RADIUS Server Load Balancing

Feature history for RADIUS server load balancing

This table provides release and platform support information for the features explained in this module.

These features are available in all the releases subsequent to the one they were introduced in, unless noted otherwise.

Release	Feature name and description	Supported platform
Cisco IOS XE 17.18.1	RADIUS server load balancing: This feature distributes AAA authentication and accounting transactions across RADIUS servers in a server group.	Cisco C9350 Series Smart Switches Cisco C9610 Series Smart Switches

Understand RADIUS server load balancing

The RADIUS Server Load Balancing feature distributes AAA authentication and accounting transactions across RADIUS servers in a server group. These servers can share the AAA load and thereby respond faster to incoming requests.

RADIUS server load balancing process

Load balancing distributes batches of transactions to RADIUS servers within a server group. Each batch of transactions is assigned by load balancing to the server with the lowest number of outstanding transactions in its queue. The process to assign a batch of transactions involves these steps:

1. Receive the first transaction for a new batch.
2. Check all server transaction queues.
3. Identify the server with the lowest number of outstanding transactions.
4. Assign the next batch of transactions to the identified server.

The batch size, configured by the user, impacts both CPU load and network throughput. A larger batch size decreases CPU load and increases throughput. Very large batch sizes might not fully utilize all server resources. As batch size decreases, CPU load increases and network throughput decreases.

Note

- There is no set number for large or small batch sizes. A batch with more than 50 transactions is considered large and a batch with fewer than 25 transactions is considered small.
- If a server group contains 10 or more servers, use a high batch size to reduce CPU load.

Load balancing across RADIUS server groups

A RADIUS server group is a collection of RADIUS servers configured to work together to provide redundancy and load balancing for AAA services. These groups allow a device to use multiple RADIUS servers as a single logical entity, improving reliability and scalability.

You can configure load balancing for either a named RADIUS server group or the global RADIUS server group. The load balancing server group must be referred to as “radius” in the AAA method lists. All public servers that are part of the RADIUS server group are then load balanced.

You can configure authentication and accounting to use the same RADIUS server or different servers. In some cases, the same server can be used for preauthentication, authentication, or accounting transactions for a session. The preferred server, which is an internal setting and is set as the default, informs AAA to use the same server for the start and stop record for a session regardless of the server cost. When using the preferred server setting, ensure the server for the initial transaction (for example, authentication) is included in any other server group used for subsequent transactions (for example, accounting).

The preferred server is not used if one of the following criteria is true:

- The **load-balance method least-outstanding ignore-preferred-server** command is used.
- The preferred server is dead.
- The preferred server is in quarantine.
- The want server flag has been set, overriding the preferred server setting.

Use the want server flag, an internal setting, when you need the same server for all stages of a multistage transaction, regardless of server cost. If the want server is not available, the transaction fails.

Use the **load-balance method least-outstanding ignore-preferred-server** command if you have either of the following configurations:

- Dedicated authentication server and a separate dedicated accounting server
- Network where you can track all call record statistics and call record details, including start and stop records and records that are stored on separate servers

If you have a configuration where authentication servers are a superset of accounting servers, the preferred server is not used.

RADIUS automated testing and server status

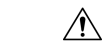
The RADIUS automated tester is a feature that periodically sends test authentication requests to configured RADIUS servers to verify their availability and responsiveness. If the server returns an Access-Reject message, it indicates the server is alive. If it does not, the server is considered either dead or quarantined.

The RADIUS Server Load Balancing feature considers the server status and assigns transaction batches only to live servers. We recommend that you test the status of all RADIUS load-balanced servers, including low usage servers (for example, backup servers).

The system does not send transactions to a server marked as dead. A server is marked dead until its timer expires, moving it to quarantine state. The RADIUS automated tester verifies a server as alive before ending its quarantine.

If a transaction is sent to an unresponsive server, it fails over to the next available server before marking the unresponsive one as dead. Using the retry reorder mode can help manage failed transactions efficiently.

Verify that the AAA servers respond to test packets sent by the NAS when using the RADIUS automated tester. If servers are incorrectly configured, packets may be dropped, and servers erroneously marked dead.



