

Authentication, Authorization, and Accounting Functions

- Feature Summary and Revision History, on page 1
- Feature Description, on page 4
- AAA Overview, on page 4
- Using RADIUS Server Group, on page 5
- Specifying Method Order, on page 6
- Defining AAA Attributes, on page 6
- Making RADIUS Server Settings, on page 9
- Message-Authenticator validation for RADIUS servers, on page 9
- Balancing Transaction Load on the RADIUS Server, on page 13
- RADIUS Change of Authorization Overview, on page 14
- User Authentication and Authorization in the Local Network, on page 18
- Service Accounting, on page 19
- Utilizing Session accounting AAA profiles for Service Accounting and Accounting Send-Stop Setup-Failure, on page 21
- RADIUS Accounting Message Handling, on page 24
- Standard Compliance, on page 25
- RADIUS Automated Testing, on page 25
- Clear RADIUS operational statistics, on page 28
- RADIUS Attributes Filtering, on page 31
- Framed Route with tag and preference support, on page 35
- Configuring AAA Functions, on page 39

Feature Summary and Revision History

Summary Data

Table 1: Summary Data

Applicable Product(s) or Functional Area	cnBNG

Applicable Platform(s)	SMI
Feature Default Setting	Disabled - Configuration Required
Related Changes in this Release	Not Applicable
Related Documentation	*Cloud Native BNG Control Plane Command Reference Guide*

Revision History

Table 2: Revision History

Revision Details	Release
Introduced support to balance transaction load on the RADIUS server based on least outstanding transactions.	2026.04.0
Introduced Message-Authenticator validation support for RADIUS servers.	2026.01.0
Introduced support for clearing RADIUS operational statistics.	2026.01.0
The **user-plane-ip** keyword is removed from **nas-ip** and **nas-ipv6** commands under **radius accounting**, and **radius attribute** configurations. Introduced an optional command, **encode-user-plane-ip**, to allow explicit control over the encoding of user-plane IPv4/IPv6 addresses.	2025.04.0
Introduced support for framed routes.	2025.03.0
Enhancement introduced to session accounting and failure management with AAA profiles.	2025.01.0
Introduced support to balance transaction load on the RADIUS server based on least outstanding transactions.	2024.03.0
Introduced Asynchronous mode for sending RADIUS Accounting messages.	2024.03.0
Introduced support for different interim intervals for service and session accounting.	2024.02.0
Introduced a conditional approach to Change of Authorization (CoA) retries based on the Error-Cause AVPs.	2024.02.0

Revision Details	Release
Support is added for IPv6 transport between cnBNG and RADIUS AAA server, and also between cnBNG endpoint and CoA client.	2024.01.0
The RADIUS Automated Testing feature is introduced.	2024.01.0
The **user-plane-ip** keyword is added to the **nas-ip** attribute under **radius accounting**, and **radius attribute** configurations.	2023.04.0
Enhancement Introduced: The AAA feature is NSO-integrated.	2021.04.0

• This release introduces support for the Multi-Action Change of Authorization.
• Updated the following sections:
  • Configuring AAA Attributes: The **user-plane** is added as the one of the **format-order** identifiers while formatting AAA attributes.
  The **format-string** keyword was added to the AAframe format of the attribute.
  • Configuring RADIUS Accounting Options: The *nas_port* variable is added to explicitly specify the **nas-port** value.
  The **{ format-e** *format_e* **{ nas-port-type** *nas_port_type* **} }** options are added to the **nas-port** keyword.
  • Configuring RADIUS Attribute Format: The *nas_port* variable is added to explicitly specify the **nas-port** value.
  The **{ format-e** *format_e* **{ nas-port-type** *nas_port_type* **} }** options are added to the **nas-port** keyword.
	2021.03.0

First introduced.	2021.01.0

Feature Description



Note This feature is Network Services Orchestrator (NSO) integrated.

Note: All references to BNG in this chapter refer to the Cloud-Native Broadband Network Gateway (cnBNG).

This chapter provides information about configuring authentication, authorization, and accounting (AAA) functions on the BNG. BNG interacts with the RADIUS server to perform AAA functions. A group of RADIUS servers form a server group that is assigned specific AAA tasks. A method list defined on a server or server group lists methods by which authorization is performed. Some of the RADIUS features include creating specific AAA attribute formats, load balancing of RADIUS servers, throttling of RADIUS records, Change of Authorization (CoA), Session Accounting, and Service Accounting for QoS.

AAA Overview

AAA acts as a framework for effective network management and security. It helps in managing network resources, enforcing policies, auditing network usage, and providing bill-related information. BNG connects to an external RADIUS server that provides the AAA functions.

The RADIUS server performs the three independent security functions (authentication, authorization, and accounting) to secure networks against unauthorized access. The RADIUS server runs the Remote Authentication Dial-In User Service (RADIUS) protocol. (For details about RADIUS protocol, refer to RFC 2865). The RADIUS server manages the AAA process by interacting with BNG, and databases and directories containing user information.

The RADIUS protocol runs on a distributed client-server system. The RADIUS client runs on BNG (Cisco ASR 9000 Series Router) that sends authentication requests to a central RADIUS server. The RADIUS server contains all user authentication and network service access information.

The AAA processes, the role of RADIUS server during these processes, and some BNG restrictions, are explained in these sections:

Authentication

The authentication process identifies a subscriber on the network, before granting access to the network and network services. The process of authentication works on a unique set of criteria that each subscriber has for gaining access to the network. Typically, the RADIUS server performs authentication by matching the credentials (user name and password) the subscriber enters with those present in the database for that subscriber. If the credentials match, the subscriber is granted access to the network. Otherwise, the authentication process fails, and network access is denied.

Authorization

After the authentication process, the subscriber is authorized for performing certain activity. Authorization is the process that determines what type of activities, resources, or services a subscriber is permitted to use. For example, after logging into the network, the subscriber may try to access a database, or a restricted website. The authorization process determines whether the subscriber has the authority to access these network resources.

AAA authorization works by assembling a set of attributes based on the authentication credentials provided by the subscriber. The RADIUS server compares these attributes, for a given username, with information contained in a database. The result is returned to BNG to determine the actual capabilities and restrictions that are to be applied for that subscriber.

Accounting

The accounting keeps track of resources used by the subscriber during network access. Accounting is used for billing, trend analysis, tracking resource utilization, and capacity planning activities. During the accounting process, a log is maintained for network usage statistics. The information monitored include, but are not limited to - subscriber identities, applied configurations on the subscriber, the start and stop times of network connections, and the number of packets and bytes transferred to, and from, the network.

BNG reports subscriber activity to the RADIUS server in the form of accounting records. Each accounting record comprises of an accounting attribute value. This value is analyzed and used by the RADIUS server for network management, client billing, auditing, etc.

The accounting records of the subscriber sessions may timeout if the BNG does not receive acknowledgments from the RADIUS server. This timeout can be due to RADIUS server being unreachable or due to network connectivity issues leading to slow performance of the RADIUS server. It is therefore recommended that a RADIUS server **deadtime** be configured on the BNG, to avoid loss of sessions. Once this value is configured, and if a particular session is not receiving an accounting response even after retries, then that particular RADIUS server is considered to be non-working and further requests are not sent to that server.

Restrictions

- On session disconnect, transmission of the Accounting-Stop request to RADIUS may be delayed for a few seconds while the system waits for the "final" session statistics to be collected from the hardware. The Event-Timestamp attribute in that Accounting-Stop request should, however, reflect the time the client disconnects, and not the transmission time.

Using RADIUS Server Group

A RADIUS server group is a named group of one or more RADIUS servers. Each server group is used for a particular service. For example, in an AAA network configuration having two RADIUS server groups, the first server group can be assigned the authentication and authorization task, while the second group can be assigned the accounting task.

Server groups can include multiple host entries for the same server. Each entry, however, must have a unique identifier. This unique identifier is created by combining an IP address and a UDP port number. Different ports of the server, therefore, can be separately defined as individual RADIUS hosts providing a specific AAA service. In other words, this unique identifier enables RADIUS requests to be sent to different UDP ports on the same server. Further, if two different host entries on the same RADIUS server are configured for the same service (like the authentication process), then the second host entry acts as a fail-over backup for the first one. That is, if the first host entry fails to provide authentication services, BNG tries with the second host entry. (The RADIUS host entries are tried in the order in which they are created.)

For assigning specific actions to the server group, see Configuring RADIUS Server Group, on page 54.

Specifying Method Order

Method order for AAA defines the methods using which authorization is performed, and the sequence in which these methods are executed. Before any defined authentication method is performed, the method order must be applied to the configuration mechanism responsible for validating user-access credentials.

On BNG, you have to specify the method order and the server group that will be used for AAA services. For specifying method order, see Configuring Method Order for AAA, on page 43.

Defining AAA Attributes

The AAA attribute is an element of RADIUS packet. A RADIUS packet transfers data between a RADIUS server and a RADIUS client. The AAA attribute parameter, and its value - form a Attribute Value Pair (AVP). The AVP carries data for both requests and responses for the AAA transaction.

The AAA attributes either can be predefined as in Internet Engineering Task Force (IETF) attributes or vendor defined as in vendor-specific attributes (VSAs). For more information about the list of BNG supported attributes, see RADIUS Attributes.

The RADIUS server provides configuration updates to BNG in the form of attributes in RADIUS messages. The configuration updates can be applied on a subscriber during session setup through two typical methods—per-user attributes, which applies configuration on a subscriber as part of the subscriber's authentication Access Accept, or through explicit domain, port, or service authorization Access Accepts. This is all controlled by the Policy Rule Engine's configuration on the subscriber.

When BNG sends an authentication or an authorization request to an external RADIUS server as an Access Request, the server sends back configuration updates to BNG as part of the Access Accept. In addition to RADIUS configuring a subscriber during setup, the server can send a change of authorization (CoA) message autonomously to the BNG during the subscriber's active session life cycle, even when the BNG did not send a request. These RADIUS CoA updates act as dynamic updates, referencing configured elements in the BNG and instructing the BNG to update a particular control policy or service policy.

BNG supports the concept of a "service", which is a group of configured features acting together to represent that service. Services can be represented as either features configured on dynamic-templates through CLI, or as features configured as RADIUS attributes inside Radius Servers. Services are activated either directly from CLI or RADIUS through configured "activate" actions on the Policy Rule Engine, or through CoA "activate-service" requests. Services can also be deactivated directly (removing all the involved features within the named service) through configured "deactivate" action on the Policy Rule Engine or through CoA "deactivate-service" requests.

The attribute values received from RADIUS interact with the subscriber session in this way:

- BNG merges the values received in the RADIUS update with the existing values that were provisioned statically by means of CLI commands, or from prior RADIUS updates.
- In all cases, values received in a RADIUS update take precedence over any corresponding CLI provisioned values or prior RADIUS updates. Even if you reconfigured the CLI provisioned values, the system does not override session attributes or features that were received in a RADIUS update.
- Changes made to CLI provision values on the dynamic template take effect immediately on all sessions using that template, assuming the template features have not already been overridden by RADIUS. Same applies to service updates made through CoA "service-update" requests.

AAA Attribute List

An attribute list is named list that contains a set of attributes. You can configure the RADIUS server to use a particular attribute list to perform the AAA function.

To create an attribute list, see Configuring RADIUS Attributes, on page 48.

AAA Attribute Format

It is possible to define a customized format for some attributes. For the configuration syntax for creating a new format, see Configuring AAA Attributes, on page 40.

Once the format is defined, the FORMAT-NAME can be applied to various AAA attributes such as username, nas-port-ID, calling-station-ID, and called-station-ID. The configurable AAA attributes that use the format capability are explained in the section Creating Attributes of Specific Format, on page 7.

To create a customized nas-port attribute and apply a predefined format to nas-port-ID attribute , see Configuring RADIUS Attribute Format, on page 49.

Specific functions can be defined for an attribute format for specific purposes. For example, if the input username is "text@abc.com", and only the portion after "@" is required as the username, a function can be defined to retain only the portion after "@" as the username. Then, "text" is dropped from the input, and the new username is "abc.com". To apply username truncation function to a named-attribute format, see Configuring AAA Attributes, on page 40.

Creating Attributes of Specific Format

BNG supports the use of configurable AAA attributes. The configurable AAA attributes have specific user-defined formats. The following sections list some of the configurable AAA attributes used by BNG.

Username

BNG has the ability to construct AAA username and other format-supported attributes for subscribers using MAC address, circuit-ID, remote-ID, and DHCP Option-60 (and a larger set of values available in CLI). The DHCP option-60 is one of the newer options that is communicated by the DHCP client to the DHCP server in its requests; it carries Vendor Class Identifier (VCI) of the DHCP client's hardware.

The MAC address attribute is specified in the CLI format in either of these forms:

- mac-address: for example, 0000.4096.3e4a
- mac-address-ietf: for example, 00-00-40-96-3E-4A
- mac-address-raw: for example, 000040963e4a
- mac-address-custom1: for example, 01.23.45.67.89.AB

  (This particular MAC address format is available only from Release 6.2.1 and later).

