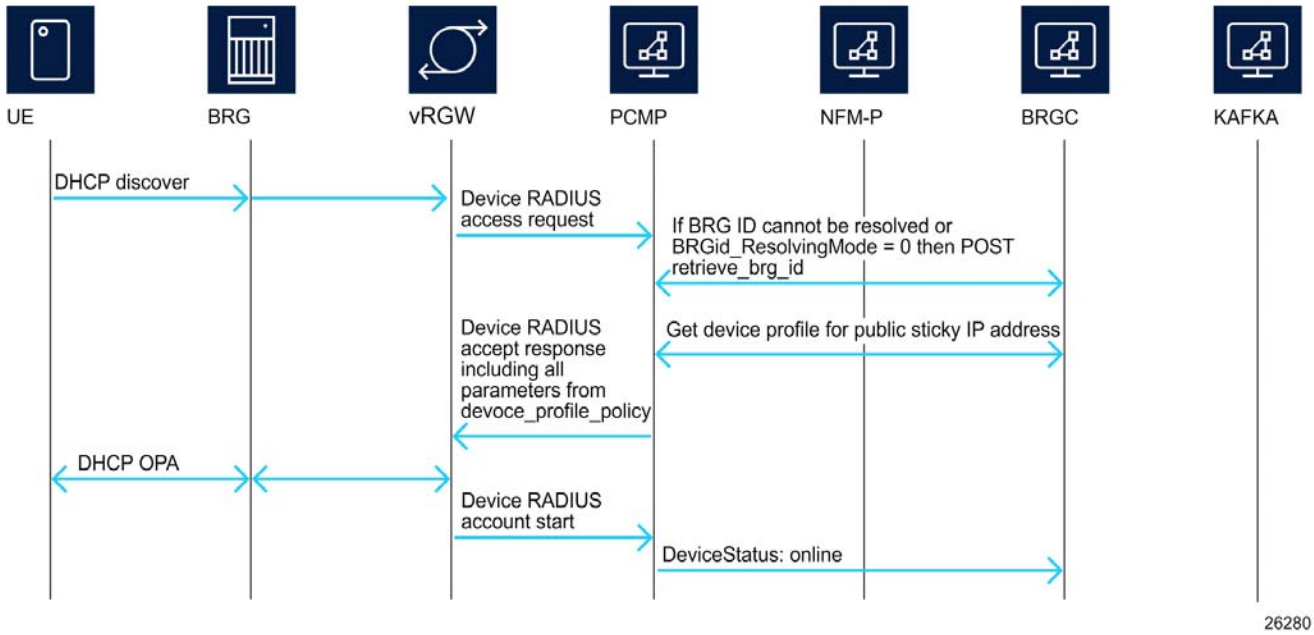
Figure 3 Detailed view



1.6.2 Second device discovery

The following diagram describes the second device discovery sequence flow.

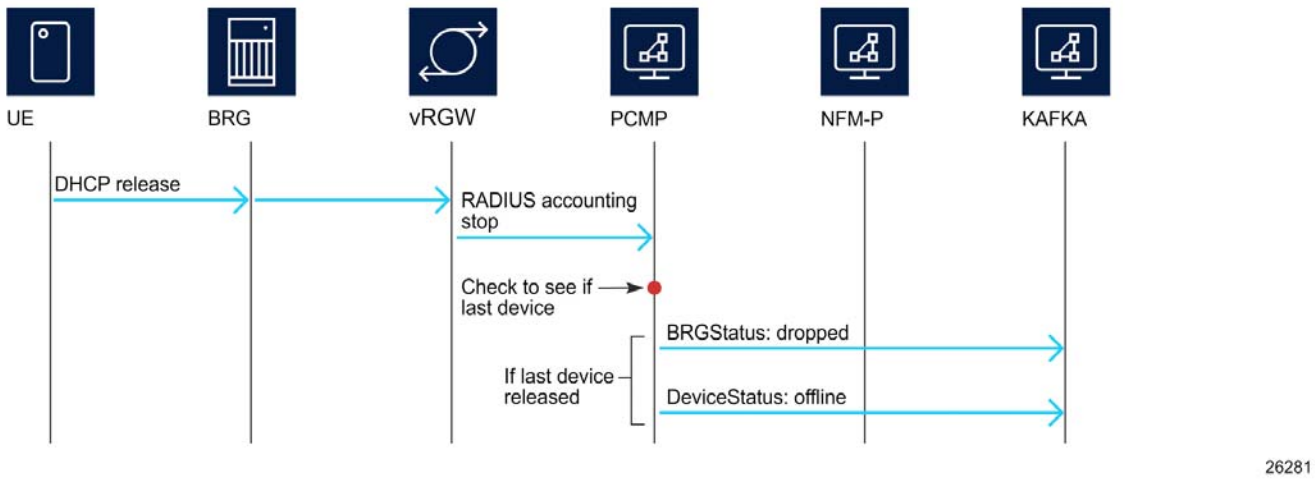
Figure 4 Second device discovery



1.6.3 Last device disconnected

The following diagram describes the last device disconnected sequence flow.

Figure 5 Last device disconnected



1.7 BRG ID resolution

1.7.1 Using Circuit ID

The PCMP attempts to resolve the BRG ID during Device Auth using the Circuit ID in format of:

XX:XX:XX:XX:XX:XX

or

XX-XX-XX-XX-XX-XX

or

XX:XX:XX:XX:XX:XX;<*ssid_name*>;<*y*>

or

XX:XX:XX:XX:XX:XX;<*ssid_name*>;<*y*>

where

y = network type (o = open, w = wired, s = secure)

1.7.2 Using BRGC query

If the BRG ID cannot be resolved using the Circuit ID, or if BRGID_ResolvingMode=0 in the pcmpConverter config file, the PCMP queries the BRGC for the BRG ID using the **/vcpe/retrieve_brg_id** API.

The POST body uses JSON to specify the identifier AVPs listed in the BRGIdLookupAVPIds section of the pcmpConverter file. If any AVPs are not available to the PCMP from the Device Access Request, they are omitted from the query. If all AVPs are not available, the PCMP sends an access reject.

See 3.5 “Retrieve BRG ID” (p. 67) for a description of the Retrieve BRG ID API.

See 3.5.3 “Supported AVPs ” (p. 67) for a list of the supported AVP keys and values that can be defined in the pcmpConverter config file.

1.8 API implementation specifics and recommendations

1.8.1 brg_id and device_mac format

The *brg_id* and *device_mac* URI parameters must be entered as a valid MAC address in uppercase without separators. For example, 1234567890AB.

1.8.2 brig_id length for sticky public IP address reservation

The brg_id must be 19 characters or less to provide enough room for a valid sticky public IP address reservation. The reservation will fail for brg_id values of 20 characters or more.

1.8.3 vlan_id

The *vlan_id* URI and object parameters are required parameters, however, the actual value is hard-coded in the PCMP configuration. The value passed via the API is ignored even though a value must be specified. A value of “1” is recommended for simplicity.

1.8.4 HTTP connection persistence

In order to reduce CPU usage, memory usage, network congestion, network latency and allow for higher PCMP scale throughput, Nokia recommends HTTP client reuse connection(s) to PCMP. In HTTPS or HTTP over SSL/TLS, persistent connections may reduce the number of costly SSL/TLS handshakes to establish security associations, in addition to the initial TCP connection set up.

1.8.5 LAN/LANv6 DNS for POST

With the DNS parameters on a POST of LAN/LANv6, if there are previous DNS overrides applied to the BRG instance and DNS is not explicitly specified on the POST, then the previous DNS overrides will persist.

1.9 Online REST API documentation

1.9.1 Accessing the API documentation

You can access the on-product HTML version of the PCMP REST API documentation by entering the following URL:

http://*pcmp_ip_address*:8080/pcmp/api-docs/

2 BRGC to PCMP REST API

2.1 Overview

2.1.1 Interface description

This interface is used by the BRGC to query BRG information and apply vRGW configuration to the network. All REST URIs are based on the URL described in 1.2.1 “Standard URL format” (p. 8). Append the URLs in this chapter to the base PCMP URL.

2.2 vRGW Configuration

2.2.1 /vrg

System configuration for the vRGW.

Supported methods: GET, POST

GET

Retrieve vRGW system configuration as data object **vrg**.

Success response example:

```
{
 "status": 200,
 "message": "Success Message",
 "data": {
  "hardware_model": "sr_shelf_7Slot",
  "vendor": "Nokia",
  "software_version": "TiMOS-C-14.0.R1",
  "default_async_report_interval_in_sec": 600
 }
}
```

POST

Configures the asynchronous reporting interval to specify how often the vRGW sends a status report to the BRGC in seconds.

Post request example:

```
{
 "default_async_report_interval_in_sec": 300
}
```

2.2.2 vrg parameters

hardware_model (string)—vRGW hardware model.

vendor (string)—vRGW vendor.

software_version (string)—vRGW software version in Major.Minor.patch format.

default_async_report_interval_in_sec (integer)—Default reporting interval (seconds) from vRGW to BRGC. Range of 300 to 900. Can be set to 0 for “no reporting”.

2.3 Software information

2.3.1 vrg/info

The running PCMP software version and build time.

Supported methods: GET

GET

Retrieves the running software version and build time as data object **info**.

Success response example:

```
{
  "status": 200,
  "message": "Success Message",
  "data": {
    "build_version": "1.1.307",
    "build_time": "2016-07-21 17:06:51 EDT"
  }
}
```

2.3.2 info parameters

build_version—the software release in Major.Minor.patch format.

build_time—the date and time that the PCMP was built.

2.4 Domain name/IP mapping

2.4.1 /vrg/domain_name_ip_mapping

Manages domain name to IP mappings. Mappings persist upon PCMP reboot.

Supported methods: GET, POST, DELETE

GET

Retrieves the current domain name to IP mappings as object **DomainNameIPAddressMappingList**.

Success response example:

```
{
  "status": 200,
  "message": "Success Message",
  "data": {
    "mappings": [
      {
        "domain_name": "www.test1.com",
        "ip": "10.1.1.1"
      },
      {
        "domain_name": "www.test2.com",
        "ip": "10.1.1.2"
      },
      {
        "domain_name": "www.test3.com",
        "ip": "10.1.1.3"
      },
      {
        "domain_name": "www.test4.com",
        "ip": "10.1.1.4"
      },
      {
        "domain_name": "www.test5.com",
        "ip": "10.1.1.5"
      },
      {
        "domain_name": "www.test6.com",
        "ip": "10.1.1.6"
      },
      {
        "domain_name": "www.test7.com",
        "ip": "10.1.1.7"
```

```
      },
      {
        "domain_name": "www.test8.com",
        "ip": "10.1.1.8"
      }
    ]
  }
}
```

POST

Appends the domain name to IP mappings provided using data object **DomainNameIPAddressMappingList** . If the entry exists in the mapping list, it updates it.