Caution We recommend using a test user who is not defined on the RADIUS server for automated testing to protect against security issues that may arise if the test user is not configured correctly.

After configuring the **automate-tester username** *name* **probe-on** parameter, a five-second dead timer starts, followed by sending a packet to the external RADIUS server. If there is a response, the server state updates. If there is no response, the packets are sent out according to the timeout interval that is configured using the **radius-server timeout** command. This will continue for 180 seconds, and if there is still no response, a new dead timer is started based on the configured **radius-server deadtime** command.

VRF-aware RADIUS automated testing

VRF-aware RADIUS automated testing allows the device to send periodic RADIUS test authentication requests within a specified VRF context to verify the availability of RADIUS servers associated with that VRF.

The RADIUS-automated tester operates at the server level and does not involve any groups, and the operation level for VRF is at the group level. All the information related to the VRF and the source-interface configurations is maintained in a group structure. The automated tester can access configuration information in the global source-interface when available. If missing in either the global source-interface or the default VRF, the automated tester marks the server as dead.

Use the **vrf** keyword with the **automate-tester** command to enable it for a non-default VRF.

Note To activate the VRF-aware automate-tester, configure the **global config ipv4/ipv6 source interface** *interface-name* **vrf** *vrf-name* command.

Prerequisites for RADIUS server load balancing

- AAA must be configured on the RADIUS server.
- AAA RADIUS server groups must be configured.
- RADIUS must be configured for functions such as authentication, accounting, or static route download.

Restrictions for RADIUS server load balancing

- Incoming RADIUS requests such as Packet of Disconnect (POD) requests are not supported.
- Load balancing is not supported on proxy RADIUS servers and for private server groups.
- 802.1x and MAC authentication bypass (MAB) authentication using RADIUS server load balancing is not supported on Cisco C9610 Series Smart Switches.

Configure RADIUS server load balancing

Configure load balancing for either per-named RADIUS server group or for the global RADIUS server group, followed by enabling the RADIUS automated tester function.

Enable load balancing for a named RADIUS server group

Perform this procedure to enable load balancing for a named RADIUS server group.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**aaa group server radius** *group-name* **Example:** `Device(config)# aaa group server radius rad-sg`	Enters server group configuration mode.

	Command or Action	Purpose
Step 4	**server ip-address** [**auth-port** *port-number*] [**acct-port** *port-number*] **Example:** `Device(config-sg-radius)# server 192.0.2.238 auth-port 2095 acct-port 2096`	Configures the IP address of the RADIUS server for the group server.
Step 5	**load-balance method least-outstanding** [**batch-size** *number*] [**ignore-preferred-server**] **Example:** `Device(config-sg-radius)# load-balance method least-outstanding batch-size 30`	Enables the least-outstanding load balancing for a named server group.
Step 6	**end** **Example:** `Device(config-sg-radius)# end`	Exits server group configuration mode and returns to privileged EXEC mode.

Enable load balancing for the global RADIUS server group

Perform this procedure to enable load balancing for the global RADIUS server group.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**radius-server load-balance method least-outstanding** [**batch-size** *number*] [**ignore-preferred-server**] **Example:** `Device(config-sg-radius)# radius-server load-balance method least-outstanding`	Enables the least-outstanding load balancing for the global RADIUS server group.
Step 4	**end** **Example:** `Device(config-sg-radius)# end`	Exits server group configuration mode and returns to privileged EXEC mode.

Enable RADIUS automated testing

Perform this procedure to enable RADIUS automated testing.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**radius server** *name* **Example:** `Device(config)# radius server myserver`	Specifies the name of the RADIUS server configuration and enters RADIUS server configuration mode.
Step 4	**address** {**ipv4** | **ipv6**} {*ip-address* | *host-name*} **auth-port** *port-numer* **acct-port** *port-number* **Example:** `Device(config-radius-server)# address ipv4 192.0.2.1 auth-port 1812 acct-port 1813`	Configures the IPv4 address for the RADIUS server accounting and authentication parameters.
Step 5	**automate-tester username** *user* {**idle-time** | **ignore-acct-port** | **ignore-auth-port** | **probe-on**} **Example:** `Device(config-radius-server)# automate-tester username user1 probe-on`	Enables RADIUS automated testing. You can configure the following parameters: • **idle-time**: Specifies the idle time after which server state should be verified. • **ignore-acct-port**: Specifies that the testing should not be performed on the accounting ports of the servers. • **ignore-auth-port**: Specifies that the testing should not be performed on the authentication port of the servers. • **probe-on**: Sends a packet to verify the server status.
Step 6	**end** **Example:** `Device(config-radius-server)# end`	Exits RADIUS server configuration mode and returns to privileged EXEC mode.