NAS-Port-ID

The NAS-Port-ID is constructed by combining BNG port information and access-node information. The BNG port information consists of a string in this form:

```
"eth phy_slot/phy_subslot/phy_port:XPI.XCI"
```

For 802.1Q tunneling (QinQ), XPI is the outer VLAN tag and XCI is the inner VLAN tag.

If the interface is QinQ, the default format of nas-port-ID includes both the VLAN tags; if the interface is single tag, it includes a single VLAN tag.

In the case of a single VLAN, only the outer VLAN is configured, using this syntax:

```
<slot>/<subslot>/<port>/<outer_vlan>
```

In the case of QinQ, the VLAN is configured using this syntax:

```
<slot>/<subslot>/<port>/<inner_vlan>.<outer_vlan>
```

In the case of a bundle-interface, the phy_slot and the phy_subslot are set to zero (0); whereas the phy_port number is the bundle number. For example, 0/0/10/30 is the NAS-Port-ID for a Bundle-Ether10.41 with an outer VLAN value 30.

The nas-port-ID command is extended to use the 'nas-port-type' option so that the customized format (configured with the command shown above) can be used on a specific interface type (nas-port-type).

If 'type' option is not specified, then the nas-port-ID for all interface types is constructed according to the format name specified in the command.

By default, the same NAS-Port-ID value is sent in **Accounting Start**, **Interim**, and **Stop** records for the entire subscriber session.

If the **service-name** attribute is included in the NAS-Port-ID format (see Configuring AAA Attributes), that format is not used for session accounting, because session accounting does not include a service-name value.

Calling-Station-ID and Called-Station-ID

BNG supports the use of configurable calling-station-ID and called-station-ID. The calling-station-ID is a RADIUS attribute that uses Automatic Number Identification (ANI), or similar technology. It allows the network access server (NAS) to send to the Access-Request packet, the phone number from which the call came from. The called-station-ID is a RADIUS attribute that uses Dialed Number Identification (DNIS), or similar technology. It allows the NAS to send to the Access-Request packet, the phone number that the user called from.

NAS-Port Format

NAS-Port is a 4-byte value that has the physical port information of the Broadband Remote Access Server (BRAS), which connects the Access Aggregation network to BNG. It is used both by Access-Request packets and Accounting-Request packets. To uniquely identify a physical port on BRAS, multiple pieces of information such as shelf, slot, adapter, and so on is used along with the port number. A configurable format called format-e is defined to allow individual bits or group of bits in 32 bits of NAS-Port to represent or encode various pieces that constitute port information.

Individual bits in NAS-Port can be encoded with these characters:

- Zero: 0
- One: 1
- PPPoX slot: S
- PPPoX adapter: A
- PPPoX port: P
- PPPoX VLAN Id: V
- PPPoX VPI: I

- PPPoX VCI: C
- Session-Id: U
- PPPoX Inner VLAN ID: Q

The permissible nas-port type values are:

Nas-port-types	Values	Whether value can be derived from associated interface
VIRTUAL_PPPOEOVLAN	36	Yes
VIRTUAL_PPPOEOQINQ	37	Yes
VIRTUAL_IPOEOVLAN	43	Yes
VIRTUAL_IPOEOQINQ	44	Yes

Note If a NAS-Port format is not configured for a NAS-Port-Type, the system looks for a default CLI configuration for the NAS-Port format. In the absence of both these configurations, for sessions with that particular NAS-Port-Type, the NAS-Port attribute is not sent to the RADIUS server.

Making RADIUS Server Settings

In order to make BNG interact with the RADIUS server, certain server specific settings must be made on the BNG router.

For more making RADIUS server settings, see Configuring RADIUS Server, on page 53.

Restriction

The service profile push or asynchronously pushing a profile to the system is not supported. To download a profile from Radius, the profile must be requested initially as part of the subscriber request. Only service-update is supported and can be used to change a service that was previously downloaded.

Message-Authenticator validation for RADIUS servers

Message-Authenticator validation for RADIUS servers is a security enhancement that

- ensures the integrity and authenticity of RADIUS authentication and accounting packets
- uses the Message-Authenticator attribute (Type 80) to protect against tampering and forgery, and
- helps mitigate protocol-level vulnerabilities such as Blast-RADIUS (CVE-2024-3596).

Blast-RADIUS (CVE-2024-3596) is a protocol-level security vulnerability that affects RADIUS servers and clients using non-EAP authentication methods over UDP or TCP. It enables a man-in-the-middle attacker to forge valid Access-Accept responses to failed authentication requests without knowing user passwords or

shared secrets, due to weaknesses in how the RADIUS protocol uses MD5 for packet authentication. Although Blast-RADIUS does not reveal user credentials, it can allow unauthorized access to network resources by exploiting the authentication process. To mitigate this vulnerability, it is recommended to enforce Message-Authenticator validation on all RADIUS packets or to migrate to secure transports such as RADIUS over TLS (RadSec).

Starting from Release 2026.03.0, Message-Authenticator validation supports both **strict** and **permissive** modes. Strict mode requires supported inbound packets to include a valid Message-Authenticator attribute. Permissive mode supports phased rollout by continuing to insert the attribute on outbound packets while accepting supported inbound packets that omit it. Packets with an invalid Message-Authenticator are silently dropped in both modes.

Table 3: Feature history

Feature Name	Release Information	Description
Enhanced Message-Authenticator validation for RADIUS servers	2026.03.0	You can now configure Message-Authenticator validation in **strict** or **permissive** mode to support phased rollout across cnBNG and AAA peers.
Message-Authenticator validation for RADIUS servers	2026.01.0	You can now protect your RADIUS authentication and accounting traffic from protocol-level attacks such as Blast-RADIUS by enabling Message-Authenticator validation. This feature ensures that all RADIUS packets include a cryptographic Message-Authenticator attribute, which both the client and server validate using a shared secret. If a packet fails validation, the system silently drops it, preventing forged or tampered responses from being accepted.

How Message-Authenticator validation works

When enabled, Message-Authenticator validation adds or verifies a cryptographic Message-Authenticator attribute on all RADIUS requests and responses. Both the RADIUS client (such as cnBNG) and server use a shared secret to generate and validate this attribute using HMAC-MD5, ensuring that only legitimate, untampered packets are accepted.

Validation behavior depends on the configured mode:

- **strict**: Access-Accept, Access-Reject, Access-Challenge, Accounting-Response, CoA-Request, and Disconnect-Request packets must include a valid Message-Authenticator attribute. Packets that omit the attribute or fail validation are silently dropped.
- **permissive**: outbound packets still include the Message-Authenticator attribute. Supported inbound packets that omit the attribute are accepted to support phased migration. Packets with an invalid Message-Authenticator are silently dropped.

For CoA and disconnect, if a request is received without the Message-Authenticator attribute in permissive mode, the request is processed and the corresponding ACK or NAK is sent without the attribute for that exchange.

Configuration considerations

You can configure Message-Authenticator validation globally, per RADIUS server, globally for CoA, or per CoA client by using **enable-msg-authenticator { strict | permissive }** command.

- The command has no default value.
- If the command is absent at both the global and entity scope, Message-Authenticator processing is off for that entity.
- A per-server or per-CoA-client setting overrides the corresponding global setting.
- If no per-entity setting is configured, the entity inherits the global value.

Note

- The previous form of **enable-msg-authenticator** is no longer used. After upgrade, existing configurations that used **enable-msg-authenticator** without a value must be re-applied as **enable-msg-authenticator strict**.
- To move from permissive mode back to full enforcement, explicitly configure **strict** keyword. Removing the command disables Message-Authenticator processing.
- Configuration changes take effect on subsequent packet transmissions. In-flight packets may continue to follow the previous behavior until the next retry.

Supported platforms

This feature is supported across all cnBNG deployment models, including AIO, multi-server, and OpenShift.

Configure Message-Authenticator validation for RADIUS servers

Enable Message-Authenticator validation to protect your cnBNG deployment from protocol-level vulnerabilities such as Blast-RADIUS by ensuring the authenticity and integrity of RADIUS packets.

Procedure

Step 1 Use the **enable-msg-authenticator { strict | permissive }** command to enable Message-Authenticator validation.

- Use this sample configuration to enable validation globally for all RADIUS servers:

```
Configure
  profile radius
   enable-msg-authenticator strict
  exit
```

- Use this sample configuration to enable validation for a specific RADIUS server. Global configuration takes effect if not set.

```
Configure
  profile radius
   server 10.105.254.43 1812
    type   auth
    enable-msg-authenticator strict
```

```
 exit
 server 10.105.254.43 1813
  type   acct
  enable-msg-authenticator strict
 exit
exit
```

- Use this sample configuration to enable validation for all CoA clients:

```
Configure
  profile coa
   enable-msg-authenticator strict
  exit
```

- Use this sample configuration to enable validation for a specific CoA client:

```
Configure
  profile coa
   client 10.105.254.43
    enable-msg-authenticator strict
   exit
  exit
```



Note For phased rollout, replace the **strict** keyword with the **permissive** keyword.

Configuration is resolved at the entity scope first. If a RADIUS server or CoA client has its own **enable-msg-authenticator** setting, that value takes precedence over the corresponding global setting. If the entity does not have its own value, it inherits the global setting. If the command is absent at both scopes, Message-Authenticator processing is off for that entity.

Use permissive mode to support phased rollout across cnBNG and AAA peers. After all peers have been updated to always include Message-Authenticator, explicitly change the configuration to **strict**.

- Use this sample configuration to enable permissive mode globally while keeping one server in strict mode:

```
Configure
  profile radius
    enable-msg-authenticator permissive
    server 10.105.254.43 1812
     type auth
     enable-msg-authenticator strict
    exit
  exit
```

In this example, the per-server **strict** setting takes precedence over the global **permissive** setting.

Step 2 Use the **show radius** command to monitor for failed authentications (for example, **bad authenticators**) and confirm feature operation.

Example:

```
bng# show radius
-------------------------------------------------------
Server: 10.105.254.105, port: 1812, status: instance 1: up  , port-type: Auth
2 requests, 0 pending, 0 retransmits
0 accepts, 0 rejects, 2 timeouts
0 bad responses, 0 bad authenticators
0 unknown types, 0 dropped, 0 ms latest rtt
Auto Test Stats:
```

```
instance 1:
  Requests:0 Pending:0 Retransmits:0
  Rejects:0 Timeouts:0 Responses:0
  BadAuth:0 Dropped:0 BadResp:0
```

Access-Accept, Access-Reject, Access-Challenge, and Accounting-Response packets that fail the Message-Authenticator check are silently dropped, and the **bad authenticator** counter is incremented. In strict mode, packets that omit the attribute are treated as failures. In permissive mode, packets that omit the attribute are accepted, but packets with an invalid Message-Authenticator are still dropped.

Example:

```
bng# show radius-dyn-auth
-------------------------------------------------------
 IP: 10.105.254.105
 ------------------
 COA:
 0 total-requests       0 inprocess-requests
     0 retry-request-drops  0 invalid-requests
     0 bad-authenticators   0 internal-errors
 0 ack-sent             0 nak-sent
 ------------------
 DISCONNECT:
 0 total-requests       0 inprocess-requests
     0 retry-request-drops  0 invalid-requests
     0 bad-authenticators   0 internal-errors
 0 ack-sent             0 nak-sent
```

CoARequest and DisconnectRequest packets that fail the Message-Authenticator check are silently dropped, and the **bad authenticator** counter is incremented. In strict mode, requests that omit the attribute are treated as failures. In permissive mode, requests that omit the attribute are accepted and processed, and the corresponding ACK or NAK is sent without the attribute for that exchange. Requests with an invalid Message-Authenticator are dropped in both modes.

Balancing Transaction Load on the RADIUS Server

Table 4: Feature History

Feature Name	Release Information	Description
Balancing Transaction Load on the RADIUS Server	2024.03.0	This feature enhances performance by distributing AAA messages across servers, ensuring faster response times. It selects the RADIUS server with the fewest outstanding transactions, rather than using the previous First server or Round Robin methods, which did not account for server load. This results in a more efficient handling of authentication, authorization, and accounting tasks.

The Balancing Transaction Load on the RADIUS Server feature provides a mechanism to share the load of RADIUS access and accounting transactions, across a set of RADIUS servers. Each AAA request processing is considered to be a transaction. cnBNG distributes batches of transactions to servers within a server group.

When the first transaction for a new batch is received, cnBNG determines the server with the lowest number of outstanding transactions in its queue. This server is assigned that batch of transactions. cnBNG keeps

repeating this determination process to ensure that the server with the least-outstanding transactions always gets a new batch. This method is known as the least-outstanding method of load balancing.

The size of each batch is configurable. Changes in the batch size may impact the CPU load and network throughput. There is no standard size for batches, but generally, more than 50 transactions is considered large, and fewer than 25 is considered small.