POST request example:

```
{
  "mappings": [
    {
      "domain_name": "www.test9.com",
      "ip": "10.1.1.9"
    }
  ]
}
```

DELETE

Deletes the domain name to IP mapping provided.

DELETE request example:

```
{
  "mappings": [
    {
      "domain_name": "www.test9.com",
      "ip": "10.1.1.9"
    }
  ]
}
```

2.4.2 DomainNameIPAddressMappingList parameters

mappings (array of object 2.4.3 “DomainNameIPAddressMapping parameters” (p. 23))—The domain name to IP address mappings.

2.4.3 DomainNameIPAddressMapping parameters

domain_name (string)—The domain name to map to the IP address.

ip (string)—The IP address to map to the domain name.

2.5 Home bandwidth override

2.5.1 /vrg/vcpe/{brg_id}/bandwidth

Provides the specific QoS configuration for a BRG to override the default settings on the vRGW. Bandwidth maximums vary depending on network configuration.

Supported methods: GET, POST

URI parameters

brg_id (string)—BRG ID.

GET

Retrieves the specific QoS settings for a BRG as the data object **bandwidth**.

Success response example:

```
{
 "status": 200,
 "message": "Success Message",
 "data": {
  "max_up": 30000000,
  "max_down": 40000000,
  "guaranty_up": -2,
  "guaranty_down": 20000000
 }
}
```

POST

Sets the QoS configuration for specific BRG using data object **bandwidth**.

POST request example:

```
{
 "max_up": 30000000,
 "max_down": 40000000,
 "guaranty_up": -2,
```

```
 "guaranty_down": 20000000
}
```

2.5.2 bandwidth parameters

Mandatory:

max_up (long integer)—Maximum upload bandwidth from BRG to vRGW in bits per second. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW. Maximum value of 2 000 000 000 000.

max_down (long integer)—Maximum download bandwidth from vRGW to BRG in bits per second. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW. Maximum value of 800 000 000 000.

Optional:

guaranty_up (long integer)—Guaranteed upload bandwidth in bits per second. Maximum value of 2 000 000 000 000. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW. **Not currently supported at the home level, will be ignored**.

guaranty_down (long integer)—Guaranteed download bandwidth in bits per second. Maximum value of 800 000 000 000. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW.

2.6 Object WAN

2.6.1 /vrg/vcpe/{brg_id}/wan

The WAN interface and firewall configuration of a BRG.

Supported methods: GET, POST

URI parameters

brg_id (string)—BRG ID.

GET

Retrieves the WAN parameters of a specific BRG as data object **WAN**.

Success response example:

```
{
 "status": 200,
```

```
 "message": "Success Message",
 "data": {
  "public_ip": "50.60.70.80",
  "filter_icmp_inbound": false,
  "dmz_enabled": true,
  "dmz_host": "192.168.1.99",
  "vlan_id": 1
 }
}
```

POST

Configure WAN parameters for a specific BRG using data object **WAN**.

```
{
 "filter_icmp_inbound": false,
 "dmz_enabled": true,
 "dmz_host": "192.168.1.99",
 "vlan_id": 1
}
```

2.6.2 wan parameters

Mandatory:

public_ip (returned by GET only) (string)—Public IP address of the BRG.

filter_icmp_inbound (Boolean)—Specifies whether inbound ICMP traffic is filtered by firewall policy. **Not supported for this release (always returns false)**.

dmz_enabled (Boolean)—Specifies whether DMZ is enabled.

dmz_host (string)—IPv4 address of the DMZ host. Is mandatory when dmz_enabled is TRUE.

Optional:

vlan_id (integer)—Indicates the VLAN that the DMZ host belongs to. **Not supported for this release.**

2.7 Public sticky IP address

2.7.1 /vrg/vcpe/{brg_id}/device/{device_mac}/publicip

Manages the reservation of the device public IP address in the vRGW DHCP server.

Supported methods: GET, POST, DELETE

Note: The brg_id must be 19 characters or less to provide enough room for a valid sticky public IP address reservation. The reservation will fail for brg_id values of 20 characters or more.

URI parameters

brg_id (string)—BRG ID.

device_mac (string)—MAC address of the target connected device.

GET

Retrieves the reserved sticky public IP settings for a connected device on the BRG as data object **sticky_public_ip**.

Success response example

```
{
 "status": 200,
 "message": "Success Message",
 "data": {
  "public_ip": "4.2.2.1"
 }
}
```

POST

Reserve a public IP for a connected device on the BRG using data object **sticky_public_ip**.

Success response example:

```
{
 "public_ip": "4.2.2.1"
}
```

DELETE

Deletes the reserved sticky public IP address for a connected device on the BRG.

Parameters: None

2.7.2 sticky_public_ip parameters

public_ip (string)—Specifies the public IPv4 address that is reserved/assigned.

2.8 LAN object settings

2.8.1 /vrg/vcpe/{brg_id}/vlan/{vlan_id}/lan

Manages the LAN configuration of the BRG.

Supported methods: GET, POST

URI parameters

brg_id (string)—BRG ID.

vlan_id (integer)—The VLAN ID assigned to the VLAN (see 1.8.3 “vlan_id” (p. 17)). Not supported for this release.

Conditions

A combination of ip_address and netmask determine the subnet for the home. The ip_address determines the inside gateway address for the home. The pool_start and max_clients determine the DHCP pool of inside IPv4 addresses that will be used for the devices in the home with private IPv4 addresses. The following conditions apply to POST operations:

1. If one of ip_address, netmask, pool_start, or max clients is specified, they are all mandatory and must all be specified. If none of these items are specified, the node default is used.
2. The ip_address, which is the inside gateway address for the home, must be pre-configured as a NAT inside address under the router on the node.
3. The resulting DHCP pool of addresses for devices must fall within the home subnet and must not include the home gateway ip_address.

GET

Retrieves the LAN configuration of a BRG as data object **lan**.

Success response example:

```
{
 "status": 200,
 "message": "Success Message",
```

```
 "data": {
  "dhcp_server": {
   "mac_binding": [
    {
     "mac_address": "1C0102030405",
     "ip_address": "192.168.1.105"
    },
    {
     "mac_address": "1C0102030406",
     "ip_address": "192.168.1.106"
    }
   ]
   "dhcp_lease_length": 86400,
   "max_clients": 50,
   "dns": [
    "8.8.8.8",
    "4.2.2.1"
   ],
   "pool_start": 100,
  }
  "ip_address": "192.168.1.1",
  "upnp_enabled": false,
  "domain": "lan",
  "netmask": "255.255.255.0",
  "static_ip": [
      {
        "mac_address": "1C01020304AD",
    "ip_address": "192.168.1.200"
      }
    ]
 }
}
```

POST

Overwrites the existing LAN configuration of the BRG using data object **lan**. If optional parameters are not specified, they are set to the default values based on the node configuration.

POST request example:

```
{
 "dhcp_server": {
  "mac_binding": [
   {
    "mac_address": "1C0102030405",
    "ip_address": "192.168.1.105"
```

```
    },
    {
     "mac_address": "1C0102030406",
     "ip_address": "192.168.1.106"
    }
   ],
   "max_clients": 50,
   "dhcp_lease_length": 86400,
   "dns": [
    "8.8.8.8",
    "4.2.2.1"
   ],
   "pool_start": 100,
  },
  "ip_address": "192.168.1.1",
  "upnp_enabled": false,
  "domain": "lan",
  "netmask": "255.255.255.0",
  "static_ip": [
     {
       "mac_address": "1C01020304AD",
    "ip_address": "192.168.1.200"
     }
  ]
}
```

2.8.2 lan parameters

Mandatory:

ip_address (string)—The subnet base IPv4 address.

netmask (string)—The IPv4 network masking of the subnet.

dhcp_server (object, see 2.8.3 “dhcp_server parameters” (p. 30))—Configuration of the DHCP server. All fields must be included.

Optional:

upnp_enabled (Boolean)—Specifies whether UPnP server is enabled.

domain (string)—The domain name advertised by the DHCP server to DHCP clients. Must be specified as “lan”.

static_ip (object, see 2.8.5 “static_ip parameters” (p. 30))—Static IP/MAC assignment.

2.8.3 dhcp_server parameters

Mandatory:

pool_start (integer)—The start of the IP pool as an offset. Must be a valid offset within the subnet defined by ip_address and netmask.

max_clients (integer)—The maximum number of IP leases available for this DHCP server from pool_start to pool_start + max_clients. Must be valid within the defined subnet.

dns (string array)—IP addresses of the DNS servers (primary/secondary) that the DHCP server uses. Up to two servers can be configured.

Note: If DNS entries are not specified, the existing DNS values are not overwritten during POST.

Optional:

dhcp_lease_length (integer)—Indicates the length of time in seconds of a DHCP lease. Range of 60 to 4294944000. Default of 86400.

mac_binding (object array, see 2.8.4 “mac_binding parameters” (p. 29))—Binding pairs of MAC and IPv4 addresses.

2.8.4 mac_binding parameters

mac_address (string)—The MAC address of the host.

ip_address (string)—The IPv4 address assigned to the MAC address.

2.8.5 static_ip parameters

mac_address (string)—The MAC address of the device.

ip_address (string)—The static IPv4 address assigned to the MAC address.

2.9 LANv6 object settings

2.9.1 /vrg/vcpe/{brg_id}/vlan/{vlan_id}/lanv6

Retrieves the V6 LAN settings of a specific vLAN of a specific vCPE.

Supported methods: GET, POST

URI parameters

brg_id (string)—BRG ID.

vlan_id (integer)—The VLAN ID assigned to the VLAN (see 1.8.3 “vlan_id” (p. 17)). Not supported for this release.

GET

Retrieves the LANv6 configuration of a BRG as data object **lanv6**.

Success response example 1:

```
{
  "status": 200,
  "message": "Success Message",
  "data": {
    "dns": [
      "33:0:0:0:0:0:0:1",
      "33:0:0:0:0:0:0:2"
    ],
    "slaacPrefix": "280:1:32:0:0:0:0:0/64",
    "slaacPoolName": ""
  }
}
```

Success response example 2:

```
{
  "status": 200,
  "message": "Success Message",
  "data": {
    "dns": [
      "33:0:0:0:0:0:0:1",
      "33:0:0:0:0:0:0:2"
    ],
    "slaacPrefix": "",
    "slaacPoolName": "slaacpool"
  }
}
```

POST

Normally, the post overwrites the existing configuration of the BRG using data object lanv6. If optional parameters are not specified, they are set to the default values based on the node configuration with the exception of DNS. With the DNS parameters on a

POST of LAN/LANv6, if there are previous DNS overrides applied to the BRG instance and DNS is not explicitly specified on the POST, then the previous DNS overrides will persist.