Enable VRF-aware RADIUS automated testing

Perform this procedure to enable RADIUS automated testing for a non-default VRF.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device>enable`	Enables privileged EXEC mode. Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**radius server** *name* **Example:** `Device(config)# radius server myserver`	Specifies the name of the RADIUS server configuration and enters RADIUS server configuration mode.
Step 4	**address** {**ipv4** | **ipv6**} {*ip-address* | *host-name*} **auth-port** *port-numer* **acct-port** *port-number* **Example:** `Device(config-radius-server)# address ipv4 192.0.2.1 auth-port 1812 acct-port 1813`	Configures the IPv4 address for the RADIUS server accounting and authentication parameters.

Step 5	**automate-tester username** *user* [**ignore-auth-port**] [**ignore-acct-port**] [**idle-time** *minutes*] **vrf** *vrf-name* or **automate-tester username** *user* **probe-on vrf** *vrf-name*
	Example:
	`Device(config-radius-server)# automate-tester username user1 idle-time 2 vrf VRF1`
	or
	`Device(config-radius-server)# automate-tester username user1 probe-on vrf VRF1`
		Enables RADIUS automated testing for a non-default VRF.

Step 6	**end** **Example:** `Device(config-radius-server)# end`	Exits RADIUS server configuration mode and returns to privileged EXEC mode.

Troubleshoot RADIUS server load balancing

After configuring the RADIUS Server Load Balancing feature, monitor the idle timer, dead timer, and load balancing server selection, and verify the server status by using a test command.

Procedure

Step 1 Use the **debug aaa test** command to determine when an idle timer or dead timer expires, when test packets are sent, and verify the server status or state.

The idle timer checks the server status and updates with or without any incoming requests. Monitor the idle timer to identify nonresponsive servers and keep the RADIUS server status updated to efficiently use available resources. For instance, an updated idle timer ensures that incoming requests are sent to active servers.

The dead timer determines whether a server is dead or updates the status of a dead server appropriately.

Monitoring helps determine how often the server selection changes. Server selection analyzes bottlenecks, queued requests, or servers processing specific requests.

A sample output from the **debug aaa test** command shows when the idle timer expired:

Example:

```
Device# debug aaa test

Jul 16 00:07:01: AAA/SG/TEST: Server (192.0.2.245:1700,1701) quarantined.
Jul 16 00:07:01: AAA/SG/TEST: Sending test request(s) to server (192.0.2.245:1700,1701)
Jul 16 00:07:01: AAA/SG/TEST: Sending 1 Access-Requests, 1 Accounting-Requests in current
batch.
Jul 16 00:07:01: AAA/SG/TEST(Req#: 1): Sending test AAA Access-Request.
Jul 16 00:07:01: AAA/SG/TEST(Req#: 1): Sending test AAA Accounting-Request.
Jul 16 00:07:01: AAA/SG/TEST: Obtained Test response from server (192.0.2.245:1700,1701)
Jul 16 00:07:01: AAA/SG/TEST: Obtained Test response from server (192.0.2.245:1700,1701)
Jul 16 00:07:01: AAA/SG/TEST: Necessary responses received from server (192.0.2.245:1700,1701)
Jul 16 00:07:01: AAA/SG/TEST: Server (192.0.2.245:1700,1701) marked ALIVE. Idle timer set
for 60 sec(s).
Jul 16 00:07:01: AAA/SG/TEST: Server (192.0.2.245:1700,1701) removed from quarantine.
```

Step 2 Use the **debug aaa sg-server selection** command to determine the server that is selected for load balancing.