Retransmitted messages go to the same server. The number of outstanding RADIUS messages is tracked per server at the pod level. Load balancing applies to the named RADIUS server groups or a global server group specified in the subscriber profile. In AAA method lists, this group must be referred to as "radius." All servers in the RADIUS group are subject to load balancing.

RADIUS Server Status and Automated Testing

The Balancing Transaction Load on the RADIUS Server feature checks the status of servers before sending transaction batches. Only live servers receive transaction batches.

Transactions are not sent to servers marked as dead. A server remains marked as dead until its timer expires. The server is included again only when the RADIUS automated tester verifies it as alive. Otherwise, the server is excluded from the selection algorithm.

The RADIUS automated tester periodically sends a request to the server. If the server returns an **Access-Reject** message, it is alive. If not, it remains marked as dead until detected as alive.

If a server is unresponsive, it is marked as dead, and transactions fail over to the next available server.

When using the RADIUS automated tester, verify that the authentication, authorization, and accounting (AAA) servers respond to test packets sent by the network access server (NAS). Incorrect configurations may cause packet drops and servers to be erroneously marked as dead.

For configuring the load balancing on the RADIUS server, see Configuring RADIUS Server Selection Logic, on page 54.

RADIUS Change of Authorization Overview

The RADIUS Change of Authorization (CoA) function allows the RADIUS server to change the authorization settings for a subscriber who is already authorized. CoA is an extension to the RADIUS standard that allows sending asynchronous messages from RADIUS servers to a RADIUS client, like BNG.



Note A CoA server can be a different from the RADIUS server.

To identify the subscriber whose configuration needs to be changed, a RADIUS CoA server supports and uses a variety of keys (RADIUS attributes) such as Accounting-Session-ID, Username, IP-Address, and ipv4:vrf-id.

The RADIUS CoA supports:

- account-update — BNG parses and applies the attributes received as part of the CoA profile. Only subscriber-specific attributes are supported and applied on the user profile.
- activate-service — BNG starts a predefined service on a subscriber. The service settings can either be defined locally by a dynamic template, or downloaded from the RADIUS server.

- deactivate-service — BNG stops a previously started service on the subscriber, which is equivalent to deactivating a dynamic-template.

For a list of supported Vendor-Specific Attributes for account operations, see Vendor-Specific Attributes for Account Operations.

Note In order for BNG to enable interim accounting, it is mandatory for the CoA request to have both accounting method list from the dynamic-template and Acct-Interim-Interval attribute from the user profile. This behavior is applicable for accounting enabled through dynamic-template. Whereas, from Cisco IOS XR Software Release 5.3.0 and later, the CoA request needs to have only the Acct-Interim-Interval attribute in the user profile.

Service Activate from CoA

BNG supports activating services through CoA requests. The CoA **service-activate** command is used for activating services. The CoA request for the service activate should contain these attributes:

- "subscriber:command=activate-service" Cisco VSA
- "subscriber:service-name=<service name>" Cisco VSA
- Other attributes that are part of the service profile

The "<subscriber:sa=<service-name>" can also be used to activate services from CoA and through RADIUS.

Duplicate service activate requests can be sent to BNG from the CoA server. BNG does not take any action on services that are already activated. BNG sends a CoA ACK message to the CoA server under these scenarios:

- When a duplicate request with identical parameters comes from the CoA for a service that is already active.
- When a duplicate request with identical parameters comes from the CoA to apply a parameterized service.

BNG sends a CoA NACK message to the CoA server with an error code as an invalid attribute under these scenarios:

- When a request comes from the CoA to deactivate a non-parameterized service that is not applied to the session.
- When a request comes from the CoA to deactivate a parameterized service that is not applied to the session.
- When a duplicate request to apply a parameterized service is made with non-identical parameters from the CoA.
- When a request with non-identical parameters comes from CoA to deactivate a parameterized service.

Service Update from CoA

The service update feature allows an existing service-profile to be updated with a new RADIUS attribute list representing the updated service. This impacts any subscriber who is already activated with the service and new subscriber who activate the service in the future. The new CoA **service-update** command is used for activating this feature. The CoA request for the service update should have these attributes:

- "subscriber:command=service-update" Cisco VSA
- "subscriber:service-name=<service name>" Cisco VSA
- Other attributes that are part of the service profile

A service update CoA should have a minimum of these attributes:

- vsa cisco generic 1 string "subscriber:command=service-update"
- vsa cisco generic 1 string "subscriber:service-name=<service name>"

Web Logon with RADIUS Based CoA

To support Web Logon, a set of Policy Rule Events need to be configured in an ordered manner. These events are as follows:

- session-start:
  - On the start of a session, a subscriber is setup to get internet connectivity. The service is activated to redirect HTTP traffic to a Web portal for web-based logon.
  - Start the timer with duration for the maximum waiting period for authentication.
- account-logon—The Web portal collects the user credentials such as username and password and triggers a CoA account-logon command. When this event is triggered, subscriber username and password are authenticated by the RADIUS server. Once the authentication is successful, the HTTP redirect service is deactivated, granting user access to already connected internet setup. Also, the timer established in session-start must be stopped. However, if the authentication fails during account-logon, BNG sends a NAK CoA request, allowing for further authentication attempts to take place.
- timer expiry—When the timer expires, the subscriber session is disconnected based on the configuration.

Multi-Action Change of Authorization

BNG supports multi-action Change of Authorization (CoA) wherein service providers can activate and deactivate multiple services using a single CoA request. Multi-action CoA is supported for **Service-Activate** and **Service-Deactivate** commands.

During the multi-action CoA request, if any of the COA requests fail to activate or deactivate, then any of the services which have been activated or deactivated as part of that CoA request is rolled back to its previous state. The session restores back to the its pre-MA-CoA state upon failure to activation or deactivation.

An Example of a Multi-Action Change of Authorization Use Case

The following example lists the sequence of events that occur in the case of a PTA session initiation.

1. PTA session's web traffic redirected to a service portal (HTTP Redirect)
2. The user activates the first level of service through the service portal. A multi-action COA request is initiated in the following sequence.
   a. Deactivate redirection
   b. Activate Turbo Button 1

c. Activate VoIP with two channels

3. The user activates the second level of service through the service portal. A multi-action COA request is initiated in the following sequence.
   a. Deactivate Turbo Button 1
   b. Activate Turbo Button 2
   c. Deactivate VoIP with two channels
   d. Activate VoIP with 4 channels

Interworking with Service-Level Accounting

BNG supports Service-Level Accounting, where a service is a collection of features that are activated and deactivated as a group. Service-Level Accounting and MA-CoA features are independent, that is, they can be applied separately. However, MA-CoA accounts for services that are activated or deactivated that have Service-Level Accounting enabled through the dynamic template configuration.

Generating Accounting Records

The following cases describes how the multi-action CoA records are generated for accounting purposes.

MA-CoA ACK Case

- If MA-CoA request contains only service activate commands, then START accounting record for those services are generated after the CoA Ack is sent out.
- If MA-CoA request contains only deactivate services or combination of activate and deactivate services, then for those services START or STOP accounting records are generated after the CoA Ack is sent out.

MA-CoA NAK Case (Rollback scenario)

- If MA-CoA request fails due to presence of invalid command formats or due to internal software failure or due to presence of invalid service names, that are not defined in the box, in such cases the accounting START or STOP messages are not generated upon rollback.
- If MA-CoA request fails due to internal feature programming failure, then the Service-START or Service-STOP accounting records may be generated for the services that were activated or deactivated before the failure. After the failure, the rollback is initiated and appropriate Service-START or Service-STOP records are generated for these services.

Sample MA-COA Request

```
exec /bin/echo
"Cisco-AVPair='subscriber:sd=svcQoSAcct1',Cisco-AVPair='subscriber:sd=svcQoSAcct2',
Cisco-AVPair='subscriber:sd=svcQoSAcct3',Cisco-AVPair='subscriber:sa=qosin_coa',
Cisco-AVPair='subscriber:sa=qosout_coa',
Acct-Session-Id=00000001" | /usr/local/bin/radclient  -r 1 -x 5.11.17.31:1700 coa coa
```

Enhanced CoA with Conditional Retry Logic

Table 5: Feature History

Feature Name	Release Information	Description
Enhanced CoA with Conditional Retry Logic	2024.02.0	We have introduced a conditional approach to Change of Authorization (CoA) retries based on the Error-Cause AVPs carried in CoA response messages. The CoA client uses the error cause reason and determines whether to initiate a CoA retry. This enhancement can reduce unnecessary traffic and processing overhead, resulting in more efficient network operations and better allocation of resources.

If a CoA request is erroneous, the cnBNG rejects it with a CoA Negative Acknowledgment (NAK) response. The CoA client treats all NAK responses equally, and attempts to send CoA requests automatically. This logic of sending unconditional retries for all error types strains network resources, leading to potential performance degradation for both the policy plane and the cn-BNG.

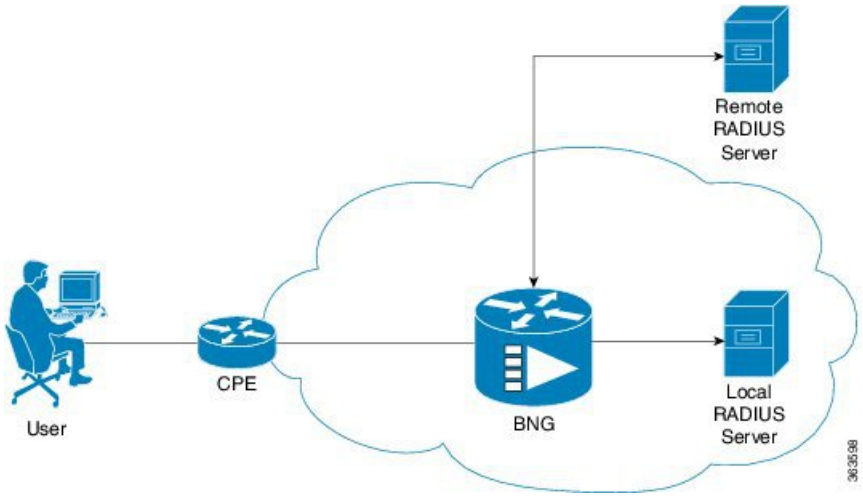
The Enhanced CoA with Conditional Retry Logic feature enables the CoA client to use the error cause carried in a CoA NAK response message and determine whether to send the CoA request again. For example, if the error cause is “503 Session Context Not found,” indicating that the subscriber session is nonexistent or already terminated, the CoA client would recognize that retrying the request would be futile as the cnBNG does not recognize the session context.

By adopting a conditional approach to CoA retries based on the Error-Cause AVPs, the system can avoid unnecessary traffic and processing overhead, resulting in a more efficient network performance.

User Authentication and Authorization in the Local Network

The user authentication and authorization in the local network feature in BNG provides the option to perform subscriber authorization locally (in a subscriber's network), instead of both remote authentication and authorization that occurs in RADIUS servers. With the User Authentication and Authorization in the Local Network feature, you can run the RADIUS server locally in your network, manage, and configure the RADIUS server locally in your network to the profile that is required for the environment. In the case of a remote RADIUS server, the RADIUS server is maintained by an external regulatory body (not within the subscriber's network) and subscriber will not be able to manage or configure the server.

Figure 1: User Authentication and Authorization in the Local Network



User Authentication and Authorization in the Local Network feature is used in a case when a user wants to perform a two-level authentication or authorization, first, a remote authentication (or authorization) followed by a local authorization (or authentication).

Note All the debug commands applicable to AAA server are applicable on User Authentication and Authorization in the Local Network feature.

Service Accounting

Accounting records for each service enabled on a subscriber can be sent to the configured RADIUS server. These records can include service-start, service-stop, and service-interim records containing the current state of the service and any associated counters. This feature is the Service Accounting feature. Service accounting records are consolidated accounting records that represent the collection of features that make up a service as part of a subscriber session.

Service accounting starts when a subscriber session comes up with a service enabled on it. This can happen through a dynamic template applied through a control policy, through access-accept (AA) messages when the session is authorized, or through a change of authorization (CoA), when a new service is applied on a subscriber session. Service accounting stops either when the session is terminated, or a service is removed from the session through CoA, or some other event that deactivates the service. Start records have no counters; interim and stop records with QoS counters are generated when service accounting is enabled for QoS. Interim accounting records can be generated, in between start and stop accounting, as an option with a pre-defined periodic interval. When the interim period is zero, interim accounting records are not created. Different interim intervals are based on every service for each session. Service accounting is enabled on each template, based on the configuration.

From Release 2024.02.0 onwards, different interim intervals for service and session accounting is supported. We recommend that you set the session interim accounting interval as a multiple of the service interim interval. For example, if the service accounting interim is set to 5 minutes, the session interim accounting interval must be set to a multiple of 5 minutes (such as 5, 10, 15 minutes, and so on).

Service Accounting is supported on bundle subscriber interfaces and line card subscriber interfaces.

For more information on service accounting for QoS, refer to Authentication, Authorization, and Accounting Functions, on page 1. For more information on commands to configure service accounting, refer to the Configuring Service Accounting.