POST request example:

```
{
  "dns": [
    "34::33","35::33"
  ]
}
```

POST response example:

```
{
  "status": 201,
  "message": "Success Message"
}
```

2.9.2 lanv6 parameters

dns (string array)—IP addresses of the DNS servers (primary/secondary) that the DHCP server uses. Up to two servers can be configured.

slaacPrefix (string)—SLAAC prefix. Not currently supported on POST.

slaacPoolName (string)—SLAAC pool name. Not currently supported on POST.

2.10 Device list

2.10.1 /vrg/vcpe/{brg_id}/vlan/{vlan_id}/device

Retrieve the list of devices connected to a specific VLAN of a specific BRG.

Supported methods: GET

URI parameters

brg_id (string)—BRG ID.

vlan_id (integer)—The VLAN ID assigned to the VLAN (see 1.8.3 “vlan_id” (p. 17)).

GET

Retrieves the connected device list of a VLAN on the BRG as data object **connected_device**.

Note: The *hostname* parameter is not yet supported and returns “null”.

Success response example:


```
{
  "status": 200,
  "message": "Success Message",
  "data": [
    {
      "device_mac": "0A5052010101",
      "connection_status": 1,
      "connection_data": {
        "device_port_range": [
    {
     "start_port": 0,
     "end_port": 0
    }
   ],
        "hostname": null,
        "dhcp_lease_start": 1458755246,
        "dhcp_lease_length": 21600,
        "ip_address": "192.70.1.101",
        "ssid": "ss",
        "sticky_public_ip_flag": false,
    "host_allocation_type": 0
      }
    },
    {
      "device_mac": "0A5052020202",
      "connection_status": 1,
      "connection_data": {
        "device_port_range": [
    {
     "start_port": 0,
     "end_port": 0
    }
   ],
        "hostname": null,
        "dhcp_lease_start": 1458776747,
        "dhcp_lease_length": 21600,
        "ip_address": "192.70.1.102",
        "ssid": "ss",
        "sticky_public_ip_flag": false
      }
    }
  ]
}
```

2.10.2 connected_device parameters

device_mac (string)—MAC address of the connected device.

connection_status (integer)—Integer enum with values of 1, 2, or 3.

- 1 = connected device online
- 2 = connected device offline
- 3 = connected device idle

connection_data (object, see 2.10.3 “connection_data parameters” (p. 33))—The connection parameters of the device. If connection_status = 2, then the data field for this object returns as empty.

2.10.3 connection_data parameters

Optional:

dhcp_lease_start (integer)—Start time of the DHCP lease as the current UNIX epoch time.

dhcp_lease_length (integer)—Maximum length of the DHCP lease in seconds. A value of –1 indicates infinite length.

ip_address (string)—IP address of the device.

ssid (string)—WIFI SSID. Is not specified when the interface is wired.

sticky_public_ip_flag (Boolean)—Specifies whether the device IP is a sticky public IP.

host_allocation_type (enum)—Enumerated integer that indicates when the original context and IP address for a binding between a host MAC address and its IP address are allocated and released:

- notApplicable (0) : there is no such context; examples: IPv6 hosts using an IP address received from Radius (Framed-IP-Address).
- dynamic (1) : the original context is created and the IP address allocated after the binding is made; the original context is destroyed after the idle timer has expired; the IP address is released after the context timer has expired.
- static (2) : the original context is created/destroyed when this row is created/destroyed.
- stickyIpAddress (3) : the original context is released after its idle timer has expired; the IP address remains allocated after the lease timer has expired.
- offered (4) : the system has offered an IP address, but the host did not yet request it; this is a transitory state, that either disappears or changes into 'dynamic'.

- public (5) : the system received a public IPv4 address from a RADIUS server (Framed-IP-Address); stickiness, if any, of the association between the IPv4 address and the host MAC address is not guaranteed by this system but by the RADIUS server.

2.11 NAT Port forward

2.11.1 /vrg/vcpe/{brg_id}/ vlan/{vlanId}/portforward

Specifies the port forwarding configuration of a BRG.

Supported methods: GET, POST, DELETE

URI parameters

brg_id (string)—BRG ID.

vlan_id (integer)—The VLAN ID assigned to the VLAN (see 1.8.3 “vlan_id” (p. 17)).

GET

Retrieves the port forwarding settings of a BRG as data object **portforward**.

Success response example:

```
{
 "status": 200,
 "message": "Success Message",
 "data" : [
  {
   "name" : "",
   "protocol" : [
    255
   ],
   "wan_port_start": 2222,
   "wan_port_end": 2222,
   "ip_address": "192.168.1.123",
   "host_port_end": 22
   "host_port_start": 22,
  },
  {
   "name" : "",
   "protocol" : [
    6
   ],
   "wan_port_start": 8000,
```

```
   "wan_port_end": 8001,
   "ip_address": "192.168.1.124",
   "host_port_end": 81
   "host_port_start": 80,
  }
 ]
}
```

POST

Sets the port forwarding parameters a BRG using data object **portforward**.

POST request example:

```
[
 {
  "protocol" : [
   255
  ],
  "wan_port_start": 2222,
  "wan_port_end": 2222,
  "ip_address": "192.168.1.123",
  "host_port_start": 22,
  "host_port_end": 22
 },
 {
  "protocol" : [
   6
  ],
  "wan_port_start": 8000,
  "wan_port_end": 8001,
  "ip_address": "192.168.1.124",
  "host_port_start": 80,
  "host_port_end": 81
 }
]
```

DELETE

Delete the port forwarding settings of a BRG. Uses the URI parameters of the GET method.

DELETE response example:

```
{
 "status": 200,
```

```
 "message": "Success Message",
}
```

2.11.2 portforward parameters

Optional:

name (string)—A user defined name used to describe this entry. ^[a-zA-Z0-9]+$ with a 32 character maximum. Is not currently supported and is ignored for POST.

Mandatory:

protocol (integer array)—Protocol IDs are derived from standard IP protocol IDs as assigned by the IANA. Must be one of the following (cannot be empty):

- 6 = TCP
- 17 = UDP
- 255 = TCP and UDP

wan_port_start (integer)—Beginning of the range of ports to forward. Range of 1 to 65535.

wan_port_end (integer)—End of the range of ports to forward. Range of 1 to 65535.

ip_address (string)—IPv4 address of host to which to forward WAN traffic from:

- wan_port_start to wan_port end
- host_port_start to host_port_end

host_port_start (integer)—Beginning of the range of ports to forward to within the LAN. Range of 1 to 65535.

host_port_end (integer)—End of the range of ports to forward to within the LAN. Range of 1 to 65535.

2.12 DNS Bridge

2.12.1 /vrg/vcpe/{brg_id}/dnsbridge

The DNS configuration for all vLANs on the target BRG. All connected devices on the VLAN resolve DNS requests using the specified DNS server.

Supported methods: GET, POST

URI parameters

brg_id (string)—BRG ID.

GET

Retrieves the DNS bridging information from PCMP as data object **dns**.

Success response example:

```
{
 "status": 200,
 "message": "Success Message",
 "data": {
  "dns": [
   "4.2.2.1:53"
  ]
 }
}
```

POST

Sets the DNS Bridge for a specific vCPE as data object **dns**.

Note: If applying this parameter on the home profile, then a NAT classifier must be bound to the applicable NAT policies. Otherwise, the BRG instance might not be instantiated.

POST request example:

```
{
  "dns": [
    "disable"
  ]
}
```

2.12.2 dnsbridge parameters

dns (string array)—The IP address and port of the DNS servers returned as *ip_address*:*port*. For POST, the value of dns can be:

- string of *ip_address*:*port*, where port is optional and must be 53 if provided
- empty string or null removes any previous override and default to the CLI config
- “disable” to disable the DNS bridge

2.13 Home ACL

2.13.1 /vrg/vcpe/{brg_id}/acl

Manages the ACL configuration at the BRG level.

Supported methods: GET, POST, DELETE

URI parameters

brg_id (string)—BRG ID.

GET

Retrieves the ACL override settings of the BRG as data object **ACL**.

Success response example:

```
{
 "status": 200,
 "message": "Success Message",
 "data": {
  "acl_mode": 2,
  "portal_url": "http://portal.com/my/redirect/portal/?param=
value"
 }
}
```

POST

Set the ACL override parameters for the BRG using data object **ACL**.

POST request example:

```
{
 "acl_mode": 2,
 "portal_url": "http://69.252.80.100"
}
```

DELETE

Delete the BRG level ACL configuration.

DELETE response example:

```
{
 "status": 200,
```

```
 "message": "Success Message",
}
```

2.13.2 ACL parameters

Mandatory:

acl_mode (integer)—Integer enum values of 0, 1, or 2:

- 0 = Allowed
- 1 = Blocked
- 2 = Portal

Optional:

portal_url (string)—URL for redirection portal when acl_mode = 2. This field is ignored when the acl_mode is 1 or 0. The URL should be of the form:

http(s)://<*host*>/<*path*><*querystrings*>

where

host is an IPv4 address or DNS name

path is the full path to a resource (can be empty)

querystrings are valid URL query strings

Can be null if acl_mode is 0 (Allowed) or 1 (Blocked).

2.14 Default device bandwidth

2.14.1 /vrg/vcpe/{brg_id}/devicebandwidth

Specifies the default bandwidth configuration for any device in a BRG, when there is no bandwidth specified at the device level.

Supported methods: GET, POST

URI parameters

brg_id (string)—BRG ID.

GET

Retrieve the default bandwidth settings as data object **bandwidth**.

Success response example:

```
{
 "status": 200,
 "message": "Success Message",
 "data": {
  "max_up": 30000000,
  "max_down": 40000000,
  "guaranty_up":10000000,
  "guaranty_down": 20000000
 }
}
```

POST

Set the default bandwidth settings on the BRG using data object **bandwidth**.

POST request example:

```
{
 "max_up": 30000000,
 "max_down": 40000000,
 "guaranty_up":10000000,
 "guaranty_down": 20000000
}
```

2.14.2 bandwidth parameters

Mandatory:

max_up (long integer)—Maximum upload bandwidth from BRG to vRGW in bits per second. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW. Maximum value of 2 000 000 000 000.

max_down (long integer)—Maximum download bandwidth from vRGW to BRG in bits per second. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW. Maximum value of 2 000 000 000 000.