A sample output from the **debug aaa sg-server selection** command shows five access requests being sent to a server group with a batch size of three:

Example:

```
Device# debug aaa sg-server selection

Jul 16 03:15:05: AAA/SG/SERVER_SELECT: Obtaining least loaded server.
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: [3] transactions remaining in batch. Reusing server.
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: Obtaining least loaded server.
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: [2] transactions remaining in batch. Reusing server.
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: Obtaining least loaded server.
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: [1] transactions remaining in batch. Reusing server.
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: Obtaining least loaded server.
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: No more transactions in batch. Obtaining a new server.
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: Obtaining a new least loaded server.
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: Server[0] load: 3
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: Server[1] load: 0
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: Server[2] load: 0
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: Selected Server[1] with load 0
```

```
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: [3] transactions remaining in batch.
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: Obtaining least loaded server.
Jul 16 03:15:05: AAA/SG/SERVER_SELECT: [2] transactions remaining in batch. Reusing server.
```

Step 3 Use the **test aaa group** command to manually verify the RADIUS load-balanced server status.

A sample output shows the response from a load-balanced RADIUS server that is alive when the username "test" does not match a user profile. The server is verified alive when it issues an Access-Reject response to a AAA packet generated using the **test aaa group** command.

Example:

```
Device# test aaa group SG1 test lab new-code

00:06:07: RADIUS/ENCODE(00000000):Orig. component type = INVALID
00:06:07: RADIUS/ENCODE(00000000): dropping service type, "radius-server attribute 6
on-for-login-auth" is off
00:06:07: RADIUS(00000000): Config NAS IP: 192.0.2.4
00:06:07: RADIUS(00000000): sending
00:06:07: RADIUS/ENCODE: Best Local IP-Address 192.0.2.141 for Radius-Server 192.0.2.176
00:06:07: RADIUS(00000000): Send Access-Request to 192.0.2.176:1645 id 1645/1, len 50
00:06:07: RADIUS:  authenticator CA DB F4 9B 7B 66 C8 A9 - D1 99 4E 8E A4 46 99 B4
00:06:07: RADIUS:  User-Password       [2]   18  *
00:06:07: RADIUS:  User-Name           [1]   6   "test"
00:06:07: RADIUS:  NAS-IP-Address      [4]   6   192.0.2.141
00:06:07: RADIUS: Received from id 1645/1 192.0.2.176:1645, Access-Reject, len 44
00:06:07: RADIUS:  authenticator 2F 69 84 3E F0 4E F1 62 - AB B8 75 5B 38 82 49 C3
00:06:07: RADIUS:  Reply-Message       [18]  24
00:06:07: RADIUS:   41 75 74 68 65 6E 74 69 63 61 74 69 6F 6E 20 66  [Authentication f]
00:06:07: RADIUS:   61 69 6C 75 72 65                               [failure]
00:06:07: RADIUS(00000000): Received from id 1645/1
00:06:07: RADIUS/DECODE: Reply-Message fragments, 22, total 22 bytes
```

Configuration examples

Refer this section for configuration examples of RADIUS server load balancing.

Example: Enable load balancing for a named RADIUS server group

The examples are divided into three parts. These include the current configuration of the RADIUS command output, debug output, and AAA server status information.

This sample output shows the relevant RADIUS configuration:

```
Device# show running-config
.
.
.
aaa group server radius server-group1
 server 192.0.2.238 auth-port 2095 acct-port 2096
 server 192.0.2.238 auth-port 2015 acct-port 2016
 load-balance method least-outstanding batch-size 5
!
aaa authentication ppp default group server-group1
aaa accounting network default start-stop group server-group1
.
.
```

```
.
Device(config-sg-radius)# load-balance method least-outstanding batch-size 30
```

The lines in the current configuration of the preceding RADIUS command output are defined here:

- The **aaa group server radius** command shows the configuration of a server group with two member servers.
- The **load-balance** command enables load balancing for global RADIUS server groups with the batch size specified.
- The **aaa authentication ppp** command authenticates all PPP users using RADIUS.
- The **aaa accounting** command enables all accounting requests to be sent to the AAA server when the client is authenticated and then disconnected using the **start-stop** keyword.