Note The policy-map associated to a dynamic template can be edited to change the service parameters. However, this does not update the accounting records. Therefore, to generate all the accounting records accurately, it is recommended that a new service with all the required service parameters be created and associated to the new service, through a CoA.

For service accounting, statistics for ingress and egress QoS policies, which are applied under each service for a given subscriber, may need to be reported as part of the accounting interim and stop records. For each service, these QoS counters can be reported as part of the accounting records:

- BytesIn — Aggregate of bytes matching all classes of the ingress QoS policy for the service minus the policer drops.
- PacketsIn — Aggregate of packets matching all classes of the ingress QoS policy for the service minus the policer drops.
- BytesOut — Aggregate of bytes matching all classes of the egress QoS policy for the service minus the queuing drops.
- PacketsOut — Aggregate of packets matching all classes of the egress QoS policy for the service minus the queuing drops

Dynamic template features that support accounting statistic collection and require that their statistics be reported in the AAA service accounting records can enable accounting statistics on their features using the newly-introduced optional **acct-stats** configuration option. This option is not available for the features that do not support statistic collection. By default, QoS accounting statistics are disabled to optimize performance.

Note The QoS counters for each direction is reported only if a QoS policy is applied for that service in the given direction. For example, if a service does not have an ingress policy applied, BytesIn and PacketsIn counters are reported as being 0.

Pre-requisites

- Subscriber accounting, the parent accounting record for service accounting, must be configured to enable the service accounting feature to work.
- The keyword **acct-stats** must be configured in service-policy configuration to enable the service accounting feature to report feature counter information as part of the records.

Restriction

- IPv4 and IPv6 subscriber sessions has a single set of service accounting records. They are merged into one set of bytes_in, bytes_out, packets_in, packets_out counters.

- Service accounting is not supported for static sessions.

Utilizing Session accounting AAA profiles for Service Accounting and Accounting Send-Stop Setup-Failure

Table 6: Feature History

Feature Name	Release Information	Description
Utilizing Session accounting AAA profiles for Service Accounting and Accounting Send-Stop Setup-Failure	2025.01.0	This feature enhances session accounting by allowing the use of a single AAA profile for both session and service accounting, reducing the need for multiple feature templates. This simplifies the configuration process when connecting to multiple RADIUS servers. Accounting Send-Stop Setup-Failure improves session management by sending an accounting stop record in case of session setup failures, ensuring that the RADIUS server clears stale session entries.

Utilizing Session accounting AAA profiles for Service Accounting

This feature allows you to efficiently manage Authentication, Authorization, and Accounting (AAA) profiles when a Control Plane (CP) is connected to multiple RADIUS servers. Normally, setting up multiple AAA profiles for authorization, session, and service accounting requires an increase in feature templates, which can complicate configuration and management. This feature aims to streamline the process by allowing the use of session accounting AAA profiles for service accounting, thereby reducing the number of necessary feature templates.

Accounting Send-Stop Setup-Failure

The RADIUS server creates a session context when authorization and authentication are successful. If the session fails to establish, a stale session context may remain on the RADIUS server. To clear this, the cnBNG Control Plane sends an accounting stop record in the event of a session setup failure. This stop record is sent if the subscriber profile is configured with the **send-stop** option.

Configure Session Accounting AAA Profiles for Service Accounting

Procedure

Step 1 Configure AAA profiles to be used for service accounting.

Example:

```
config
  profile aaa aaa_name
    authentication
```

```
      method-order custom_server_group
      exit
    authorization
      password password
      type subscriber method-order custom_server_group
      username value value
      password password
      exit
    accounting
      method-order custom_server_group
      exit
    exit
```

Here is a sample configuration for creating AAA profiles named **east-aaaprofile** and **west-aaaprofile.**

```
profile aaa east-aaaprofile
 authentication
  method-order server-group1
 exit
 authorization
  type subscriber method-order server-group1
  username value cnbng
  password cisco
 exit
 accounting
  method-order server-group1
 exit
exit
```

```
profile aaa west-aaaprofile
 authentication
  method-order server-group2
 exit
 authorization
  type subscriber method-order server-group2
  username value cnbng
  password cisco
 exit
 accounting
  method-order server-group2
 exit
exit
```

Step 2 Configure session accounting AAA profiles in feature templates.

Example:

```
Config
  profile feature-template basetemplate1
   session-accounting
     enable
     aaa-profile east-aaaprofile
    periodic-interval 2000
  exit

  profile feature-template basetemplate2
   session-accounting
     enable
     aaa-profile west-aaaprofile
```

```
   periodic-interval 2000
  exit

  profile feature-template plan-100Mbps
   qos
    in-policy inpolicy
    out-policy outpolicy
   exit
    service-accounting
     enable
     aaa-profile use-session-aaaprofile
     periodic-interval 2000
  exit
 exit
```

According to this sample configuration, subscribers using **basetemplate1** with the **plan-100Mbps** feature template will use **east-aaaprofile** for service accounting. Conversely, subscribers using **basetemplate2** with the **plan-100Mbps** feature template will use **west-aaaprofile** for service accounting.

Configure Accounting Send-Stop Setup-Failure

Procedure

Configure the subscriber profile to enable the sending of accounting stop records upon session bring-up failures.

Example:

```
configure
  profile subscriber subsprofile1
   accounting send-stop setup-failure aaa-profile aaa_profile_name
  exit
```

If the AAA_Profile_Name is	then
use-author-profile	the AAA profile configured for authorization is used for sending the accounting stop record in case of session bringup failure. This configuration is used for IPoE sessions.
use-authen-profile	the AAA profile configured for authentication is used for sending the accounting stop record in case of session bringup failure. This configuration is used for PPPoE, LAC or LNS sessions.

This configuration is crucial for managing IPoE sessions, as well as PPPoE, LAC, or LNS sessions, ensuring that session contexts are appropriately cleared from the RADIUS server.

RADIUS Accounting Message Handling

Table 7: Feature History

Feature Name	Release Information	Description
RADIUS Accounting Message Handling	2024.03.0	We have enhanced the system performance by preventing packet identifier (PID) exhaustion with the new Asynchronous mode for sending RADIUS Accounting messages. The cnBNG now releases PIDs after dispatching messages, rather than waiting for a long time to receive server responses. This change allows for PID reuse, mitigating scale issues and improving KPIs.

The cnBNG currently sends RADIUS Accounting messages to servers in synchronous mode. It reserves a packet identifier (PID) for each message. The PID is only released when a response arrives from the server. If the server fails to respond promptly, the cnBNG attempts to resend the message. The PID remains reserved until all timeouts and retransmissions are complete. During periods of high traffic, this can lead to PID exhaustion. As a result, cnBNG may drop new incoming RADIUS messages due to the lack of available PIDs, leading to a decrease in Key Performance Indicators (KPIs).

Asynchronous RADIUS Accounting Messages

To improve performance, cnBNG can switch to asynchronous mode for sending RADIUS Accounting messages. In this mode, cnBNG does not wait for a server response. After sending a message, the cnBNG waits for a timeout of one second, and then releases the PID. This allows the PID to be reused for subsequent messages. However, a brief delay is implemented before reusing a PID. This change can prevent PID exhaustion during high traffic and enhance KPIs. Note that the asynchronous mode applies only to RADIUS Accounting messages.

Configure Asynchronous RADIUS Accounting

Configure Asynchronous mode for Session Accounting

Use this configuration to enable asynchronous mode for session accounting.

```
config
  profile aaa aaa_name
    accounting
      session { acct-interim-async true/false | acct-start-async true/false |
acct-stop-async true/false }
        commit
```

Configure Asynchronous mode for Service Accounting

Use this configuration to enable asynchronous mode for service accounting.

```
config
  profile aaa aaa_name
    accounting
      service { acct-interim-async true/false | acct-start-async true/false |
```

```
acct-stop-async true/false }
      commit
```

Note If asynchronous RADIUS accounting is configured, message retransmission functionality will be unavailable. Lost messages due to network issues or server errors will not be retried by the Control Plane.

NOTES:

- **profile aaa** *aaa_name*: Specifies the AAA profile name and enters the AAA configuration mode.
- **accounting**: Enters the accounting sub-mode.
- **session [service] acct-interim-async** *true/false* : Specifies the option to enable or disable the asynchronous mode when interim updates are sent.
- **session [service] acct-start-async** *true/false*: Specifies the option to enable or disable the asynchronous mode when an Accounting-Start request is sent to AAA.
- **session [service] acct-stop-async** *true/false*: Specifies the option to enable or disable the asynchronous mode when an Accounting-Stop request is sent.

Standard Compliance

The AAA features are aligned with the following standards:

- RFC 2865 - Remote Authentication Dial In User Service (RADIUS)
- RFC 2866 - RADIUS Accounting
- RFC 5176 - Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)

RADIUS Automated Testing

For subscriber authentication and accounting, the cnBNG-CP chooses a RADIUS server from the list of configured servers, and sends RADIUS messages. If the RADIUS server is unreachable, it is considered dead, and is excluded from the selection algorithm until a 'dead-timer' expires. After this timer expires, the dead server is re-included in the selection list without checking if it is now reachable. This causes the cnBNB-CP to retransmit messages to the still-unreachable server, causing retransmissions and delays, which negatively impact Key Performance Indicators (KPIs).

The RADIUS Automated Testing feature allows the cnBNG-CP to periodically check the status of the RADIUS server until the server is considered dead or the dead-timer expires. With this feature, if the dead-timer expires, the cnBNG-CP attempts to send authentication and accounting TEST messages to the RADIUS server that is currently unreachable. If the server does not respond to these messages, it is marked as dead, and this process continues until the server is reachable. If the RADIUS server responds within the set number of retransmissions and timeouts, it is marked as available, and is then included in the selection algorithm list that is used to choose RADIUS servers.

There are two configuration scenarios:

- The following happens when RADIUS Automated Testing is enabled without the Idle-timer functionality:

  When the dead-detection criteria is met, the dead-timer starts with the value configured for **deadtime** CLI. Whenever the dead-timer expires, TEST messages are sent to the RADIUS server to check if the server is reachable. If the RADIUS server responds, the dead-timer stops, and the RADIUS server is marked UP. If the RADIUS server doesn't respond, the dead-timer is restarted.

- The following happens when the **idle-timer** CLI along with **auto-test enable** is configured:

  When the RADIUS server is UP, the status of the server is checked periodically by sending TEST RADIUS messages to the selected RADIUS server as per the configured idle-timer value. If the RADIUS server doesn't respond to the TEST messages, then the RADIUS server is marked as dead. The dead-timer starts as per the value configured in the **deadtime** CLI, and the periodic TEST message is stopped. Once the dead-timer expires, TEST RADIUS messages are sent to the RADIUS server, and the dead-timer is restarted. If the server responds to the TEST messages, the server is marked as UP, the dead-timer is stopped, and periodic TEST messages are restarted. If the server does not respond to TEST messages, on every dead-timer expiry, TEST RADIUS messages are sent to check reachability.

Restrictions

- RADIUS server availability based on VRF is not supported.
- **Show peers** command is not instance-aware.
- Round-trip time (RTT) is based on the server level, not the instance level.
- Server status is maintained per instance. All other statuses are maintained at a global level.
- Presently, only two VIPs are supported.

Software Upgrades

When you upgrade your current software (release version prior to cnBNG 2024.01) to cnBNG 2024.01 and later releases, perform the following steps:

- Fetch the keys of the RADIUS server from ETCD using the following command:

  ```
  kubectl exec -it -n bng <etcd pod name> -- etcdctl get --prefix "" --keys-only | grep serv
  ```

- Delete the key from ETCD using the following command:

  ```
  kubectl exec -it -n bng <etcd pod name> -- etcdctl del key “serverkey”
  ```



Note Rolling upgrade is not supported for this software upgrade scenario.

Configure RADIUS Automated Testing

This section describes how to enable RADIUS Automated Testing feature for accounting and authorization.

```
config
  profile radius
    server ipv4_address port_number
```

```
type { acct | auth }
auto-test enable
auto-test enable idle-timer number
commit
```

To disable the RADIUS Automated Testing feature, use the **no auto-test enable** command, and to disable the idle-timer functionality, use the **no auto-test idle-timer** command.

Note When the RADIUS Automated Testing feature is disabled, the status of the RADIUS server is reset to UP (reachable).

NOTES:

- **profile radius**: Enters the RADIUS configuration mode.
- **server** *ipv4_address port_number*: Specifies the IPv4 address and port of the RADIUS server.
- **type { acct | auth }**: Specifies the type of the RADIUS server. It can be one of the following:
  - **acct**: RADIUS server used for the accounting requests
  - **auth**: RADIUS server used for the authentication requests
- **auto-test enable**: Enables RADIUS automated testing.
- **auto-test enable idle-timer** *minutes*: Enables the idle-timer functionality. *minutes* value ranges from 1 through 30.
- **commit**: Commits the configuration.