Optional:

guaranty_up (long integer)—Guaranteed upload bandwidth in bits per second. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW. Maximum value of 2 000 000 000 000.

guaranty_down (long integer)—Guaranteed download bandwidth in bits per second. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW. Maximum value of 2 000 000 000 000.

2.15 Device ACL

2.15.1 /vrg/vcpe/{brg_id}/device/{device_mac}/acl

Manages the ACL configuration for a specific connected device on the BRG.

Supported methods: GET, POST, DELETE

URI parameters

brg_id (string)—BRG ID.

device_mac (string)—The MAC address of the target connected device.

GET

Retrieves the ACL override settings for a connected device on a VLAN of the BRG as data object **ACL**.

Success response example:

```
{
 "status": 200,
 "message": "Success Message",
 "data": {
  "acl_mode": 2,
  "portal_url": "http://portal.com/my/redirect/portal/?param=
value"
 }
}
```

POST

Set the ACL override settings for a connected device on a VLAN of the BRG using data object **ACL**.

POST request example:

```
{
 "acl_mode": 2,
 "portal_url": "http://69.252.80.100"
}
```

DELETE

Delete the ACL settings on a specific connected device on a specific BRG.

DELETE response example:

```
{
 "status": 200,
 "message": "Success Message",
}
```

2.15.2 ACL parameters

Mandatory:

acl_mode (integer)—Integer enum values of 0, 1, or 2:

- 0 = Allowed
- 1 = Blocked
- 2 = Portal

Optional:

portal_url (string)—URL for redirection portal when acl_mode = 2. This field is ignored when the acl_mode is 1 or 0. The URL should be of the form:

http(s)://<*host*>/<*path*><*querystrings*>

where

host is an IPv4 address or DNS name

path is the full path to a resource (can be empty)

querystrings are valid URL query strings

Can be null if acl_mode is 0 (Allowed) or 1 (Blocked).

2.16 Device bandwidth

2.16.1 /vrg/vcpe/{brg_id}/device/{device_mac}/bandwidth

Allows the configuration of QoS parameters for a specific connected device on the BRG.

URI parameters

brg_id (string)—BRG ID.

device_mac (string)—The MAC address of the target connected device.

GET

Retrieve the bandwidth settings for a specific connected device as data object **bandwidth**.

Success response example:

```
{
 "status": 200,
 "message": "Success Message",
 "data": {
  "max_up": 30000000,
  "max_down": 40000000,
  "guaranty_up":10000000,
  "guaranty_down": 20000000
 }
}
```

POST

Configure the bandwidth settings of a device on the BRG using data object **bandwidth**.

POST request example:

```
{
 "max_up": 30000000,
 "max_down": 40000000,
 "guaranty_up":10000000,
 "guaranty_down": 20000000
}
```

2.16.2 bandwidth parameters

Mandatory:

max_up (long integer)—Maximum upload bandwidth from BRG to vRGW in bits per second. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW. Maximum value of 2 000 000 000 000.

max_down (long integer)—Maximum download bandwidth from vRGW to BRG in bits per second. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW. Maximum value of 2 000 000 000 000.

Optional:

guaranty_up (long integer)—Guaranteed upload bandwidth in bits per second. Maximum value of 2 000 000 000 000.

guaranty_down (long integer)—Guaranteed download bandwidth in bits per second. Maximum value of 2 000 000 000 000.

2.17 DHCP stream disable

2.17.1 /vrg/vcpe/{brg_id}/device/{device_mac}/dhcpstream

Sets the value of the DHCP stream disabled switch of a specific device.

Supported methods: POST

URI parameters

brg_id (string)—BRG ID.

device_mac (string)—The MAC address of the target connected device.

POST

Configure the bandwidth settings of a device on the BRG using data object **dhcpstream**.

POST request example:

```
{
  "dhcp_stream_disabled": false
}
```

2.17.2 dhcpstream parameters

dhcp_stream_disabled (boolean)—Indicates if device DHCP streaming is disabled.

2.18 BRG cleanup

2.18.1 /vrg/vcpe/{brg_id}

Clears the BRG and removes the associated resources.

Supported methods: DELETE

URI parameters

brg_id (string)—BRG ID.

DELETE

Typically used by the BRGC when detects it BRG tunnel creation through a new vRGW in order to clean up resources on the old vRGW.

Delete response example:

```
{
 "status": 200,
 "message": "Success Message"
}
```

2.19 Subscriber profile

2.19.1 /vrg/vcpe/{brg_id}/SubscriberProfile

Sets and retrieves the subscriber profile assigned to the BRG.

Supported methods: GET, POST

URI parameters

brg_id (string)—BRG ID.

GET

Retrieves the subscriber profile string as sub_prof_str..

Success response example:

```
{
 "status": 200,
 "message": "Success Message",
 "data": {
```

```
  "sub_prof_str": "sub_prof_50_1"
 }
}
```

POST

Sets the subscriber profile string as sub_prof_str.

POST request example:

```
{
 "sub_prof_str": "sub_prof_50_2"
}
```

2.19.2 subscriber_profile parameters

sub_prof_str (string)—the subscriber profile string.

2.20 Get AAA servers

2.20.1 /vrg/getAAAServer

Retrieves AAA server information.

Supported methods: GET

GET

Retrieves the IP address of each AAA server and the current redundancy state as data object **PortalAuthAAAServerInfo**.

Success response example:

```
{
  "status": 200,
  "message": "Success Message",
  "data": {
    "primary_aaa_server": "10.1.8.79",
    "secondary_aaa_server": "10.1.8.80",
    "redundancy_state": 1
  }
}
```

PortalAuthAAAServerInfo parameters

primary_aaa_server (string)—IP address of the primary AAA server.

secondary_aaa_server (string)—IP address of the secondary AAA server.

redundancy_state (integer)—Redundancy State, where 1 = Primary Active, 2 = Secondary Active, 5 = Redundancy Disabled.

GET response messages

HTTP Status Code	Reason
200	Success
400	Bad request
500	Internal server error

2.21 Set active AAA server

2.21.1 /vrg/setActiveAAAServer

Sets the active AAA server.

Supported methods: POST

POST

Sets the active AAA server.

POST request example:

```
{
  "ip": "10.1.8.79",
  "port": "string"
}
```

2.21.2 setActiveAAAServer parameters

ip (string)—IP address of the active AAA server.

port (string)—Port of the active AAA server.

3 PCMP to BRGC REST API

3.1 Overview

3.1.1 Interface description

This interface is used by the PCMP retrieve policy profiles from the BRGC. The BRGC must be configured with these REST API URLs available to the PCMP. The base URL for the BRGC can be specified by following the steps in 1.3 “To specify the BRGC IP and provisioning URL” (p. 8). The URLs described in this chapter are appended to the base BRGC URL.

3.2 Home policy profile

3.2.1 /vcpe/{brg_id}/home_policy_profile

Retrieves the home gateway policy for the BRG.

Supported methods: GET

URI parameters

brg_id (string)—BRG ID.

Conditions

A combination of ip_address and netmask determine the subnet for the home. The ip_address determines the inside gateway address for the home. The pool_start and max_clients determine the DHCP pool of inside IPv4 addresses that will be used for the devices in the home with private IPv4 addresses. The following conditions apply:

1. If one of ip_address, netmask, pool_start, or max clients is specified, they are all mandatory and must all be specified. If none of these items are specified, the node default is used.
2. The ip_address, which is the inside gateway address for the home, must be pre-configured as a NAT inside address under the router on the node.
3. The resulting DHCP pool of addresses for devices must fall within the home subnet and must not include the home gateway ip_address.

GET

Retrieves the HOME_POLICY_PROFILE as data object **home_policy**.

Success response example:


```
{
 "status": 200,
 "message": "Success Message",
 "data": {
  "bandwidth": {
   "max_up": 30000000,
   "max_down": 4000100000,
   "guaranty_up": -2,
   "guaranty_down": 20000000
  },
  "wan": {
   "filter_icmp_inbound": false,
   "dmz_enabled": true,
   "dmz_host": "192.168.1.99",
   "vlan_id": 1
  }
  "acl": {
   "acl_mode": 2,
   "portal_url": "http://69.252.80.100"
  }
  "vlan": [
    {
     "vlan_id": 1,
     "lan": {
      "ip_address": "192.168.1.1",
      "netmask": "255.255.255.0",
      "upnp_enabled": true,
      "domain": "lan",
      "dhcp_server": {
       "pool_start": 100,
       "max_clients": 50,
       "dhcp_lease_length": 86400,
       "dns": [
        "8.8.8.8",
        "4.2.2.1"
       ],
       "mac_binding": [
          {
           "mac_address": "1C0102030405",
           "ip_address": "192.168.1.105"
          },
```

```
      {
       "mac_address": "1C0102030406",
       "ip_address": "192.168.1.106"
      }
    ]
   },
  "static_ip": [
     {
      "mac_address": "1C01020304AD",
      "ip_address": "192.168.1.200"
     }
  ]
},
"portforward": [
   {
    "name": "",
    "protocol": [
       255
      ],
    "wan_port_start": 2222,
    "wan_port_end": 2222,
    "ip_address": "192.168.1.123",
    "host_port_start": 22,
    "host_port_end": 22
   },
   {
    "name": "",
    "protocol": [
       6
    ],
    "wan_port_start": 8000,
    "wan_port_end": 8001,
    "ip_address": "192.168.1.124",
    "host_port_start": 80,
    "host_port_end": 81
   }
],
"sticky_public_ip_list": [
      {
       "device_mac": "1C01020304AB",
       "public_ip": "65.24.34.23"
      },
      {
       "device_mac": "1C01020304AC",
       "public_ip": "65.24.34.25"
      }
],
```

```
    }
   ]
  "vlanV6": [
    {
     "vlan_id": 1,
     "lanV6": {
      "dns": [
       "33::1",
       "33::2"
      ],
            "slaacPrefix": null,
             "slaacPoolName": "slaac-IES207"
     }
    }
  ],
  "dnsbridge": {
   "dns": [
     "4.2.2.1:53"
   ]
  },
  "port_range": "500",
 }
}
```

3.2.2 home_policy parameters

Mandatory:

bandwidth (object, see 3.2.3 “bandwidth parameters” (p. 52))

wan (object, see 3.2.4 “wan parameters” (p. 53))—The public_ip parameter is not returned.

Optional:

ACL (object, see 3.2.5 “ACL parameters” (p. 53))

vlan (object array, see 3.2.6 “vlan parameters” (p. 54))

vlanV6 (object, see 3.2.7 “vlanV6 parameters” (p. 54))

port_range (integer)—Specifies the port range.