This sample output shows the selection of the preferred server and the processing of requests for the preceding configuration:

```
Device# show debug

*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT(0000002C):No preferred server available.
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT:Obtaining least loaded server.
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT:No more transactions in batch. Obtaining a new
server.
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT:Obtaining a new least loaded server.
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT:Server[0] load:0
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT:Server[1] load:0
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT:Selected Server[0] with load 0
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT:[5] transactions remaining in batch.
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT(0000002C):Server (192.0.2.238:2095,2096) now being
 used as preferred server
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT(0000002D):No preferred server available.
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT:Obtaining least loaded server.
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT:[4] transactions remaining in batch. Reusing
server.
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT(0000002D):Server (192.0.2.238:2095,2096) now being
 used as preferred server
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT(0000002E):No preferred server available.
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT:Obtaining least loaded server.
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT:[3] transactions remaining in batch. Reusing
server.
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT(0000002E):Server (192.0.2.238:2095,2096) now being
 used as preferred server
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT(0000002F):No preferred server available.
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT:Obtaining least loaded server.
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT:[2] transactions remaining in batch. Reusing
server.
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT(0000002F):Server (192.0.2.238:2095,2096) now being
 used as preferred server
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT(00000030):No preferred server available.
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT:Obtaining least loaded server.
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT:[1] transactions remaining in batch. Reusing
server.
*Feb 28 13:51:16.019:AAA/SG/SERVER_SELECT(00000030):Server (192.0.2.238:2095,2096) now being
 used as preferred server
*Feb 28 13:51:16.023:AAA/SG/SERVER_SELECT(00000031):No preferred server available.
*Feb 28 13:51:16.023:AAA/SG/SERVER_SELECT:Obtaining least loaded server.
*Feb 28 13:51:16.023:AAA/SG/SERVER_SELECT:No more transactions in batch. Obtaining a new
server.
*Feb 28 13:51:16.023:AAA/SG/SERVER_SELECT:Obtaining a new least loaded server.
*Feb 28 13:51:16.023:AAA/SG/SERVER_SELECT:Server[1] load:0
*Feb 28 13:51:16.023:AAA/SG/SERVER_SELECT:Server[0] load:5
```

```
*Feb 28 13:51:16.023:AAA/SG/SERVER_SELECT:Selected Server[1] with load 0
*Feb 28 13:51:16.023:AAA/SG/SERVER_SELECT:[5] transactions remaining in batch.
*Feb 28 13:51:16.023:AAA/SG/SERVER_SELECT(00000031):Server (192.0.2.238:2015,2016) now being
 used as preferred server
*Feb 28 13:51:16.023:AAA/SG/SERVER_SELECT(00000032):No preferred server available.
*Feb 28 13:51:16.023:AAA/SG/SERVER_SELECT:Obtaining least loaded server.
*Feb 28 13:51:16.023:AAA/SG/SERVER_SELECT:[4] transactions remaining in batch. Reusing
server.
.
.
.
```

This sample output from the **show aaa servers** command shows the AAA server status for the named RADIUS server group configuration:

```
Device# show aaa servers

RADIUS: id 3, priority 1, host 9:76:239::219, auth-port 1812, acct-port 1813, hostname r6
State: current UP, duration 223000s, previous duration 301s
Dead: total time 682s, count 2
Platform State from SMD: current UP, duration 222972s, previous duration 258s
SMD Platform Dead: total time 702s, count 3
Platform State from WNCD (1) : current UP
Platform State from WNCD (2) : current UP
Platform State from WNCD (3) : current UP
Platform State from WNCD (4) : current UP
Platform State from WNCD (5) : current UP
Platform State from WNCD (6) : current UP
Platform State from WNCD (7) : current UP
Platform State from WNCD (8) : current UP, duration 2451264s, previous duration 258s
Platform Dead: total time 703s, count 3
Quarantined: No
Authen: request 68, timeouts 68, failover 0, retransmission 53


Sates defination:
State: current UP. ===> this is IOSD state
Platform State from SMD: current UP. ====> This is wired BINOS i,e SMD
Platform State from WNCD (1) : current UP ===> This is wireless BINOS i.e WNCD instance 1
Platform State from WNCD (2) : current UP. ===> This is wireless BINOS i.e WNCD instance 2
Platform State from WNCD (3) : current UP
Platform State from WNCD (4) : current UP
Platform State from WNCD (5) : current UP
Platform State from WNCD (6) : current UP
Platform State from WNCD (7) : current UP
Platform State from WNCD (8) : current UP. ===> This is wireless BINOS i.e WNCD instance 8
```

Example: Monitor RADIUS server state and automated tester function

These examples show idle timer and related server state for load balancing enabled for a named RADIUS server group and debug command output.