Verifying RADIUS Automated Testing

Use the following **show** commands to verify the RADIUS Automated Testing feature.

```
bng# show radius auth-server 10.1.45.112:1812
Mon Nov  6  11:01:25.394 UTC+00:00
-----------------------------------------------------------
 Server: 10.1.45.112, port: 1812, status: instance 1: up , port-type: Auth
 75 requests, 0 pending, 0 retransmits
 0 accepts, 75 rejects, 0 timeouts
 0 bad responses, 0 bad authenticators
 0 unknown types, 0 dropped, 1004 ms latest rtt
 Auto Test Stats:
 instance 1:
  Requests:74 Pending:0 Retransmits:0
  Rejects:74 Timeouts:0 Responses:0
  BadAuth:0 Dropped:0 BadResp:0
-----------------------------------------------------------

bng# show radius acct-server 10.1.45.112:1813
Mon Nov  6  11:22:58.330 UTC+00:00
-----------------------------------------------------------
 Server: 10.1.45.112, port: 1813, status: instance 1: up , port-type: Acct
 338 requests, 0 pending, 0 retransmits
 338 responses, 0 timeouts
 0 bad responses, 0 bad authenticators
 0 unknown types, 0 dropped, 3 ms latest rtt
```

```
Auto Test Stats:
instance 1:
 Requests:337 Pending:0 Retransmits:0
 Rejects:0 Timeouts:0 Responses:337
 BadAuth:0 Dropped:0 BadResp:0
```

Clear RADIUS operational statistics

Clearing RADIUS operational statistics is a management function that

- resets selected operational counters for RADIUS authentication, accounting, and dynamic authorization (CoA or Disconnect-Message) servers or clients
- enables administrators to perform targeted or comprehensive resets based on server role, IP address, and port, and
- requires explicit user confirmation to ensure intentional execution.

Table 8: Feature history

Feature Name	Release Information	Description
Clear RADIUS operational statistics	2026.01.0	You can simplify RADIUS server monitoring and troubleshooting by quickly resetting operational statistics as needed. This feature allows you to clear statistics for specific or all servers and clients, with a confirmation prompt to prevent accidental resets.

This feature allows you to selectively clear operational statistics for all RADIUS servers, all authentication servers, all accounting servers, specific authentication or accounting servers by IP and port, all dynamic authorization clients, or specific clients by IP address. Clearing statistics does not affect the UP/DOWN status of any RADIUS server.

Note that some counters, such as pending or in-process requests, are not reset by these commands.

Clear RADIUS server statistics

Reset authentication and accounting statistics for RADIUS servers to aid in operational troubleshooting.

Clearing counters can help operators monitor current activity and isolate issues.

Procedure

Step 1 Use the appropriate CLI command to clear statistics. These options are available:

Command	Description
clear radius all	Clear statistics for all RADIUS servers.
clear radius authentication all	Clear all authentication servers.

Command	Description
clear radius authentication server *ip* port *port*	Clear a specific authentication server. For example, `clear radius authentication server 192.168.1.100 port 1812`
clear radius accounting all	Clear all accounting servers.
clear radius accounting server *ip* port *port*	Clear a specific accounting server. For example, `clear radius accounting server 192.168.1.100 port 1812`

Step 2 When prompted with **This will clear RADIUS server statistics. Continue? [yes/no]:** type **`yes`** to confirm the operation. If you do not confirm, the operation is canceled.

Step 3 Use the **show radius** command to verify the server's operational status before and after clearing statistics.

Example:

Before clearing statistics

```
bng# show radius
-------------------------------------------------------
 Server: 10.105.254.43, port: 1842, status: instance 1: up  , port-type: Auth
 10 requests, 0 pending, 0 retransmits
 5 accepts, 5 rejects, 0 timeouts
 0 bad responses, 0 bad authenticators
 0 unknown types, 0 dropped, 39 ms latest rtt
 Auto Test Stats:
 instance 1:
  Requests:0 Pending:0 Retransmits:0
  Rejects:0 Timeouts:0 Responses:0
  BadAuth:0 Dropped:0 BadResp:0
```

Example:

After clearing statistics

```
bng# show radius
-------------------------------------------------------
 Server: 10.105.254.43, port: 1842, status: instance 1: up  , port-type: Auth
 0 requests, 0 pending, 0 retransmits
 0 accepts, 0 rejects, 0 timeouts
 0 bad responses, 0 bad authenticators
 0 unknown types, 0 dropped, 0 ms latest rtt
 Auto Test Stats:
 instance 1:
  Requests:0 Pending:0 Retransmits:0
  Rejects:0 Timeouts:0 Responses:0
  BadAuth:0 Dropped:0 BadResp:0
```

RADIUS statistics for the selected servers are reset to zero, except for `pending` or `inProcess` counters.

Clear dynamic authorization client statistics

Reset statistics for dynamic authorization (CoA or Disconnect-Message) clients.

Procedure

Step 1 Use the **clear radius-dyn-auth all** command to clear all clients.

Step 2 Use the **clear radius-dyn-auth client** *ip-address* command to clear a specific client.

Step 3 When prompted with **This will clear RADIUS server statistics. Continue? [yes/no]:**, type **yes** to confirm the operation. If you do not confirm, the operation is canceled.

Step 4 Use the **show radius-dyn-auth** command to verify the operational status of the client before and after clearing statistics.

Example:

Before clearing statistics

```
bng# show radius-dyn-auth
------------------------------------------------------------
------------------------------------------------------------
 IP: 10.1.2.105
 ------------------
 COA:
 10 total-requests       0 inprocess-requests
     0 retry-request-drops  0 invalid-requests
     0 bad-authenticators   0 internal-errors
 5 ack-sent             5 nak-sent
 ------------------
 DISCONNECT:
 20 total-requests       0 inprocess-requests
     0 retry-request-drops  0 invalid-requests
     0 bad-authenticators   0 internal-errors
 10 ack-sent             10 nak-sent
 ------------------
 UnknownTypesRcvd: 0
```

Example:

After clearing statistics

```
bng# show radius-dyn-auth
------------------------------------------------------------
IP: 10.1.2.105
 ------------------
 COA:
 0 total-requests       0 inprocess-requests
     0 retry-request-drops  0 invalid-requests
     0 bad-authenticators   0 internal-errors
 0 ack-sent             0 nak-sent
 ------------------
 DISCONNECT:
 0 total-requests       0 inprocess-requests
     0 retry-request-drops  0 invalid-requests
     0 bad-authenticators   0 internal-errors
 0 ack-sent             0 nak-sent
 ------------------
 UnknownTypesRcvd: 0
```

Dynamic authorization statistics are reset to zero for the selected clients, except for `inprocess-requests`.

RADIUS Attributes Filtering

Table 9: Feature History

Feature Name	Release Information	Description
RADIUS Attributes Filtering	2025.01.0	You can now customize RADIUS messages by including or excluding specific attributes to meet your server's requirements. This feature enhances your control over shared data between RADIUS clients and servers and optimizes message performance by filtering unnecessary attributes, thus reducing server load.

The RADIUS Attributes Filtering feature allows you to manage which RADIUS attributes are included or excluded from RADIUS messages. This is achieved through CLI commands that specify which IETF and vendor-specific attributes should be filtered. This feature applies only to RADIUS Authentication and Accounting Request and Reply message types, and does not affect CoA and Disconnect messages originated by the AAA Client.

This feature does not control the behavior of the AAA server. Instead, it manages how attributes are included or excluded in outbound messages based on specific actions:

- **Request Processing**: During the **Request** phase, the system processes Accept or Reject actions to determine the inclusion or exclusion of specific attributes in outbound Access-Request and Accounting-Request messages (such as START, INTERIM, or STOP messages) as necessary.
- **Reply Processing**: In the **Reply** phase, even if the AAA Server sends specific attributes, the cnBNG Control Plane (CP) uses its predefined configurations to determine whether to accept or reject these attributes. These configurations guide the cnBNG CP in deciding whether to apply or ignore the attribute values for sessions. This process ensures that attributes are correctly applied without modifying the original messages from the AAA Server.

Restrictions and Guidelines for RADIUS Attributes Filtering

These restrictions apply to the RADIUS Attributes Filtering feature:

- This feature is not supported for CoA and Disconnect messages originated by the AAA Client.
- This feature is only applicable to Authentication and Accounting Request and Reply message types.
- Attribute filtering is not supported for global RADIUS configurations.
- Multiple attribute lists can be configured, but only one attribute list per request and reply can be attached. Configuring multiple lists per request or reply for authentication and accounting is not possible.
- If there is a mismatch between the server-group list name attached to request/reply and the one configured in RADIUS server attribute configuration, the list is considered invalid, and attribute filtering will function as PASS. Similarly, if the list is not present, the feature will work as PASS.
- There is no automatic validation for the number of attributes or the names of Vendor Specific Attributes (VSAs). You must ensure that the IETF or numbered attributes are configured correctly and VSA names are spelled accurately. The Ops Center does not check the attribute list values for correctness.

- No change is introduced by this feature in how monitor protocol and monitor subscriber work.
- No change is introduced by this feature in other profile radius CLIs.

Configure RADIUS Attributes Filtering

Procedure

Step 1 Define the IETF and vendor-specific attributes in lists using the **profile radius** configuration.

Example:

```
configure
  profile radius
    radius-server
     attribute attr-list list_name
     ietf-attributes ietf_attributes
     vendor-attribute vendor-id vendor_id vendor-type vendor_type_value name [ addrv6
client-mac-address ]
    exit
```

The following is a sample configuration:

```
profile radius
 radius-server
  attribute attr-list list1
   ietf-attributes [ 88 100 30 2 ]
   vendor-attribute vendor-id 9 vendor-type 1 name [ addrv6 client-mac-address ]
   vendor-attribute vendor-id 9 vendor-type 56
   vendor-attribute vendor-id 9 vendor-type 60
   vendor-attribute vendor-id 3561 vendor-type 1
  exit
  attribute attr-list list2
   ietf-attributes [ 40 42 47 61 88 100 ]
   vendor-attribute vendor-id 9 vendor-type 56
   vendor-attribute vendor-id 3561 vendor-type 1
  exit
 exit
```

NOTES:

- **profile radius**: Enters the RADIUS configuration mode.
- **radius-server attribute attr-list** *list_name*:Specifies the RADIUS server attributes list.
- **ietf-attributes** *ietf_attributes*: Configures the list of Internet Engineering Task Force (IETF) attributes. For example, **ietf-attributes [88 8 30 2]** indicates that IETF attributes such as Framed-IP-Pool (88), Framed-IP-Address (8), Called-Station-Id(30), User-Password(2) are configured.
- **vendor-attribute vendor-id** *vendor_id*: Configures RADIUS attribute filtering for Vendor Specific Attributes (VSAs) by specifying vendor information such as the vendor-id in the RADIUS attribute list.
- **vendor-type** *value*: Configures the vendor specific information such as the vendor-type in the RADIUS attribute list.

Note For **vendor-type** *1* and **vendor-type** *9*, attribute values are specified in the name list.

For example, **vendor-attribute vendor-id 9 vendor-type 56** configures Cisco vendor attributes `cisco-vsa-sub-qos-pol-out`, and `cisco-vsa-sub-activate-service` with vendor-id `9` and vendor-type `56`.

vendor-attribute vendor-id 3561 vendor-type 56 configures the non-Cisco vendor attribute `Agent-Circuit-Id` with vendor-id `3561` and vendor type `1`.

- **name**: [Optional] Specifies the attribute name for a Cisco generic Vendor Specific Attribute (VSA).

  The **name** option is only available for vendor-id 9 and vendor-type 1.

Note For vendor-type 1, if names are not configured, the accept or reject action applies to all vendor-type attributes of the configured vendor ID.

- **addrv6 client-mac-address**: [Optional] Specifies the client MAC address in AABB.CCDD.EEFF format.

Step 2 Configure the accept attributes list for both request and reply messages in RADIUS authentication and accounting.

Example:

```
config
  profile radius
   server-group server_group_name
   accounting
    request accept list_1_name
    reply accept list_1_name
   exit
   authentication
    request accept list_2_name
    reply accept list_2_name
   exit
   server auth ip_address port_number
   exit
   server acct ip_address port_number
   commit
```

Step 3 Configure the reject attributes list for both request and reply messages in RADIUS authentication and accounting.

Example:

```
config
  profile radius
   server-group server_group_name
   accounting
    request reject list_1_name
    reply reject list_1_name
   exit
   authentication
```

```
 request reject list_2_name
 reply reject list_2_name
exit
server auth ip_address port_number
exit
server acct ip_address port_number
commit
```

NOTES:

- **server-group** *server_group_name*: Specifies the profile server group name to enter the Profile Server Group configuration mode.
- **accounting**: Enters the accounting sub-mode.
  - **request accept** *list_1_name*: Configures the accept attributes list for request messages in RADIUS accounting.
  - **reply accept** *list_1_name*: Configures the accept attributes list for reply messages in RADIUS accounting.
  - **request reject** *list_1_name*: Configures the reject attributes list for request messages in RADIUS accounting.
  - **reply reject** *list_1_name*: Configures the reject attributes list for reply messages in RADIUS accounting.
- **authentication**: Enters the authentication sub-mode.
  - **request accept** *list_1_name*: Configures the accept attributes list for request messages in RADIUS authentication.
  - **reply accept** *list_1_name*: Configures the accept attributes list for reply messages in RADIUS authentication.
  - **request reject** *list_1_name*: Configures the reject attributes list for request messages in RADIUS authentication.
  - **reply reject** *list_1_name*: Configures the reject attributes list for reply messages in RADIUS authentication.
- **server auth** *ip_address port_number*: Configures the authentication server.
- **server acct** *ip_address port_number*: Configures the accounting server.