3.2.3 bandwidth parameters

Mandatory:

max_up (long integer)—Maximum upload bandwidth from BRG to vRGW in bits per second. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW. Maximum value of 2 000 000 000 000.

max_down (long integer)—Maximum download bandwidth from vRGW to BRG in bits per second. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW. Maximum value of 800 000 000 000.

Optional:

guaranty_up (long integer)—Guaranteed upload bandwidth in bits per second. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW. Maximum value of 2 000 000 000 000. **Not currently supported at the home level, will be ignored**.

guaranty_down (long integer)—Guaranteed download bandwidth in bits per second. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW. Maximum value of 800 000 000 000.

3.2.4 wan parameters

Mandatory:

public_ip (returned by GET only) (string)—Public IP address of the BRG.

filter_icmp_inbound (Boolean)—Specifies whether inbound ICMP traffic is filtered by firewall policy. Not supported for this release (always returns false).

dmz_enabled (Boolean)—Specifies whether DMZ is enabled.

dmz_host (string)—IPv4 address of the DMZ host. Is mandatory when dmz_enabled is TRUE.

Optional:

vlan_id (integer)—Indicates the VLAN that the DMZ host belongs to. Not supported for this release.

3.2.5 ACL parameters

Mandatory:

acl_mode (integer)—Integer enum values of 0, 1, or 2:

- 0 = Allowed
- 1 = Blocked
- 2 = Portal

Optional:

portal_url (string)—URL for redirection portal when acl_mode = 2. The URL should be of the form:

http(s)://<*host*>/<*path*><*querystrings*>

where

host is an IPv4 address or DNS name

path is the full path to a resource (can be empty)

querystrings are valid URL query strings

Can be null if acl_mode is 0 (Allowed) or 1 (Blocked).

3.2.6 vlan parameters

Mandatory:

vlan_id (integer)—ID of the VLAN.

lan (object, see 3.2.8 “lan parameters” (p. 54))

Optional:

associated_ips (object array, see 3.2.11 “associated_ips parameters” (p. 56))

portforward (object array, see 3.2.12 “portforward parameters” (p. 56))

sticky_public_ip_list (object array, see 3.2.13 “sticky_public_ip_list parameters” (p. 56))—Some of the connected devices could be assigned public IP addresses based on customer needs. This list provides a mapping between device MAC and public IP.

3.2.7 vlanV6 parameters

vlan_id (integer)—ID of the VLAN.

lanv6 (object, see 3.2.14 “lanv6 parameters” (p. 57))

3.2.8 lan parameters

Mandatory:

ip_address (string)—The subnet base IPv4 address.

netmask (string)—The IPv4 network masking of the subnet.

dhcp_server (object, see 3.2.9 “dhcp_server parameters” (p. 54))—Configuration of the DHCP server. All fields must be included.

Optional:

static_ip (object, see 3.2.15 “static_ip parameters” (p. 57))—Static IP/MAC assignment.

upnp_enabled (Boolean)—Specifies whether UPnP server is enabled.

domain (string)—The domain name advertised by the DHCP server to DHCP clients. Must be specified as “lan”.

3.2.9 dhcp_server parameters

Mandatory:

pool_start (integer)—The start of the IP pool as an offset. Must be a valid offset within the subnet defined by ip_address and netmask.

max_clients (integer)—The maximum number of IP leases available for this DHCP server from pool_start to pool_start + max_clients. Must be valid within the defined subnet.

dns (string array)—IP address of the the DNS servers (primary/secondary) that the DHCP server uses. Can specify up to two DNS servers.

Optional:

dhcp_lease_length (integer)—Indicates the length of time in seconds of a DHCP lease. Range of 60 to 4294944000. Default of 86400.

mac_binding (object array, see 3.2.10 “mac_binding parameters” (p. 55))—Binding pairs of MAC and IPv4 addresses.

3.2.10 mac_binding parameters

Mandatory:

mac_address (string)—The MAC address of the host.

ip_address (string)—The IPv4 address assigned to the MAC address.

3.2.11 associated_ips parameters

Mandatory:

device_mac (string)—The MAC address of the connected device.

ip_address (string)—IP address that was previously assigned to the connected device.

3.2.12 portforward parameters

Optional:

name (string)—A user defined name used to describe this entry. ^[a-zA-Z0-9]+$ with a 32 character maximum.

Mandatory:

protocol (integer array)—Protocol IDs are derived from standard IP protocol IDs as assigned by the IANA. Must be one of the following (cannot be empty):

- 6 = TCP
- 17 = UDP
- 255 = TCP and UDP

wan_port_start (integer)—Beginning of the range of ports to forward. Range of 1 to 65535.

wan_port_end (integer)—End of the range of ports to forward. Range of 1 to 65535.

ip_address (string)—IPv4 address of host to which to forward WAN traffic from:

- wan_port_start to wan_port end
- host_port_start to host_port_end

host_port_start (integer)—Beginning of the range of ports to forward to within the LAN. Range of 1 to 65535.

host_port_end (integer)—End of the range of ports to forward to within the LAN. Range of 1 to 65535.

3.2.13 sticky_public_ip_list parameters

Mandatory:

device_mac (string)—The MAC address of the device.

ip_address (string)—Public IPv4 address reserved for the device MAC. If left empty, the DHCP server selects the next available IP address. Must be 19 characters or less.

3.2.14 lanv6 parameters

dns (string array)—IP addresses of the DNS servers (primary/secondary) that the DHCP server uses. Up to two servers can be configured.

slaacPrefix (string)—SLAAC prefix. Not currently supported on POST.

slaacPoolName (string)—SLAAC pool name. Not currently supported on POST.

3.2.15 static_ip parameters

Mandatory:

mac_address (string)—The MAC address of the device.

ip_address (string)—The static IPv4 address assigned to the MAC address.

3.3 Device policy profile

3.3.1 /vcpe/{brg_id}/device/{device_mac}/device_policy_profile

Defines a set of device specific policies for the BRG.

Supported methods: GET

URI parameters

brg_id (string)—BRG ID.

device_mac (string)—The MAC address of the target connected device.

GET

Retrieves all connected device policies as the data object **device_policy**. See 3.3.2 “device_policy parameters” (p. 58)

Success response example:

```
{
 "status": 200,
 "message": "Success Message",
 "data": {
  "dhcp_stream_disabled": "true",
  "bandwidth":{
   "max_up": 30000000,
   "max_down": 40000000,
```

```
    "guaranty_up":10000000,
    "guaranty_down": 20000000
   },
   "acl": {
    "acl_mode": 2,
    "portal_url": "http://192.168.80.10"
   }
  }
}
```

3.3.2 device_policy parameters

Optional:

bandwidth (object, see 3.3.3 “bandwidth parameters” (p. 57))

sticky_public_ip (object, see 3.3.4 “sticky_public_ip parameters” (p. 59))

ACL (object, see 3.3.5 “ACL parameters” (p. 59))

dhcp_stream_disabled (Boolean)—Indicates whether DHCP streaming is disabled.

3.3.3 bandwidth parameters

Mandatory:

max_up (long integer)—Maximum upload bandwidth from BRG to vRGW in bits per second. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW. Maximum value of 2 000 000 000 000.

max_down (long integer)—Maximum download bandwidth from vRGW to BRG in bits per second. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW. Maximum value of 2 000 000 000 000.

Optional:

guaranty_up (long integer)—Guaranteed upload bandwidth in bits per second. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW. Maximum value of 2 000 000 000 000.

guaranty_down (long integer)—Guaranteed download bandwidth in bits per second. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW. Maximum value of 2 000 000 000 000.

3.3.4 sticky_public_ip parameters

Mandatory:

public_ip (string)—Specifies the IPv4 address to assign to the device on connection. If left empty, the DHCP server selects the next available IP address.

3.3.5 ACL parameters

Mandatory:

acl_mode (integer)—Integer enum values of 0, 1, or 2:
- 0 = Allowed
- 1 = Blocked
- 2 = Portal

Optional:

portal_url (string)—URL for redirection portal when acl_mode = 2. The URL should be of the form:

http(s)://<*host*>/<*path*><*querystrings*>

where

host is an IPv4 address or DNS name

path is the full path to a resource (can be empty)

querystrings are valid URL query strings

Can be null if acl_mode is 0 (Allowed) or 1 (Blocked).

3.4 Redirect profile

3.4.1 /vcpe/{brg_id}/default_redirect_profile

Retrieves the home gateway default redirect policy (DEFAULT_REDIRECT_PROFILE) as a list of objects corresponding to policy.

An AuthStatusCode that indicates reason for failure can be supplied. 0 indicates success. A non-zero value indicates the failure code.

URI parameters

brg_id (string)—BRG ID.

GET

Retrieves the DEVICE_REDIRECT_PROFILE as data object **redirect_policy**.

Success response example:


```
{
 "status": 200,
 "message": "Success Message",
 "data": {
  "bandwidth": {
   "max_up": 30000000,
   "max_down": 4000100000,
   "guaranty_up": -2,
   "guaranty_down": 20000000
  },
  "wan": {
   "filter_icmp_inbound": false,
   "dmz_enabled": true,
   "dmz_host": "192.168.1.99",
   "vlan_id": 1
  }
  "acl": {
   "acl_mode": 2,
   "portal_url": "http://69.252.80.100"
  }
  "vlan": [
    {
     "vlan_id": 1,
     "lan": {
      "ip_address": "192.168.1.1",
      "netmask": "255.255.255.0",
      "upnp_enabled": true,
      "domain": "lan",
      "dhcp_server": {
       "pool_start": 100,
       "max_clients": 50,
       "dhcp_lease_length": 86400,
       "dns": [
        "8.8.8.8",
        "4.2.2.1"
       ],
       "mac_binding": [
          {
           "mac_address": "1C0102030405",
           "ip_address": "192.168.1.105"
          },
```

```
          {
           "mac_address": "1C0102030406",
           "ip_address": "192.168.1.106"
          }
        ]
       },
      "static_ip": [
         {
          "mac_address": "1C01020304AD",
          "ip_address": "192.168.1.200"
         }
      ]
     },
     "portforward": [
        {
         "name": "",
         "protocol": [
            255
          ],
         "wan_port_start": 2222,
         "wan_port_end": 2222,
         "ip_address": "192.168.1.123",
         "host_port_start": 22,
         "host_port_end": 22
        },
        {
         "name": "",
         "protocol": [
            6
         ],
         "wan_port_start": 8000,
         "wan_port_end": 8001,
         "ip_address": "192.168.1.124",
         "host_port_start": 80,
         "host_port_end": 81
        }
     ],
    }
   ]
  "vlanV6": [
    {
     "vlan_id": 1,
     "lanV6": {
      "dns": [
       "33::1",
       "33::2"
      ],
```

```
            "slaacPrefix": null,
             "slaacPoolName": "slaac-IES207"
      }
     }
   ],
   "port_range": "500",
  }
}
```

3.4.2 redirect_policy parameters

Mandatory:

bandwidth (object, see 3.4.3 “bandwidth parameters” (p. 61))

wan (object, see 3.4.4 “wan parameters” (p. 63))—The public_ip parameter is not returned.