This sample output shows the relevant RADIUS configuration:

```
Device(config)# do show run aaa

aaa group server radius server-group1
radius server server1
address ipv4 192.0.2.1 auth-port 1812 acct-port 1813
automate-tester username user1 idle-time 2 vrf VRF1
radius-server load-balance method least-outstanding batch-size 5
```

The lines in the current configuration of the preceding RADIUS command output are defined here:

- The **aaa group server radius** command shows the configuration of a server group.
- The **radius server** and **address** command defines the RADIUS server name and IP address of the RADIUS server with authorization and accounting ports specified.
- The **radius-server load-balance** command enables load balancing for the RADIUS server with the batch size specified.

This sample output shows test requests being sent to servers. The response to the test request sent to the server is received, the server is appropriately removed from quarantine, marked alive, and the idle timer is reset.

```
Device# show debug

*Feb 28 13:52:20.835:AAA/SG/TEST:Server (192.0.2.238:2015,2016) quarantined.
*Feb 28 13:52:20.835:AAA/SG/TEST:Sending test request(s) to server (192.0.2.238:2015,2016)
*Feb 28 13:52:20.835:AAA/SG/TEST:Sending 1 Access-Requests, 1 Accounting-Requests in current
 batch.
*Feb 28 13:52:20.835:AAA/SG/TEST(Req#:1):Sending test AAA Access-Request.
*Feb 28 13:52:20.835:AAA/SG/TEST(Req#:1):Sending test AAA Accounting-Request.
*Feb 28 13:52:21.087:AAA/SG/TEST:Obtained Test response from server (192.0.2.238:2015,2016)
*Feb 28 13:52:22.651:AAA/SG/TEST:Obtained Test response from server (192.0.2.238:2015,2016)
*Feb 28 13:52:22.651:AAA/SG/TEST:Necessary responses received from server
(192.0.2.238:2015,2016)
*Feb 28 13:52:22.651:AAA/SG/TEST:Server (192.0.2.238:2015,2016) marked ALIVE. Idle timer
set for 60 secs(s).
*Feb 28 13:52:22.651:AAA/SG/TEST:Server (192.0.2.238:2015,2016) removed from quarantine.
.
.
```

Example: Configure the preferred server with the same authentication and authorization server

This example shows an authentication server group and an authorization server group that use the same servers 209.165.200.225 and 209.165.200.226. Both server groups have the preferred server flag enabled.

```
Device> enable
Device# configure terminal
Device(config)# aaa group server radius authentication-group
Device(config-sg-radius)# server 209.165.200.225 key radkey1
Device(config-sg-radius)# server 209.165.200.226 key radkey2
Device(config-sg-radius)# exit
Device(config)# aaa group server radius accounting-group
Device(config-sg-radius)# server 209.165.200.225 key radkey1
Device(config-sg-radius)# server 209.165.200.226 key radkey2
Device(config-sg-radius)# end
```

When a preferred server is selected for a session, all transactions for that session will continue to use the original preferred server. The load balancing for servers 209.165.200.225 and 209.165.200.226 is based on sessions instead of transactions.

Example: Configure the preferred server with different authentication and authorization servers

This example shows an authentication server group that uses servers 209.165.200.225 and 209.165.200.226 and an authorization server group that uses servers 209.165.201.1 and 209.165.201.2. Both server groups have the preferred server flag enabled.

```
Device> enable
Device# configure terminal
Device(config)# aaa group server radius authentication-group
Device(config-sg-radius)# server 209.165.200.225 key radkey1
Device(config-sg-radius)# server 209.165.200.226 key radkey2
Device(config-sg-radius)# exit
Device(config)# aaa group server radius accounting-group
Device(config-sg-radius)# server 209.165.201.1 key radkey3
Device(config-sg-radius)# server 209.165.201.2 key radkey4
Device(config-sg-radius)# end
```

The authentication server group and the accounting server group do not share any common servers. A preferred server is never found for accounting transactions; therefore, authentication and accounting servers are load-balanced based on transactions. Start and stop records are sent to the same server for a session.