Step 4 Use the **monitor protocol interface radius pcap yes** command to display and capture data packets from RADIUS interface messages.

NOTES:

- **pcap yes**: Enables PCAP file generation. By default, it is set to **no**.

Step 5 Use the **monitor subscriber supi** *supi_id* **capture-duration** *duration* **internal-messages yes** command to enable subscriber monitoring based on subscriber identity (SUPI).

NOTES:

- **supi** *supi_id*: Specifies the subscriber identity. For example, *aa11.0000.0001*, where *aa11.0000.0001* is the mac-id.
- **capture-duration** *duration*: [Optional] Specifies the duration, in seconds, for which the monitor subscriber is enabled. The default duration is 300 seconds.
- **internal-messages yes**: Enables internal messages, which are disabled by default.

Framed Route with tag and preference support

A framed route is a routing attribute that

- allows you to define specific IP prefixes and next hops for subscriber sessions
- enables the use of additional attributes such as administrative distance (preference) and tag values, and
- supports both IPv4 and IPv6 address families for flexible routing.

Table 10: Feature History

Feature Name	Release Information	Description
Framed Route with tag and preference support	2025.03.0	You can now prioritize and control subscriber traffic by assigning static routes with preference and tag values during authentication. By allowing the assignment of IPv4 or IPv6 framed routes through RADIUS attributes, this feature enables your network to automatically select the optimal path for both IPv4 and IPv6 subscribers.

Framed routes enable operators to define static routes via RADIUS attributes during subscriber authentication. These routes can include preference (admin distance) and tag values, allowing for prioritization and policy-based routing.

Examples:

- IPv4: 192.168.100.0/24 0.0.0.0 10 tag 123 pref 10
- IPv6: 2001:DB8::/64 2::1 10 tag 123 pref 10

Use case: Multi-homed subscriber routing scenarios

In multi-homed subscriber routing scenarios, a subscriber device connects to the broadband network using two or more access links, typically designating one as the primary (preferred) path and the other as a backup. This configuration is used to provide high availability and uninterrupted connectivity for subscribers.

Key characteristics:

- The subscriber device is anchored at two different BNGs, each associated with a separate access link (for example, fiber for primary and wireless for backup).
- Both BNGs advertise the subscriber's route to the core network, but each assigns a different administrative distance (preference value) to indicate priority.
- The core network uses the administrative distance or preference value to select the route for downstream traffic, ensuring the primary path is preferred whenever available.

Typical workflow:

1. The primary link (such as fiber) is used under normal circumstances for subscriber traffic.
2. If the primary link fails, the backup link (such as mobile broadband) automatically takes over, maintaining service continuity.

3. The administrative distance or preference value is dynamically set using RADIUS attributes provided by the AAA server, allowing flexible path prioritization.

Supported framed route formats

The cnBNG platform supports several variations of the Framed-Route and Framed-IPv6-Route attributes, including the use of tag and preference (admin distance) fields.

Format Example	Description
1. `192.168.100.0/24 0.0.0.0`	Basic static route
2. `192.168.100.0/24 0.0.0.0 10`	With metric
3. `192.168.100.0/24 0.0.0.0 10 tag 123`	With metric and tag
4. `192.168.100.0/24 0.0.0.0 10 tag 123 pref 100`	With metric, tag, and preference
5. `192.168.100.0/24 0.0.0.0 10 pref 100`	With metric and preference
6. `192.168.100.0/24 0.0.0.0 pref 100`	With preference only

The same flexibility applies for Framed-IPv6-Route attributes.

Backward compatibility considerations

This table outlines how different combinations of Control Plane (CP) and User Plane (UP) software versions affect framed route processing and subscriber session behavior.

CP version	UP version	Framed Route attributes	Behavior / outcome
New (2025.03.0 or newer)	Old (25.3.x or lower)	With `pref` and `tag`	UP cannot decode the preference (pref) attribute, but the session still establishes successfully. The UP installs the framed route with an administrative distance or metric of 1, and the pref attribute has no effect on the UP.
New (2025.03.0 or newer)	Old (25.3.x or lower)	Without `pref`/`tag`	CP and UP process normally. Session establishment succeeds.
Old (2025.02.0 or lower)	New (25.4.x or newer)	With `pref` and `tag`	CP cannot process; session establishment not attempted. Subscriber not connected.
Old (2025.02.0 or lower)	New (25.4.x or newer)	Without `pref`/`tag`	CP and UP process normally. Session establishment succeeds.

Standards compliance

The cnBNG framed route feature aligns with relevant industry standards:

- RFC 2865 (RADIUS – Framed-Route attribute for IPv4)
- RFC 3162 (RADIUS – Framed-IPv6-Route attribute for IPv6)
- Broadband Forum TR-459 (BNG requirements and interoperability)

Restrictions on Framed Route with tag and preference support

When deploying framed route preference and tag support on cnBNG, consider the following restrictions:

- **Maximum framed routes:**

  Each subscriber is limited to a maximum of 4 IPv4 framed routes and 4 IPv6 framed routes per address family.

- **Framed-Route attribute formats:**

  Only the supported attribute formats documented for cnBNG are accepted. Unsupported or invalid formats may cause route installation to fail.

- **Route gateway address:**

  If the gateway is specified as `0.0.0.0` (IPv4) or `::` (IPv6), the subscriber's IP address is used as the next-hop gateway by default.

- **Prefix length deduction:**

  If the prefix length is omitted in the framed route, the system infers the length based on the IP class (Class A: /8, Class B: /16, Class C: /24).

- **High availability:**

  Only the active UP installs routes; the standby UP holds the information but does not install until a switchover occurs.

- **Software version compatibility:**

  Full functionality requires both Control Plane (CP) and User Plane (UP) to be running compatible software versions that support framed route preference/tag attributes.

- **Accounting support:**

  Framed-Route attributes are not supported in RADIUS accounting messages on cnBNG.

- **Platform differences:**

  Feature support and limits may vary between different Cisco BNG platforms. Always verify compatibility when planning mixed deployments.

Configure framed route preference support

Enable and customize framed route administrative distance (preference) handling on cnBNG.

Use this task to specify how cnBNG interprets the metric or preference values in framed route attributes.

Before you begin

Ensure the system is running a software version that supports this feature.

Follow these steps to configure framed route preference support:

Procedure

Step 1 By default, cnBNG uses the **pref** value as the administrative distance for framed routes. However, if you apply this configuration on the User Plane, the metric value received in the framed-route will be used as the administrative distance instead.

Example:

```
configure
 cnbng-nal location 0/RSP0/CPU0
  framed-route-metric-as-distance
!
```



Note Apply this configuration when there are no subscriber sessions present on the router.

Step 2 You do not need to execute any commands on the Control Plane to configure this feature.

Step 3 Use the **show subscriber session detail command** command on the Control Plane to view the framed route.

Example:

```
bng# show subscriber session detail | more
Thu Jun  12 04:00:23.852 UTC+00:00
subscriber-details
{
  "subResponses": [
      <snip>
      "subcfgInfo": {
        "committedAttrs": {
          "attrs": {
            "accounting-list": "automation-aaaprofile",
            "acct-interval": "2000",
            "addr-pool": "automation-poolv4",
            "ipv4-mtu": "1400",
           "ipv6-route": "vrf prefix-vrf 2001:db8:1::/64 vrf gw-vrf 2001:db8:100::1 100 tag 101 pref
 200,2001:db8:2::/64 2001:db8:100::2 101 tag 12 pref 200,2001:db8:3::/64 0:0:0:0:0:0:0:0 pref 300
tag 444 199,2001:db8:4::/64 :: 103 tag 12 pref 400",
            "ppp-ipcp-reneg-ignore": "true",
            "ppp-ipv6cp-reneg-ignore": "true",
            "ppp-lcp-reneg-ignore": "true",
            "route": "vrf prefix-vrf 192.168.1.0/24 vrf gw-vrf 192.168.1.1 100 tag 101 pref
200,192.168.3.0/24 192.168.3.100 101 pref 200 tag 102,172.10.1.0 172.10.1.1 pref 200 tag 102
405,10.10.1.0 10.10.1.1 tag 102 555 pref 399",
            "session-acct-enabled": "true",
            "vrf": "automation-vrf"
          }
        },
        "activatedServices": [
          <snip>
        ]
      },
      <snip>
      "v4FramedRoute": [
        "vrf prefix-vrf 192.168.1.0/24 vrf gw-vrf 192.168.1.1 100 tag 101 pref 200",
        "192.168.3.0/24 192.168.3.100 101 pref 200 tag 102",
        "172.10.1.0 172.10.1.1 pref 200 tag 102 405",
```

```
      "10.10.1.0 10.10.1.1 tag 102 555 pref 399"
    ],
    "v6FramedRoute": [
      "vrf prefix-vrf 2001:db8:1::/64 vrf gw-vrf 2001:db8:100::1 100 tag 101 pref 200",
      "2001:db8:2::/64 2001:db8:100::2 101 tag 12 pref 200",
      "2001:db8:3::/64 0:0:0:0:0:0:0:0 pref 300 tag 444 199",
      "2001:db8:4::/64 :: 103 tag 12 pref 400"
    ],
    <snip>
```

Step 4 Use the **show cnbng-nal subscriber all detail** command on the User Plane to view the tag and admin distance in a framed-route session.

Example:

```
Router# show cnbng-nal subscriber all detail
IPv4 Framed Route:
  Prefix:                  192.168.1.0/24
  Next Hop:                192.168.1.1
  Tag:                     101
  Metric:                  100
  Distance:                200
  Next Hop VRF:            abc
  Prefix:                  192.168.2.0/24
  Next Hop:                192.168.2.1
  Tag:                     101
  Metric:                  100
  Distance:                200
  Next Hop VRF:            abc
IPv6 IANA Address:         2001:DB8::7002
IPv6 IAPD Prefix:          ::/0
IPv6 Slaac Prefix:         ::/0
CPE link local Address:    ::
IPv6 Framed Route:
  Prefix:                  2001:DB8:4004:800::/64
  Next Hop:                ::
  Tag:                     101
  Metric:                  120
  Distance:                71
  Next Hop VRF:            abc
  Prefix:                  2001:DB8:4700::/64
  Next Hop:                2001:DB8:35::35
  Tag:                     23
  Metric:                  99
  Distance:                0
  Next Hop VRF:            abc
```

Configuring AAA Functions

This section describes how to configure the following Authentication, Authorization, and Accounting (AAA) functions on the Control Plane (CP).

The configuration of the AAA functions involves the following procedures:

- Configuring AAA Attributes
- Configuring the CoA-NAS Interface
- Configuring Method Order for AAA

- Configuring RADIUS Accounting Options
- Configuring RADIUS Accounting Server Group
- Configuring RADIUS Attributes
- Configuring RADIUS-Dead Time
- Configuring RADIUS-Detect Dead Server
- Configuring RADIUS Pod
- Configuring RADIUS Maximum Retry
- Configuring RADIUS NAS-IP
- Configuring RADIUS Server
- Configuring RADIUS Server Group
- Configuring RADIUS Server Selection Logic
- Configuring RADIUS Timeout
- Configuring Radius PID-Quarantine

Configuring AAA Attributes

Use the following commands to configure a function for the AAA attribute format.

```
config
  profile attribute-format attribute_format_name
    format-order { addr | circuit-id-tag | client-mac-address |
      addr | circuit-id-tag | client-mac-address |
      client-mac-address-custom1 | client-mac-address-custom2 |
      client-mac-address-ietf | client-mac-address-raw |
      dhcp-client-id | dhcp-client-id-spl | dhcp-user-class |
      dhcp-vendor-class | dhcpv4-client-id-spl |
      dhcpv4-vendor-class | dhcpv6-client-id-ent-ident |
      dhcpv6-interface-id | dhcpv6-vendor-class-string |
      inner-vlan-id | outer-vlan-id | physical-adapter |
      physical-chassis | physical-port | physical-slot |
      physical-subslot | port-type | pppoe-session-id |
      remote-id-tag | service-name | user-plane | username }
  format-string format_string
  commit
```

NOTES:

- **profile attribute-format** *attribute_format_name*: Specifies the AAA attributes and enters the Attribute Format Configuration mode.
- **authorization**: Enters the Authorization sub-mode.
- **format-order** *attribute_format* **| identifier { addr | circuit-id-tag | client-mac-address | client-mac-address-custom1 | client-mac-address-custom2 | client-mac-address-ietf | client-mac-address-raw | dhcp-client-id | dhcp-client-id-spl | dhcp-user-class | dhcp-vendor-class |**

dhcpv4-client-id-spl | dhcpv4-vendor-class | dhcpv6-client-id-ent-ident | dhcpv6-interface-id | dhcpv6-vendor-class-string | inner-vlan-id | outer-vlan-id | physical-adapter | physical-chassis | physical-port | physical-slot | physical-subslot | port-type | pppoe-session-id | remote-id-tag | service-name | username } | value *value* **}**: Specifies the AAA attribute format order as follows:

- **addr**: Specifies the IPv4 address of the subscriber.
- **circuit-id-tag**: Specifies the circuit identifier tag.
- **client-mac-address**: Specifies the client MAC address in AABB.CCDD.EEFF format.
- **client-mac-address-custom1**: Specifies the first custom client MAC address in AABB.CCDD.EEFF format.
- **client-mac-address-custom2**: Specifies the second custom client MAC address in AABB.CCDD.EEFF format.
- **client-mac-address-ietf**: Specifies the client MAC address in Internet Engineering Task Force (IETF) format. That is, AA-BB-CC-DD-EE-FF format.
- **client-mac-address-raw**: Specifies the client MAC address in raw (AABBCCDDEEFF) format.
- **dhcp-client-id**: Specifies the DHCP client identifier.
- **dhcp-client-id-spl**: Specifies the DHCP client identifier special string.
- **dhcp-user-class**: Specifies the DHCP user class.
- **dhcp-vendor-class**: Specifies the DHCP vendor class.
- **dhcpv4-client-id-spl**: Specifies the DHCPv4 client identifier special string.
- **dhcpv4-vendor-class**: Specifies the DHCPv4 vendor class.
- **dhcpv6-client-id-ent-ident**: Specifies the DHCPv6 client and enterprise identifiers.
- **dhcpv6-interface-id**: Specifies the DHCPv6 interface identifier.
- **dhcpv6-vendor-class-string**: Specifies the DHCPv6 vendor class string.
- **inner-vlan-id**: Specifies the inner VLAN identifier.
- **outer-vlan-id**: Specifies the outer VLAN identifier.
- **physical-adapter**: Specifies the physical adapter.
- **physical-chassis**: Specifies the physical chassis.
- **physical-port**: Specifies the physical port.
- **physical-slot**: Specifies the physical slot.
- **physical-subslot**: Specifies the physical subslot.
- **port-type**: Specifies the interface or port type.
- **pppoe-session-id**: Specifies the PPPoE physical identifier.
- **remote-id-tag**: Specifies the remote identifier tag.
- **service-name**: Specifies the service name.

- **user-plane**: Specifies the User Plane (UP).
- **username**: Specifies the username.

- **format-string** *format_string*: Specifies the AAA format pattern. The *format_string* specifies the format string. Each identifier is represented by '%s' tuple. Any other character set is treated as a delimiter. For each '%s' in the format-string, the format-order identifier is used.

Note

- Validation on the number of '%s' in format-string and number of entries in format-order are not performed.
- For backward compatibility, the format-order still takes the delimiter configuration. In this scenario, the format-order takes precedence and the format-string is silently ignored.
- Use the delimiters either in the format-order (as in Release 2021.01) or in format-string (as in Release 2021.03).

Configuring the CoA-NAS Interface

Use the following configuration to define Change of Authorization (CoA) NAS interface in the RADIUS endpoint.

```
config
   endpoint radius
      interface coa-nas
         vip-ip ipv4_address vip-port port_number
         vip-ipv6 ipv6_address vip-ipv6-port port_number
         end
```

NOTES:

- **endpoint radius**: Enters the RADIUS endpoint configuration mode.
- **interface coa-nas**: This keyword defines a new interface "coa-nas", and allows to enter the CoA NAS interface configuration mode.
- **vip-ip** *ipv4_address* **vip-port** *port_number*: Configures the IPv4 address of the host. *ipv4_address* must be in standard IPv4 dotted decimal notation.

  You can configure a list of VIP-IPs to listen to the inbound CoA or DM requests.

  vip-port *port_number*: Specify the port number of the UDP proxy. By default, the port number is 3799. This default value is used only when the VIP-IP is specified.

Important This configuration allows only port to be specified per IP.

The BNG (udp-pxy) listens to the inbound CoA or DM request messages on these ports and ACK or NAK messages sent with the respective source IP and port.

- **vip-ipv6** *ipv6_address* **vip-ipv6-port** *port_number*: Configures the IPv6 address of the host.

  vip-ipv6-port*port_number*: Specify the port number of the UDP proxy.

Configuring Method Order for AAA

Use the following commands to assign the method order for the server group to use for subscriber authentication, authorization, and accounting.

Authentication

```
config
  profile aaa aaa_name
    authentication
      method-order custom_server_group
      commit
```

NOTES:

- **profile aaa** *aaa_name*: Specifies the AAA profile name and enters the AAA Configuration mode.
- **authentication**: Enters the Authentication sub-mode.
- **method-order** *custom_server_group*: Specifies the method-order to be applied by default for subscriber authentication.

  custom_server_group specifies the name of the server group where the method-order is applied.

Authorization

```
config
  profile aaa aaa_name
    authorization
      password password
      enc-password password
      type subscriber method-order custom_server_group
     username { format attribute_format | identifier { addr | circuit-id-tag
 | client-mac-address | client-mac-address-custom1 |
client-mac-address-custom2 | client-mac-address-ietf |
client-mac-address-raw | dhcp-client-id | dhcp-client-id-spl |
dhcp-user-class | dhcp-vendor-class | dhcpv4-client-id-spl |
dhcpv4-vendor-class | dhcpv6-client-id-ent-ident | dhcpv6-interface-id |
 dhcpv6-vendor-class-string | inner-vlan-id | outer-vlan-id |
physical-adapter | physical-chassis | physical-port | physical-slot |
physical-subslot | port-type | pppoe-session-id | remote-id-tag |
service-name | username } | value value }
      commit
```

NOTES:

- **profile aaa** *aaa_name*: Specifies the AAA profile name and enters the AAA Configuration mode.
- **authorization**: Enters the Authorization sub-mode.
- **password** *password* : Specifies the password for subscriber authentication.

- **enc-password** *password* : Specifies the password for subscriber authentication.

  The enc-password value is stored in an encrypted format. The show command displays the password as an encrypted string. After upgrading to the 2026.04.0 release, the plain-text password is restored from the previous configuration. If needed, the user can convert the password to the encrypted format.

  The password and enc-password options are mutually exclusive. You can configure only one at a time.

- **type subscriber method-order** *custom_server_group*: Specifies the method-order to be applied by default for subscriber authorization.

  custom_server_group specifies the name of the server group where the method-order is applied.

- **username { format** *attribute_format* **| identifier { addr | circuit-id-tag | client-mac-address | client-mac-address-custom1 | client-mac-address-custom2 | client-mac-address-ietf | client-mac-address-raw | dhcp-client-id | dhcp-client-id-spl | dhcp-user-class | dhcp-vendor-class | dhcpv4-client-id-spl | dhcpv4-vendor-class | dhcpv6-client-id-ent-ident | dhcpv6-interface-id | dhcpv6-vendor-class-string | inner-vlan-id | outer-vlan-id | physical-adapter | physical-chassis | physical-port | physical-slot | physical-subslot | port-type | pppoe-session-id | remote-id-tag | service-name | username } | value** *value* **}**: Specifies the username format, identifier, or value.

  - **format** *attribute_format*: Specifies the username attribute format.
  - **identifier { addr | circuit-id-tag | client-mac-address | client-mac-address-custom1 | client-mac-address-custom2 | client-mac-address-ietf | client-mac-address-raw | dhcp-client-id | dhcp-client-id-spl | dhcp-user-class | dhcp-vendor-class | dhcpv4-client-id-spl | dhcpv4-vendor-class | dhcpv6-client-id-ent-ident | dhcpv6-interface-id | dhcpv6-vendor-class-string | inner-vlan-id | outer-vlan-id | physical-adapter | physical-chassis | physical-port | physical-slot | physical-subslot | port-type | pppoe-session-id | remote-id-tag | service-name | username }**: Specifies the username identifiers as follows:
    - **addr**: Specifies the IPv4 address of the subscriber.
    - **circuit-id-tag**: Specifies the circuit identifier tag.
    - **client-mac-address**: Specifies the client MAC address in AABB.CCDD.EEFF format.
    - **client-mac-address-custom1**: Specifies the first custom client MAC address in AABB.CCDD.EEFF format.
    - **client-mac-address-custom2**: Specifies the second custom client MAC address in AABB.CCDD.EEFF format.
    - **client-mac-address-ietf**: Specifies the client MAC address in Internet Engineering Task Force (IETF) format. That is, AA-BB-CC-DD-EE-FF format.
    - **client-mac-address-raw**: Specifies the client MAC address in raw (AABBCCDDEEFF) format.
    - **dhcp-client-id**: Specifies the DHCP client identifier.
    - **dhcp-client-id-spl**: Specifies the DHCP client identifier special string.
    - **dhcp-user-class**: Specifies the DHCP user class.
    - **dhcp-vendor-class**: Specifies the DHCP vendor class.
    - **dhcpv4-client-id-spl**: Specifies the DHCPv4 client identifier special string.
    - **dhcpv4-vendor-class**: Specifies the DHCPv4 vendor class.

- **dhcpv6-client-id-ent-ident**: Specifies the DHCPv6 client and enterprise identifiers.
- **dhcpv6-interface-id**: Specifies the DHCPv6 interface identifier.
- **dhcpv6-vendor-class-string**: Specifies the DHCPv6 vendor class string.
- **inner-vlan-id**: Specifies the inner VLAN identifier.
- **outer-vlan-id**: Specifies the outer VLAN identifier.
- **physical-adapter**: Specifies the physical adapter.
- **physical-chassis**: Specifies the physical chassis.
- **physical-port**: Specifies the physical port.
- **physical-slot**: Specifies the physical slot.
- **physical-subslot**: Specifies the physical subslot.
- **port-type**: Specifies the interface or port type.
- **pppoe-session-id**: Specifies the PPPoE physical identifier.
- **remote-id-tag**: Specifies the remote identifier tag.
- **service-name**: Specifies the service name.
- **username**: Specifies the username.

Accounting

```
config
  profile aaa aaa_name
    accounting
      method-order custom_server_group
      commit
```

NOTES:

- **profile aaa** *aaa_name*: Specifies the AAA profile name and enters the AAA Configuration mode.
- **accounting**: Enters the Accounting sub-mode.
- **method-order** *custom_server_group*: Specifies the method-order to be applied by default for subscriber accounting.

  custom_server_group specifies the name of the server group where the method-order is applied.

Configuring RADIUS Accounting Options

This section describes how to configure the RADIUS accounting options.

```
config
  profile radius accounting
    algorithm { first-server | round-robin }
    attribute
```

```
      instance instance_id
      { nas-identifier value  | nas-ip ipv4_address  | nas-ipv6    ipv6_address
|
      nas-port { nas_port } | { format-e format_e
      { nas-port-type nas_port_type } }
      deadtime value
      detect-dead-server response-timeout value
      max-retry value
      timeout value
      commit
```

NOTES:

- **profile radius accounting**: Enters the RADIUS accounting configuration mode.
- **algorithm { first-server | round-robin }**: Defines the algorithm for selecting the RADIUS server.
  - **first-server**: Sets the selection logic as highest priority first. This is the default behavior.
  - **round-robin**: Sets the selection logic as round-robin order of servers.
- **attribute**: Configures the RADIUS identification parameters.
- **instance** *instance_id*: Specifies the instance ID.
  - **nas-identifier** *value*: Specifies the attribute name by which the system will be identified in Accounting-Request messages. *value* must be an alphanumeric string.
  - **nas-ip** *ipv4_address*: Specifies the NAS IPv4 address. *ipv4_address* must be an IPv4 address in dotted decimal notation.
  - **nas-ipv6** *ipv6_address*: Specifies the NAS IPv6 address.
  - **nas-port {** *nas_port* **} | { format-e** *format_e* **{ nas-port-type** *nas_port_type* **} }**: Specifies the nas-port attributes.
    - *nas_port* configures the NAS port value. The NAS port value ranges from 1 to 4294967295.

Note If none of the NAS port configurations are present, the existing default nas-port logic is applied. That is, setting a fixed-number per radius-pod.

  - **format-e** *format_e_value* : Specifies the custom attribute formation support for nas-port. The nas-port is a 32 bit integer format. The configuration takes a 32 length of characters, each presenting a particular attribute mapping. The *format_e_value* pattern is: 01FSAPRiLUVQ]*):
  - 0 – Set bit to 0

    1 – Set bit to 1

    F - PHY_SHELF

    S – PHY_SLOT

    A – PHY_ADAPTER

    P - PHY_PORT

R - PHY_CHASSIS

i - PHY_SUBSLOT

L - PHY_CHANNEL

V - OUTER_VLAN_ID

Q - INNER_VLAN_ID

U - PPPOE_SESSION_ID

nas-port-type *nas_port_type*: Specifies the NAS port type. The supported values range from 0 to 44.