Optional:

ACL (object, see 3.4.5 “ACL parameters” (p. 63))

vlan (object array, see 3.4.6 “vlan parameters” (p. 64))

vlanV6 (object, see 3.4.7 “vlanV6 parameters” (p. 64))

port_range (integer)—Specifies the port range.

3.4.3 bandwidth parameters

Mandatory:

max_up (long integer)—Maximum upload bandwidth from BRG to vRGW in bits per second. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW. Maximum value of 2 000 000 000 000.

max_down (long integer)—Maximum download bandwidth from vRGW to BRG in bits per second. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW. Maximum value of 800 000 000 000.

Optional:

guaranty_up (long integer)—Guaranteed upload bandwidth in bits per second. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW. Maximum value of 2 000 000 000 000. **Not currently supported at the home level, will be ignored**.

guaranty_down (long integer)—Guaranteed download bandwidth in bits per second. -1 indicates unlimited bandwidth. -2 indicates not set, takes the default value from the vRGW. Maximum value of 800 000 000 000.

3.4.4 wan parameters

Mandatory:

public_ip (returned by GET only) (string)—Public IP address of the BRG.

filter_icmp_inbound (Boolean)—Specifies whether inbound ICMP traffic is filtered by firewall policy. Not supported for this release (always returns false).

dmz_enabled (Boolean)—Specifies whether DMZ is enabled.

dmz_host (string)—IPv4 address of the DMZ host. Is mandatory when dmz_enabled is TRUE.

Optional:

vlan_id (integer)—Indicates the VLAN that the DMZ host belongs to. Not supported for this release.

3.4.5 ACL parameters

Mandatory:

acl_mode (integer)—Integer enum values of 0, 1, or 2:
- 0 = Allowed
- 1 = Blocked
- 2 = Portal

Optional:

portal_url (string)—URL for redirection portal when acl_mode = 2. The URL should be of the form:

http(s)://<*host*>/<*path*><*querystrings*>

where

host is an IPv4 address or DNS name

path is the full path to a resource (can be empty)

querystrings are valid URL query strings

Can be null if acl_mode is 0 (Allowed) or 1 (Blocked).

3.4.6 vlan parameters

Mandatory:

vlan_id (integer)—ID of the VLAN.

lan (object, see 3.4.8 “lan parameters” (p. 63))

Optional:

associated_ips (object array, see 3.4.11 “associated_ips parameters” (p. 65))

portforward (object array, see 3.4.12 “portforward parameters” (p. 65))

3.4.7 vlanV6 parameters

vlan_id (integer)—ID of the VLAN.

lanv6 (object, see 3.2.14 “lanv6 parameters” (p. 57))

3.4.8 lan parameters

Mandatory:

ip_address (string)—The subnet base IPv4 address.

netmask (string)—The IPv4 network masking of the subnet.

dhcp_server (object, see 3.4.9 “dhcp_server parameters” (p. 64))—Configuration of the DHCP server. All fields must be included.

Optional:

static_ip (object, see 3.4.14 “static_ip parameters” (p. 66))—Static IP/MAC assignment.

upnp_enabled (Boolean)—Specifies whether UPnP server is enabled.

domain (string)—The domain name advertised by the DHCP server to DHCP clients. Must be specified as “lan”.

3.4.9 dhcp_server parameters

Mandatory:

pool_start (integer)—The start of the IP pool as an offset. Must be a valid offset within the subnet defined by ip_address and netmask.

max_clients (integer)—The maximum number of IP leases available for this DHCP server from pool_start to pool_start + max_clients. Must be valid within the defined subnet.

dns (string array)—IP address of the the DNS servers (primary/secondary) that the DHCP server uses. Can specify up to two DNS servers.

Optional:

dhcp_lease_length (integer)—Indicates the length of time in seconds of a DHCP lease. Range of 60 to 4294944000. Default of 86400.

mac_binding (object array, see 3.4.10 “mac_binding parameters” (p. 64))—Binding pairs of MAC and IPv4 addresses.

3.4.10 mac_binding parameters

Mandatory:

mac_address (string)—The MAC address of the host.

ip_address (string)—The IPv4 address assigned to the MAC address.

3.4.11 associated_ips parameters

Mandatory:

device_mac (string)—The MAC address of the connected device.

ip_address (string)—IP address that was previously assigned to the connected device.

3.4.12 portforward parameters

Optional:

name (string)—A user defined name used to describe this entry. ^[a-zA-Z0-9]+$ with a 32 character maximum.

Mandatory:

protocol (integer array)—Protocol IDs are derived from standard IP protocol IDs as assigned by the IANA. Must be one of the following (cannot be empty):

- 6 = TCP
- 17 = UDP
- 255 = TCP and UDP

wan_port_start (integer)—Beginning of the range of ports to forward. Range of 1 to 65535.

wan_port_end (integer)—End of the range of ports to forward. Range of 1 to 65535.

ip_address (string)—IPv4 address of host to which to forward WAN traffic from:

- wan_port_start to wan_port end
- host_port_start to host_port_end

host_port_start (integer)—Beginning of the range of ports to forward to within the LAN. Range of 1 to 65535.

host_port_end (integer)—End of the range of ports to forward to within the LAN. Range of 1 to 65535.

3.4.13 lanv6 parameters

dns (string array)—IP addresses of the DNS servers (primary/secondary) that the DHCP server uses. Up to two servers can be configured.

slaacPrefix (string)—SLAAC prefix. Not currently supported on POST.

slaacPoolName (string)—SLAAC pool name. Not currently supported on POST.

3.4.14 static_ip parameters

Mandatory:

mac_address (string)—The MAC address of the device.

ip_address (string)—The static IPv4 address assigned to the MAC address.

3.5 Retrieve BRG ID

3.5.1 /vcpe/retrieve_brg_id

Retrieves the BRG ID from the BRGC during Device Auth. The method is POST. The PCMP populates the JSON body with key and value pairs available from the Device Access Request.

This method is used by the PCMP when the BRGId_ResolvingMode=0 (always ask) OR when BRGId_ResolvingMode=1 (try to resolve) but resolve is not possible based on the incoming Access Request. The resolving mode is defined in the pcmpConverter config file.

Supported methods: POST

POST

Retrieves the BRG ID.

```
[
{"value":"65.97.3.45","key":"4"}
,
{"value":"4/1/14:500.1","key":"87"}
]
```

Sample success response:

```
{"status":200,"message":"Success Message","data":"Home1"}
```

Sample response if no BRG ID is found:

```
{"status":500,"message":"No BRG ID found.","error_code":5101}
```

3.5.2 Parameters

value (string)—The PCMP will populate this value based on the access request.

key (string)—The data type of the AVP.

3.5.3 Supported AVPs

The following are the supported AVPs for querying the BRGC for BRG ID:

```
1  : User Name
4  : NAS-IP-Address
30 : Called-Station-Id
```

```
31 : Calling-Station-Id
32 : NAS-Identifier
61 : NAS-Port-Type
87 : NAS-Port-Id
3561-1  : Agent-Circuit-Id
3561-2  : Agent-Remote-Id
6527-27 : Alc-Client-Hardware-Address
```

Keys 4 and 87 are typically used to retrieve the BRG ID.

The possible keys are defined in the pcmpConverter config file under "pcmp" in "BRGIdLookupAVPIds". For the keys and values that are available to PCMP from the Device Auth request, PCMP will add them to the query. If the key value is unknown to PCMP, the key and value pair will be omitted in the request. If all AVP values are missing, the PCMP sends an access reject.

4 Notifications and reporting API

4.1 PCMP to BRGC notification API

4.1.1 Overview

This section describes the notifications that the PCMP sends to the BRGC about the BRG and connected devices. This API uses Apache Kafka in queuing model for the transportation layer. Notification messages use Google protocol buffers. The MessageEnvelope message is used to encapsulate requests while the MessageType property identifies the message type with a value that corresponds to the BRG-Notifications and BRG-Reports that can be sent from PCMP to BRGC.

MessageType enum (all supported notifications)

```
enum MessageType {
 // real time notifications
 MSG_DEVICE_STATUS = 1;
 MSG_DEVICE_DHCP_STREAM = 2;
 MSG_BRG_STATUS = 3;
 MSG_DEVICE_PORT_RANGE_STATUS = 4;
 // periodic reports
 MSG_DEVICE_REPORT = 100;
}
```

MessageEnvelope protocol message buffer

```
message MessageEnvelope {
 required MessageType message_type = 1;
 optional string cpe_mac = 2;
 optional uint64 message_time = 3;
 extensions 8 to 1024;
}
extend MessageEnvelope {
 optional MessageDeviceStatus message_device_status = 8;
 optional MessageDeviceDHCPStream message_device_dhcp_stream = 9;

 optional MessageBrgStatus message_brg_status = 10;
 optional MessageDeviceReport message_device_report = 11;
 optional MessageDevicePortRangeStatus message_device_portrange_
status = 12;
}
```

Name	Type	Notes
cpe_mac	String	MAC address of the target BRG.
message_type	Enum	Is value from enum MessageType and is equal to notification method.
message_time	uint64	Time in seconds since epoch the message was generated.

4.1.2 MessageDeviceStatus

When a connected device connects or disconnects from the BRG, the PCMP notifies the BRGC using this API.

```
message MessageDeviceStatus {
 required string device_mac = 1;
 required DeviceActivity device_activity = 2;
 optional DeviceConnectionData device_connection_data = 4;
 required acct_session_id = 5;
 required acct_multi_session_id = 6;
}
```

Table 1 MessageDeviceStatus parameters

Name	Type	Notes
device_mac	String	The MAC address of a connected device.
device_activity	DeviceActivity enum	`enum DeviceActivity { ACTIVE = 1; INACTIVE = 2; OFFLINE = 3; }`
device_ connection_data	DeviceConnec-tionData object	See Table 2, “DeviceConnectionData parameters” (p. 71).
acct_session_id	String	Accounting session ID.
acct_multi_ session_id	String	Accounting multi-session ID.

```
message DeviceConnectionData {
 required string ip_address = 1;
```

```
 optional string ipv6_prefix = 2;
 optional uint32 vlan_id = 3;
 required string hostname = 4;
 optional Interface interface = 6;
 optional string ssid = 7;
}
```

Table 2 DeviceConnectionData parameters

Name	Type	Notes
ip_address	string	IP address of the new connected device
ipv6_prefix	string	IPv6 prefix of the new connected device
vlan_id	uint32	vLANId to which connected device is connected
hostname	string	hostname of connected device (if given one) as per RFC1123
interface	enum	`enum Interface { UNKNOWN = 0 WIRED = 1; WIRELESS = 2; }`
ssid (Conditional)	string	WIFI SSID the device is connecting to. If interface is WIRED, ssid field should not be supplied.

```
message DevicePortRange {
 required uint32 startport = 1;
 required uint32 endport = 2;
}
```

Table 3 DevicePortRange parameters

Name	Type	Notes
startport	uint32	Start of a port range allocated to this connected device.