Example: Configure the preferred server with overlapping authentication and authorization servers

This example shows an authentication server group that uses servers 209.165.200.225, 209.165.200.226, and 209.165.201.1 and an accounting server group that uses servers 209.165.201.1 and 209.165.201.2. Both server groups have the preferred server flag enabled.

```
Device> enable
Device# configure terminal
Device(config)# aaa group server radius authentication-group
Device(config-sg-radius)# server 209.165.200.225 key radkey1
Device(config-sg-radius)# server 209.165.200.226 key radkey2
Device(config-sg-radius)# server 209.165.201.1 key radkey3
Device(config-sg-radius)# exit
Device(config)# aaa group server radius accounting-group
Device(config-sg-radius)# server 209.165.201.1 key radkey3
Device(config-sg-radius)# server 209.165.201.2 key radkey4
Device(config-sg-radius)# end
```

If all servers have equal transaction processing capability, one-third of all authentication and accounting transactions are directed toward the server 209.165.201.1. The remaining two-third of accounting transactions are load balanced equally between servers 209.165.201.1 and 209.165.201.2. The server 209.165.201.1 receives fewer authentication transactions because the server 209.165.201.1 has outstanding accounting transactions.

Example: Configure the preferred server with authentication servers as a subset of authorization servers

This example shows an authentication server group that uses servers 209.165.200.225 and 209.165.200.226 and an authorization server group that uses servers 209.165.200.225, 209.165.200.226, and 209.165.201.1. Both server groups have the preferred server flag enabled.

```
Device> enable
Device# configure terminal
Device(config)# aaa group server radius authentication-group
Device(config-sg-radius)# server 209.165.200.225 key radkey1
Device(config-sg-radius)# server 209.165.200.226 key radkey2
Device(config-sg-radius)# exit
Device(config)# aaa group server radius accounting-group
Device(config-sg-radius)# server 209.165.200.225 key radkey1
Device(config-sg-radius)# server 209.165.200.226 key radkey2
```

```
Device(config-sg-radius)# server 209.165.201.1 key radkey3
Device(config-sg-radius)# end
```

Authentication transactions are evenly split between server 209.165.200.225 and server 209.165.200.226. Servers 209.165.200.225 and 209.165.200.226 are preferred for both authentication and accounting transaction. Therefore, there is an equal distribution of authentication and accounting transactions across servers 209.165.200.225 and 209.165.200.226. The server 209.165.201.1 is relatively unused.

Example: Configure the preferred server with authentication servers as a superset of authorization servers

This example shows an authentication server group that uses servers 209.165.200.225, 209.165.200.226, and 209.165.201.1 and an authorization server group that uses servers 209.165.200.225 and 209.165.200.226. Both server groups have the preferred server flag enabled.

```
Device> enable
Device# configure terminal
Device(config)# aaa group server radius authentication-groupp
Device(config-sg-radius)# server 209.165.200.225 key radkey1
Device(config-sg-radius)# server 209.165.200.226 key radkey2
Device(config-sg-radius)# server 209.165.201.1 key radkey3
Device(config-sg-radius)# exit
Device(config)# aaa group server radius accounting-group
Device(config-sg-radius)# server 209.165.200.225 key radkey1
Device(config-sg-radius)# server 209.165.200.226 key radkey2
Device(config-sg-radius)# end
```

Initially, one-third of authentication transactions are assigned to each server in the authorization server group. As accounting transactions are generated for more sessions, servers 209.165.200.225 and 209.165.200.226 handle these transactions because the preferred server flag is on. For increased transaction volumes, authentication requests are redirected to server 209.165.201.1. Transaction requests authenticated by server 209.165.201.1 lack a preferred server setting and are divided between servers 209.165.200.225 and 209.165.200.226, negating the use of the preferred server flag. This configuration should be used cautiously.