Last Updated: 2/3/2012

Overview

This article addresses troubleshooting all issues that have to do with Radius authentication and accounting. Sometimes when troubleshooting issues, it is not obvious that radius authentication is the root cause. This article attempts to describe the various commands to determine where and if there is an issue. It explains how radius authentication and accounting tie into the call flow, what are the relevant radius configurables, the state machine behavior – how does it maintain the state of the configured servers, the radius probe feature, overload issues, recovery methods, and high-level descriptions of the types of issues you might expect to encounter, along with what information (traces, logs, counters, etc.) you might need to collect. It is not designed as a list of steps that should be followed in order, as there are so many things that could happen, and the specific information you have access to outside of the SXxx realm could vary widely. Therefore

developing such a universal guide would be difficult. Rather, you should use this article for background knowledge, reference, and ideas and guidelines that can be used throughout the troubleshooting process that you are responsible for formulating.

How Radius Authentication and Accounting works

A look at a monitor subscriber trace will make it obvious how radius authentication works. It is important to know when authentication and accounting take place so you know where to look for them and what to expect in a trace.

Whether it be SIP or MIP, a call starts with A11 protocol message exchange followed by some PPP negotiation.
For SIP, as soon as the username is received during PPP negotiation, radius authentication can take place, followed by further PPP negotiation where compression is negotiated along with the DNS and mobile IP address being given out.
For MIP, unlike for SIP, the username is not received during PPP negotiation. Rather, the DNS information is given out, and header compression negotiation takes place. At this point PPP is finished and MIP protocol takes over with MIP router advertisement and a MIP RRQ being received with the username, at which time radius authentication takes place.
So the key for both SIP and MIP radius authentication is getting the username.
Finally Radius accounting takes place. Radius accounting also takes place at call teardown.

It is recommended to study some example MIP and SIP traces to understand the full call flow and to see how authentication and accounting play a role. Grabbing a trace from your live network using monitor subscriber is as easy as any way to get started.

Here is an example of an authentication (properly qualified by the term “access”) request for a SIP call to give an idea of what's involved:

```
<<<<OUTBOUND  13:26:27:350 Eventid:23901(6)
RADIUS AUTHENTICATION Tx PDU, from 10.211.16.30:32798 to 66.174.77.10:1645
(302) PDU-dict=custom9
 Code: 1 (Access-Request)
 Id: 94
 Length: 302
 Authenticator: 53 3E A1 0B 53 C0 31 DA 19 8A E0 F3 73 61 F3 F5
      Calling-Station-Id = 111117742723726
      User-Name = 7746330615@test.com
      NAS-IP-Address = 10.211.16.30
      3GPP2-Correlation-Id = 2LxCl6hC
      3GPP2-MEID =
      CHAP-Challenge = F8 AE 5B 23 61 F5 F8 30 98 5A D0 C0 58 13 0F EF
      CHAP-Password = 01 1C 19 45 8B 40 E8 4A AF FD 05 F4 19 78 1B 85 <more>
      3GPP2-BSID = 001C000400D1
      3GPP2-RP-Session-ID = 1611188877
      3GPP2-Service-Option = HSPD
      3GPP2-User-Zone = 0
      Service-Type = Framed
      Framed-Protocol = PPP
```

```
      SN1-Service-Address = 10.211.16.30
      NAS-Port-Type = Wireless_Other
      3GPP2-Serving-PCF = 10.211.18.11
      Event-Timestamp = 1223645187
      SN1-Service-Type = PDSN
      3GPP2-IP-Technology = Simple-IP
      NAS-Port = 81484
      3GPP2-Session-Term-Capability =
Both_Dynamic_Auth_And_Reg_Revocation_in_MIP
      3GPP2-Service-Reference-ID = 01 04 00 01 02 04 00 01
                SR-ID = 1
                Main-SI-Indicator = Main-SI
      3GPP2-Pre-Paid-Acct-Capability = 01 06 00 00 00 03
                Available-In-Client = Supported_Volume_And_Duration


INBOUND>>>>>  13:26:27:385 Eventid:23900(6)
RADIUS AUTHENTICATION Rx PDU, from 66.174.77.10:1645 to 10.211.16.30:32798
(113) PDU-dict=custom9
 Code: 2 (Access-Accept)
 Id: 94
 Length: 113
 Authenticator: FA 6D CA 3D 1C 35 38 44 61 95 B6 22 CA 06 A9 B0
      Service-Type = Framed
      Framed-Protocol = PPP
      Framed-IP-Address = 255.255.255.254
      Framed-IP-Netmask = 255.255.255.255
      Framed-MTU = 1514
      Framed-Compression = None
      Class = 42 57 53 00 01 00 0E 2F 00 16 00 02 03 07 01 27 <more>
      Session-Timeout = 86100
      Idle-Timeout = 86340
      Acct-Interim-Interval = 44000
      Framed-Pool = RestrictAcc01
```