Table 3 DevicePortRange parameters (continued)

Name	Type	Notes
endport	uint32	End of a port range allocated to this connected device.

Sample message:

```
message_type: MSG_DEVICE_STATUS
cpe_mac: "0A0A0A555555"
message_time: 1454446226
[message_device_status] {
  device_mac: "0A5552010101"
  device_activity: ACTIVE
  device_connection_data {
    ip_address: "192.168.1.123"
    hostname: ""
    interface: WIRELESS
    ssid: "ts"
  }
  acct_session_id: "905BFF00000076325011E3"
  acct_multi_session_id: "905BFF0000006A325003B0"
}
```

4.1.3 BRGStatus

```
message BRGStatus {
 required string brg_id = 1;
 required BrgStatus status = 2;
 required string vrgapi_endpoint_address = 3;
 optional MessageTunnelStatus tunnel_status = 4;
 optional MessageL2Status l2_status = 5;
 optional string sub_prof_str = 101;
}
```

Table 4 BRGStatus parameters

Name	Type	Notes
brg_id	string	Unique ID of the BRG.

Table 4 BRGStatus parameters (continued)

Name	Type	Notes
status	enum	`enum BrgStatus { FIRST_DEVICE_ONLINE = 1; LAST_DEVICE_OFFLINE = 2; }`
status_reason_code	uint32	status_reason code corresponding to specific Tunnel Status. Tunnel_Present is possible interim status while tunnel is being established. If Tunnel setup was not completed, the status_reason_code indicates cause of such state.
vrgapi_endpoint_address	string	IP address and port of the PCMP REST API.
tunnel_status	TunnelStatus object	See 4.1.4 “TunnelStatus” (p. 74).
l2_status	L2Status object	See 4.1.5 “L2Status” (p. 75).
sub_prof_str	string	The subscriber profile string.

Sample message 1:

```
message_type: MSG_BRG_STATUS
cpe_mac: "0D0D0D505050"
message_time: 1466080126
[message_brg_status] {
 brg_id: "0D0D0D505050"
 status: FIRST_DEVICE_ONLINE
 vrgapi_endpoint_address: "10.1.1.1:8080"
 l2_status {
  port_id: "4/1/14:500.2"
 }
 sub_prof_str: "sub_prof_50_1"
}
```

Sample message 2:

```
message_type: MSG_BRG_STATUS
cpe_mac: "0A0A0A505050"
message_time: 1466080722
[message_brg_status] {
```

```
 brg_id: "0A0A0A505050"
 status: FIRST_DEVICE_ONLINE
 vrgapi_endpoint_address: "10.1.1.1:8080"
 tunnel_status {
  tunnel_connection_status: TUNNEL_ESTABLISHED
  wan_ip_address: "10.2.2.3"
  lan_ip_address: "10.1.1.4"
  tunnel_id: 875836419
 }
}
```

4.1.4 TunnelStatus

When a hCPE establishes or drops a GRE tunnel to the BRG, the PCMP notifies the BRGC by calling this API.

```
message MessageTunnelStatus {
 required TunnelConnectionStatus tunnel_connection_status = 1;
 required string wan_ip_address = 2;
 required string lan_ip_address = 3;
 required uint32 tunnel_id = 4;
}
```

Table 5 TunnelStatus parameters

Name	Type	Notes

tunnel_ connection_status	enum	Status of the GRE tunnel between hCPE and BRG.
		`enum TunnelConnectionStatus {`
		` TUNNEL_ESTABLISHED = 1;`
		` TUNNEL_DROPPED = 2;`
		` TUNNEL_PRESENT = 3;`
		`}`

wan_ip_address	string	BRG remote interface address.
lan_ip_address	string	vRGW address on the node.
tunnel_id	uint32	Unique id that represents the tunnel used by BRG to connect to vRGW.

Sample message 1:

```
message_type: MSG_TUNNEL_STATUS
cpe_mac: "009100010101"
message_time: 835131613
[message_tunnel_status] {
```

```
  tunnel_connection_status: TUNNEL_DROPPED
  wan_ip_address: "91.91.91.91"
  lan_ip_address: "10.91.91.4"
  tunnel_id: 1532713819
}
```

Sample message 2:

```
message_type: MSG_TUNNEL_STATUS
cpe_mac: "009100010101"
message_time: 835131620
[message_tunnel_status] {
  tunnel_connection_status: TUNNEL_ESTABLISHED
  wan_ip_address: "91.91.91.91"
  lan_ip_address: "10.91.91.4"
  tunnel_id: 1532713819
}
```

4.1.5 L2Status

```
message MessageL2Status {
 required string port_id = 1;
}
```

Table 6 L2Status parameters

Name	Type	Notes
port_id	string	Port as identified on the vRGW.

4.1.6 DevicePortrangeStatus

```
message MessageDevicePortrangeStatus {
 required string device_mac = 1;
 required DevicePortRange device_port_range = 2;
 required ip_address = 3;
 required vlan_id = 4;
 required device_activity = 5;
 }
```

Table 7 DevicePortrangeStatus parameters

Name	Type	Notes
device_mac	string	The MAC address of a connected device.

Table 7 DevicePortrangeStatus parameters (continued)

Name	Type	Notes
DevicePortRange	object	See Table 3, “DevicePortRange parameters” (p. 71).
ip_address	string	IP address of the device.
vlan_id	uint32	vLANId to which connected device is connected

device_activity	enum	`enum DeviceActivity {`
		` ACTIVE = 1;`
		` INACTIVE = 2;`
		` OFFLINE = 3;`
		`}`

Sample message:

```
message_type: MSG_DEVICE_PORTRANGE_STATUS
cpe_mac: "0A0A0A505050"
message_time: 1470127145
[message_device_portrange_status] {
  device_mac: "200000000001"
  device_port_range {
    startport: 5001
    endport: 5015
  }
  ip_address: "192.168.1.2"
  vlan_id: "23"
  device_activity: ACTIVE
}
```

4.2 PCMP to BRGC reporting API

4.2.1 MessageDeviceReport

The PCMP uses this API to report a list of connected devices information from the BRG to the BRGC.

```
message MessageDeviceReport {
 repeated Device devices = 1;
}
```

Table 8 Device parameters

Name	Type	Notes
device_mac	string	The MAC address of a connected device.
device_ connection_data	Device-Connec-tionData object	See Table 2, “DeviceConnectionData parameters” (p. 71).
traffic_data	TrafficData/ TrafficDa-taElem objects	See Table 9, “TrafficDataElem parameters” (p. 78).
acct_session_id	string	Accounting session ID.
acct_multi_ session_id	string	Accounting multi-session ID.

Sample message:

```
message_type: MSG_DEVICE_REPORT
brg_id: "0A0A0A010101"
message_time: 1463688835
[message_device_report] {
 devices {
  device_mac: "500500010101"
  device_connection_data {
   ip_address: "192.168.101.2"
   ipv6_prefix: "280:1:32:0:0:0:0:0/64"
   vlan_id: 0
   hostname: ""
   interface: WIRED
   ssid: ""
  }
  traffic_data {
   traffic_data_elem {
    start_time: 1463666003
    end_time: 1463688835
    up: 25840
    down: 25840
   }
  }
  acct_session_id: "9039FF00005FFD573DC553"
  acct_multi_session_id: "9039FF00005FFF573DC553"
 }
}
```

4.2.2 TrafficDataElem

```
message TrafficDataElem {
 required uint64 start__time = 1;
 required uint64 end_time = 2;
 required uint32 up = 3;
 required uint32 down = 4;
}
```

Table 9 TrafficDataElem parameters

Name	Type	Notes
start_time	uint64	Start time.
end_time	uint64	End time.
up	uint32	Bitrate up.
down	uint32	Bitrate down.

5 PCMP use cases

5.1 Web portal authentication

5.1.1 Web portal triggered AAA authentication using PCMP

In a TPSDA deployment, one of the ways to authenticate subscribers is through the use of a web portal where the user enters his/her credentials which are then authenticated by a AAA server.

PCMP can send AAA authentication requests on behalf of the web portal, and then send AAA Change of Authorization (CoA) messages to the 7750 SR that contain the proper subscriber configuration if the authentication is successful.

5.1.2 Authentication request flow diagrams

The following flow diagrams describe the sequence flow for the use case.