Note

- The nas-port-type configuration is not in scope of the Control Plane. It is derived from the interface-type.
- The supported NAS port types are 36 , 37, 43, and 44.
- The NAS port type value takes precedence over the common NAS port format-e.

- **deadtime** *value*: Sets the time to elapse between RADIUS server marked unreachable and when we can re-attempt to connect.

  value must be an integer from 0 through 65535. Default: 10 minutes.

- **detect-dead-server response-timeout** *value*: Sets the timeout value that marks a server as "dead" when a packet is not received for the specified number of seconds.

  value must be an integer from 1 through 65535. Default: 10 seconds.

- **max-retry** *value*: Sets the maximum number of times that the system will attempt retry with the RADIUS server.

  value must be an integer from 0 through 65535. Default: 2

- **timeout** *value*: Sets the time to wait for response from the RADIUS server before retransmitting.

  value must be an integer from 1 through 65535. Default: 2 seconds.

- **commit**: Commits the configuration.
- All the keyword options under the RADIUS accounting configuration mode are also available within the RADIUS configuration mode.

Configuring RADIUS Accounting Server Group

This section describes how to configure the RADIUS server group.

```
configure
  profile radius
     server-group group_name
     commit
```

NOTES:

- **profile radius**: Enters the RADIUS configuration mode.
- **server group** *group_name*: Specifies the name of server group for use in RADIUS accounting. *group_name* must be an alphanumeric string.
- **commit**: Commits the configuration.

Configuring RADIUS Attributes

This section describes how to configure the RADIUS attributes for authentication and accounting.

```
config
  profile radius
        attribute
          instance instance_id
          { encode-user-plane-ip | nas-identifier value | nas-ip ipv4_address
 | nas-ipv6 ipv6_address }
           commit
```

NOTES:

- **profile radius**: Enters the RADIUS configuration mode.
- **attribute**: Configures the RADIUS identification parameters.
- **instance** *instance_id*: Specifies the instance ID.
  - **nas-identifier** *value*: Specifies the attribute name by which the system will be identified in Accounting-Request messages. *value* must be an alphanumeric string.
  - **nas-ip** *ipv4_address*: Specifies the NAS IPv4 address. *ipv4_address* must be an IPv4 address in dotted decimal notation.
  - **nas-ipv6** *ipv6_address*: Specifies the NAS IPv6 address.
- **encode-user-plane-ip**: Enables encoding User Plane IP address in RADIUS packets.
- **commit**: Commits the configuration.

Sample Configuration

The following is a sample configuration.

```
config
   profile radius
      attribute
        instance 1
          nas-identifier Ciscobng
          nas-ip 10.1.32.83
          nas-ipv6 2001::dB8:0
        exit
     exit
```

Configuring RADIUS Attribute Format

Use the following commands to configure the RADIUS identification parameters.

```
configure
profile radius attribute
  instance instance_id
     called-station-id { format-name format_name
               | nas-port-type nas_port_type }
     calling-station-id { format-name format_name
               | nas-port-type nas_port_type }
     nas-identifier nas_identifier
     nas-identifier-format nas_identifier_format
     nas-ip { ipv4_address |
     nas-ipv6 ipv6_address |
     nas-port { nas_port } | { format-e format_e
               { nas-port-type nas_port_type } }
     nas-port-id nas_port_id { format-name format_name|
               | nas-port-type nas_port_type }
     commit
```

NOTES:

- **profile radius attribute**: Enters the Profile RADIUS Attribute Configuration mode.
- **instance** *instance_id*: Specifies the instance ID.
- **called-station-id { format-name** *format_name* **| nas-port-type** *nas_port_type* **}** : Specifies the AAA called-station-id attribute.

  format-name *format_name*: Specifies the called-station-id format name.

  nas-port-type *nas_port_type*: Specifies the NAS port type. The supported values range from 0 to 44.

  - nas-port-type configuration is not in scope of the Control Plane. It is derived dependng on the interface-type.
  - The supported NAS port types are 36 , 37, 43, and 44.

- **calling-station-id { format-name** *format_name* **| nas-port-type** *nas_port_type* **}** : Specifies the AAA calling-station-id attribute.
- **nas-identifier { format-name** *format_name* **| nas-port-type** *nas_port_type* **}**: Specifies the attribute name with which the system is identified in the Access-Request messages. The identifier string ranges from 1 to 32 characters.
- **nas-identifier-format { format-name** *format_name* **| nas-port-type** *nas_port_type* **}**: Specifies the AAA nas-identifier-format attribute.
- **nas-ip** *ipv4_address*: Specifies the AAA NAS IPv4 address.
- **nas-ipv6** *ipv6_address*: Specifies the NAS IPv6 address.
- **nas-port {** *nas_port* **} | { format-e** *format_e* **{ nas-port-type** *nas_port_type* **} }**: Specifies the nas-port attributes.
  - *nas_port* configures the NAS port value. The NAS port value ranges from 1 to 4294967295.

Note If none of the NAS port configurations are present, the existing default nas-port logic is applied. That is, setting a fixed-number per radius-pod.

- **format-e** *format_e_value* : Specifies the custom attribute formation support for nas-port. The nas-port is a 32 bit integer format. The configuration takes a 32 length of characters, each presenting a particular attribute mapping. The *format_e_value* pattern is: 01FSAPRiLUVQ]*):
- 0 – Set bit to 0

  1 – Set bit to 1

  F - PHY_SHELF

  S – PHY_SLOT

  A – PHY_ADAPTER

  P - PHY_PORT

  R - PHY_CHASSIS

  i - PHY_SUBSLOT

  L - PHY_CHANNEL

  V - OUTER_VLAN_ID

  Q - INNER_VLAN_ID

  U - PPPOE_SESSION_ID

  nas-port-type *nas_port_type*: Specifies the NAS port type. The supported values range from 0 to 44.

Note
- The nas-port-type configuration is not in scope of the Control Plane. It is derived from the interface-type.
- The supported NAS port types are 36 , 37, 43, and 44.
- The NAS port type value takes precedence over the common NAS port format-e.

- **nas-port-id** *nas_port_id*: Specifies the AAA NAS port-id attribute.

Configuring RADIUS Dead Time

This section describes how to configure the RADIUS dead time.

```
config
  profile radius
    deadtime value
    commit
```

NOTES:

- **profile radius**: Enters the RADIUS configuration mode.
- **deadtime** *value*: Sets the time to elapse between RADIUS server marked unreachable and when an reattempt to connect can be made.

  value must be an integer from 0 through 65535. Default: 10 minutes.
- **commit**: Commits the configuration.

Sample Configuration

The following is a sample configuration.

```
config
   profile radius
     deadtime 15
     exit
```

Configuring RADIUS Detect Dead Server

This section describes how to configure the RADIUS detect dead server.

```
config
   profile radius
      detect-dead-server response-timeout value
      commit
```

NOTES:

- **profile radius**: Enters the RADIUS configuration mode.
- **detect-dead-server response-timeout** *value*: Sets the timeout value that marks a server as "dead" when a packet is not received for the specified number of seconds.

  value must be an integer from 1 through 65535. Default: 10 seconds.
- **commit**: Commits the configuration.

Sample Configuration

The following is a sample configuration.

```
config
   profile radius
      detect-dead-server response-timeout 100
      exit
```

Configuring RADIUS NAS-IP

This section describes how to configure the RADIUS NAS-IP.

Global RADIUS NAS-IP Configuration

Use the following configuration to configure the NAS-IP address.

```
config
   endpoint radius-dns
      interface radius-client
         vip-ip ipv4_address
         commit
```

NOTES:

- **endpoint radius-dns**: Enters the endpoint radius-ep configuration mode.
- **interface radius-client**: Enters the radius-client interface-type configuration mode.
- **vip-ip** *ipv4_address*: Sets the NAS-IP value, which is also used as the source-IP in UDP requests towards the RADIUS server.
- **commit**: Commits the configuration.

Configuration Example:

```
config
   endpoint radius-dns
      interface radius-client
      vip-ip 209.165.200.228
      exit
   exit
exit
```

Configuring RADIUS Pod

This section describes how to configure the RADIUS pod.

```
config
   endpoint radius
      replicas number_of_replicas
      commit
```

NOTES:

- **endpoint radius**: Enters the RADIUS endpoint configuration mode.
- **replicas** *number_of_replicas*: Sets the number of replicas required.
- **commit**: Commits the configuration.

Sample Configuration

The following is a sample configuration.

```
config
   endpoint radius
      replicas 3
      exit
```

Configuring RADIUS Retries

This section describes how to configure the maximum RADIUS retries.

```
config
  profile radius
    max-retry value
    commit
```

NOTES:

- **profile radius**: Enters the RADIUS configuration mode.
- **max-retry** *value*: Sets the maximum number of times that the system will attempt retry with the RADIUS server.

  value must be an integer from 0 through 65535. Default: 2
- **commit**: Commits the configuration.

Sample Configuration

The following is a sample configuration.

```
config
   profile radius
      max-retry 2
      exit
```

Configuring RADIUS Server

This section describes how to configure the RADIUS server settings.

```
config
  profile radius
    server ipv4/ipv6_address port_number
      secret secret_key
      priority priority_value
      type { acct | auth }
      commit
```

NOTES:

- **profile radius**: Enters the RADIUS configuration mode.
- **server** *ipv4/ipv6_address port_number*: Specifies the IPv4/IPv6 address and port of the RADIUS server.
- **secret** *secret_key*: Specifies the secret key.
- **priority** *priority_value*: Specifies the server priority.
- **type { acct | auth }**: Specifies the type of the RADIUS server. It can be one of the following:
  - acct: RADIUS server used for the accounting requests
  - auth: RADIUS server used for the authentication requests
- **commit**: Commits the configuration.

Configuring RADIUS Server Group

Use the following commands to configure the RADIUS server group.

```
config
  profile server-group server_group_name
    radius-group radius_server_group_name
    commit
```

NOTES:

- **profile server-group** *server_group_name*: Specifies the profile server group name to enter the Profile Server Group Configuration mode.
- **radius-group** *radius_server_group_name*: Specifies the RADIUS group server name.

Sample Configuration

The following is a sample configuration:

```
server-group automation-server-group
server auth 10.1.36.121 1812
exit
server acct 10.1.36.121 1813
exit
server auth 2001::10:1:36:121 1812
exit
server acct 2001::10:1:36:121 1813
exit
```

Configuring RADIUS Server Selection Logic

This section describes how to configure the RADIUS server selection logic.

```
config
  profile radius
    algorithm { first-server | round-robin | least-outstanding [
batch-size number ] }
    commit
```

NOTES:

- **profile radius**: Enters the RADIUS configuration mode.
- **algorithm { first-server | round-robin | least-outstanding [batch-size] }**: Defines the algorithm for selecting the RADIUS server.
  - **first-server**: Sets the selection logic as highest priority first. This is the default behavior.
  - **round-robin**: Sets the selection logic as round-robin order of servers.
  - **least-outstanding** : Sets the selection logic based on the server with the lowest number of outstanding transactions in its queue.
    - **batch-size** *number*: (Optional) Specifies the size of the batch.

      If you do not configure the **batch-size** value in the CLI command, the system takes the default batch size.

- **commit**: Commits the configuration.

Sample Configuration

The following is a sample configuration.

```
config
   profile radius
      algorithm least-outstanding batch-size 30
      exit
```

Configuring RADIUS Timeout

This section describes how to configure the RADIUS timeout.

```
config
   profile radius
      timeout value
      commit
```

NOTES:

- **profile radius**: Enters the RADIUS configuration mode.
- **timeout** *value_in_seconds*: Sets the time to wait for response from the RADIUS server before retransmitting.

  value must be an integer from 1 through 65535. Default: 2 seconds.
- **commit**: Commits the configuration.

Sample Configuration

The following is a sample configuration.

```
config
   profile radius
      timeout 4
      exit
```

Configuring Radius PID-Quarantine

The RADIUS PID-Quarantine feature temporarily reserves a Port Identifier (PID), comprising a UDP source port and an 8-bit RADIUS Identifier, after a request completes or times out. By preventing the immediate reuse of PIDs, the system avoids incorrect transaction correlation caused by late-arriving responses.

Table 11: Feature History Table

Feature Name	Release Information	Feature Description
Radius PID-Quarantine	2026.04.0	Enables temporary reservation of Port Identifiers (PIDs) for RADIUS requests after completion or timeout. This prevents late-arriving responses from being incorrectly correlated with new requests, enhancing overall RADIUS signaling reliability.

Sample Configuration

To configure the PID quarantine duration:

```
config
 profile radius
  pid-quarantine <5..60>
 commit
```

Parameters:

pid-quarantine: Specifies the PID quarantine value in seconds. The default value is 10 seconds. When a request times out, the PID is moved to a quarantine queue for the configured duration before it can be reused.