Figure 6 User accepted

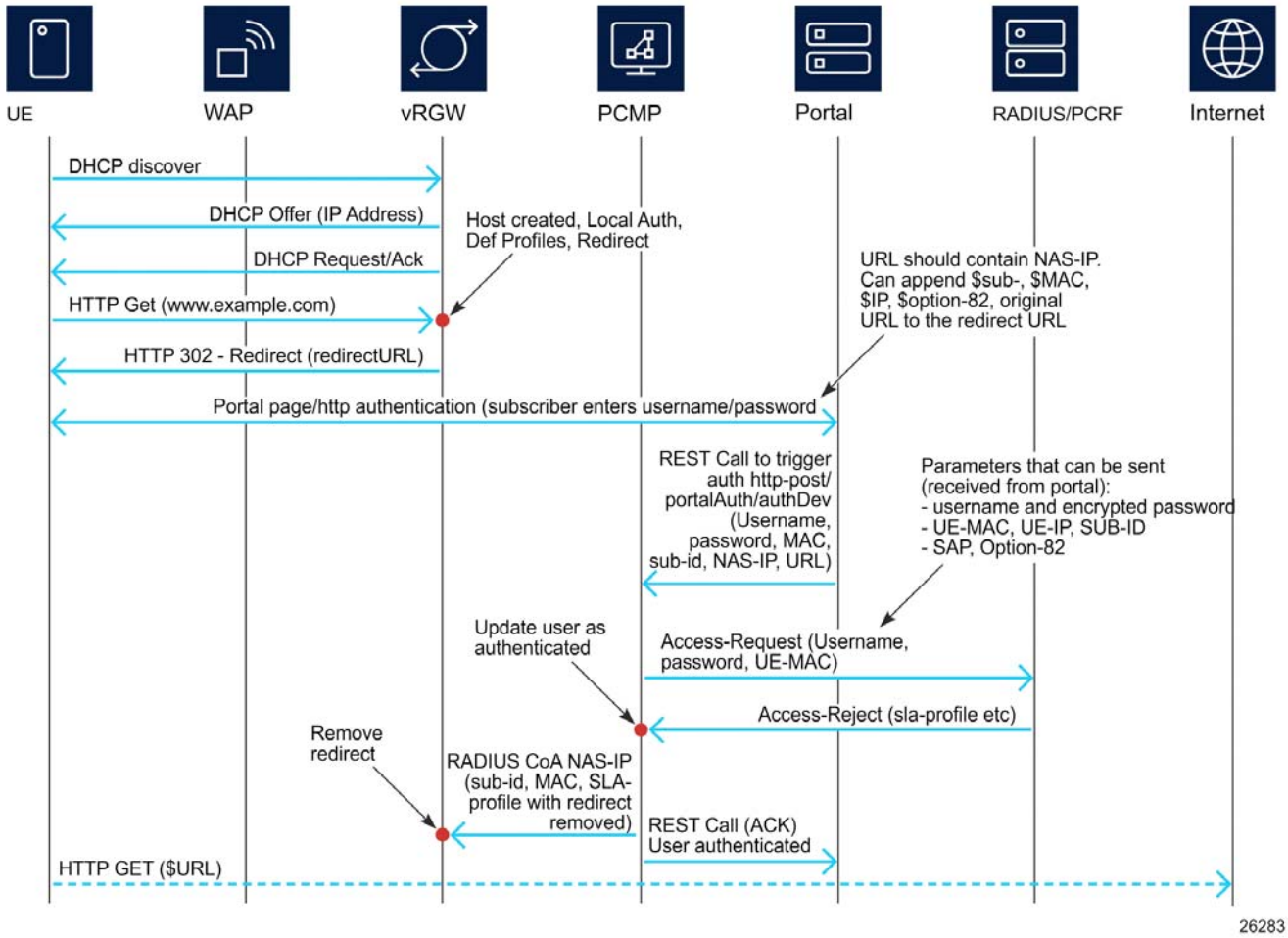
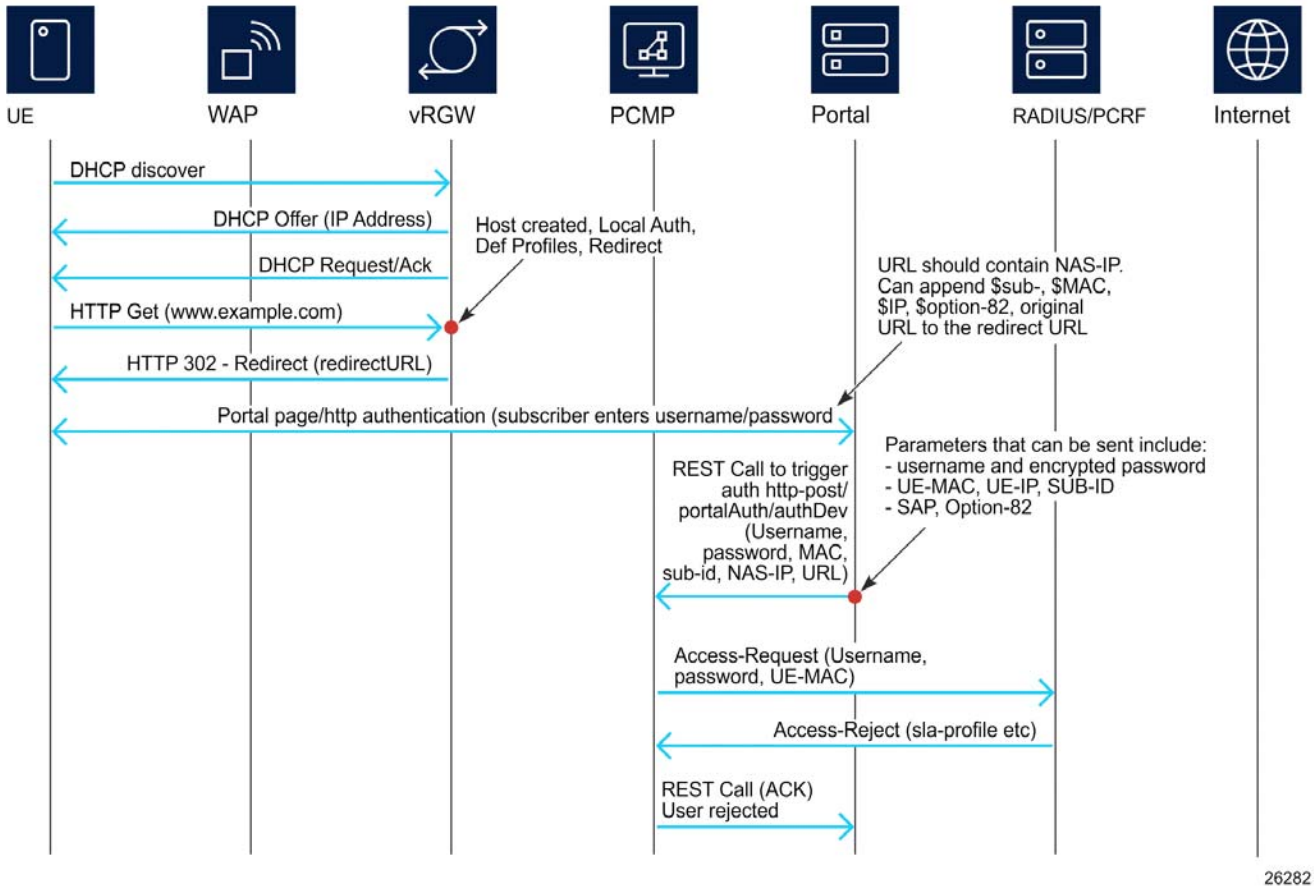


Figure 7 User rejected



5.1.3 Implementation notes

Consider the following information regarding use case and API implementation:

- The MAC notation type (capitalization and separator format) used must match the type used by the node. This is an exception to the general PCMP rule specified in 1.8 “API implementation specifics and recommendations” (p. 17).
- For this use case, there is a 1:1 ratio of PCMP and 7750 SR.
- Supported IP type is IPv4 only.

5.2 REST API for Web Portal Authentication

5.2.1 /portalAuth/authDev

Forwards authentication request to AAA server. If accepted, VSAs returned by the AAA server in the Access-Accept response will be sent by PCMP to the vRGW in a Change of Authorization (CoA) Call.

Supported methods: POST

POST

Sends the authentication credentials as object **PortalAuth**.

POST request example:

```
{
  "username": "john_doe",
  "password": "password",
  "dev_mac": "00:03:02:03:03:99",
  "subscriber_id": "subscriber",
  "nas_ip": "10.1.1.1",
  "origin_url": "http://example.com",
  "option_82": "optional"
}
```

PortalAuth parameters

Mandatory:

username (string)—Name of the user to authenticate.

password (string)—User password.

dev_mac (string)—MAC address of the device to authenticate.

subscriber_id (string)—Subscriber ID associated with the device to authenticate.

nas_ip (string)—7750 SR IP address. This IP address is hardcoded in the PCMP configuration and while it must be passed in this method, the actual value does not matter.

Optional:

origin_url (string)—Origin URL.

option_82 (string)—DHCP option 82.

POST response messages

HTTP Status Code	Reason
201	Success (authentication successful)
400	Bad request
500	Internal server error (authentication failed, invalid request)

POST response example (authentication successful)

Corresponds to message 11a in sequence flow Figure 6, “User accepted” (p. 80).

```
{
  "status": 201,
  "message": "Success Message",
  "data": {
    "username": "john_doe",
    "password": "password",
    "dev_mac": "00:03:02:03:03:99",
    "subscriber_id": "sub2",
    "nas_ip": "10.0.1.1",
    "origin_url": "http://example.com",
    "option_82": "optional"
   }
}
```

POST response example (authentication failed)

Corresponds to message 11b in the sequence flow Figure 7, “User rejected” (p. 81).

```
{
  "status": 500,
  "message": "Access rejected",
  "error_code": 4000
}
```

5.3 Web portal AAA calls

5.3.1 Purpose

The following AAA calls are provided for reference.

5.3.2 AAA Access Request

Message 8 in the sequence flow diagrams (Figure 6, “User accepted” (p. 80) and Figure 7, “User rejected” (p. 81)).

```
rad_recv: Access-Request packet from host 138.120.200.60 port
35716, id=3, length=109
               User-Password = "password"
               User-Name = "john_doe"
               Proxy-State = 0x4e78436c69656e740000000000000003
               NAS-IP-Address = 10.0.1.1
               Alc-Subsc-ID-Str = "sub2"
               Alc-Client-Hardware-Addr = "00:03:02:03:03:99"
```

5.3.3 AAA Access-Accept Response expected from AAA server

Authentication successful (message 9a in the sequence flow Figure 6, “User accepted” (p. 80)).

```
Sending Access-Accept of id 3 to 138.120.200.60 port 35716
               Alc-SLA-Prof-Str = "subscriber111"
               Alc-Subsc-Prof-Str = "sub_prof_subscriber"
               Proxy-State = 0x4e78436c69656e740000000000000003
```

5.3.4 CoA message sent to SR by PCMP

Message 10 in the sequence flow Figure 6, “User accepted” (p. 80). PCMP sends the following type of CoA (for example):

```
2 2016/08/26 14:08:19.98 EST MINOR: DEBUG #2001 Base RADIUS
"RADIUS: Receive
  Change of Authorization(43) id 4 len 115 from 5.5.5.4:38970
vrid 1
    PROXY STATE [33] 16 NxClient
    VSA [26] 11 Alcatel(6527)
      SLA PROF STR [13] 9 subscriber111
    VSA [26] 17 Alcatel(6527)
      SUBSC PROF STR [12] 15 sub_prof_subscriber
    VSA [26] 6 Alcatel(6527)
      SUBSC ID STR [11] 4 sub2
    VSA [26] 19 Alcatel(6527)
      CHADDR [27] 17 00:03:02:03:03:99
```

Note: The VSAs returned by the AAA Access-Accept (telmex111, sub_prof_telmex) will be funneled into the CoA call. It is important to ensure that only CoA-compatible VSAs are returned by the AAA server, as the PCMP will not filter out incompatible VSAs. Sending CoA with incompatible VSAs may cause the CoA call to fail.