Here is an example of a radius accounting exchange for the same call above. This is an accounting start. Note that the responses for accounting messages do not contain any real information like authentication responses do.

```
<<<<OUTBOUND  13:26:28:037 Eventid:24901(6)
RADIUS ACCOUNTING Tx PDU, from 10.211.16.30:32798 to 66.174.77.10:1646 (695)
PDU-dict=custom9
 Code: 4 (Accounting-Request)
 Id: 108
 Length: 695
 Authenticator: 2B 1C F7 30 66 69 56 1C 15 CC A9 DB 0E 5B 25 96
      User-Name = 7746330615@vzw3g.com
      Calling-Station-Id = 111117742723726
      NAS-IP-Address = 10.211.16.30
      Acct-Status-Type = Start
      Acct-Session-Id = 093C66E3
      3GPP2-Correlation-Id = 2LxCl6hC
      3GPP2-MEID =
      3GPP2-BSID = 001C000400D1
      3GPP2-RP-Session-ID = 1611188877
```

```
        3GPP2-Service-Option = HSPD
        3GPP2-User-Zone = 0
        Service-Type = Framed
        Framed-Protocol = PPP
        NAS-Port-Type = Wireless_Other
        3GPP2-Serving-PCF = 10.211.18.11
        Acct-Authentic = RADIUS
        SN1-Local-IP-Address = 66.174.120.30
        SN1-Primary-DNS-Server = 66.174.95.44
        SN1-Secondary-DNS-Server = 69.78.96.14
        SN1-Primary-NBNS-Server = 0.0.0.0
        SN1-Secondary-NBNS-Server = 0.0.0.0
        3GPP2-Always-On = Inactive
        Framed-MTU = 1500
        Framed-Compression = None
        SN1-PPP-Data-Compression = None
        SN1-PPP-Data-Compression-Mode = Normal
        SN1-VPN-Name = destination
        SN1-Rulebase = SIP
        SN1-Proxy-MIP = Disabled
        Class = 42 57 53 00 01 00 0E 2F 00 16 00 02 03 07 01 27 <more>
        SN1-Firewall-Policy =
        SN1-VPN-ID = 3
        3GPP2-IP-Technology = Simple-IP
        3GPP2-Comp-Tunnel-Indicator = No-Tunnel
        3GPP2-Forward-Mux-Option = 2337
        3GPP2-Reverse-Mux-Option = 2337
        3GPP2-Forward-Fundamental-Rate = 0
        3GPP2-Reverse-Fundamental-Rate = 0
        3GPP2-Forward-Traffic-Type = Primary
        3GPP2-Reverse-Traffic-Type = Primary
        3GPP2-Fundamental-Frame-Size = 20ms
        3GPP2-Forward-Fundamental-RC = 3
        3GPP2-Reverse-Fundamental-RC = 3
        3GPP2-Air-QOS = 12
        3GPP2-Airlink-Sequence-Number = 1
        3GPP2-Airlink-Record-Type = Active-Start
        3GPP2-Number-Active-Transitions = 0
        3GPP2-SDB-Input-Octets = 0
        3GPP2-SDB-Output-Octets = 0
        3GPP2-Num-SDB-Input = 0
        3GPP2-Num-SDB-Output = 0
        3GPP2-Num-Bytes-Received-Total = 0
        3GPP2-Beginning-Session = True
        Event-Timestamp = 1223645188
        3GPP2-Service-Reference-ID = 01 04 00 01 02 04 00 01
                  SR-ID = 1
                  Main-SI-Indicator = Main-SI
        Framed-IP-Address = 72.98.40.38
        Framed-IP-Netmask = 255.255.255.255
        SN1-Service-Type = PDSN
        NAS-Port = 81484


INBOUND>>>>>  13:26:28:042 Eventid:24900(6)
RADIUS ACCOUNTING Rx PDU, from 66.174.77.10:1646 to 10.211.16.30:32798 (20)
PDU-dict=custom9
```

```
 Code: 5 (Accounting-Response)
 Id: 108
 Length: 20
 Authenticator: 6F 55 66 1F 6C 2B A8 38 1D 5D 23 B9 38 2C 56 E3
```

And here is the accounting exchange at call teardown. This is an accounting stop.

Note that Accounting starts and stops also occur during a call when intra-pdsn handoffs or ppp renegotiations occur. There is also an accounting "Interim-Update" type for communicating accounting information without marking a new accounting session.

```
<<<<OUTBOUND  13:31:33:152 Eventid:24901(6)
RADIUS ACCOUNTING Tx PDU, from 10.211.16.30:32798 to 66.174.77.10:1646 (875)
PDU-dict=custom9
 Code: 4 (Accounting-Request)
 Id: 221
 Length: 875
 Authenticator: 68 92 88 5A 0C 03 64 28 19 96 68 41 88 D9 01 2D
      User-Name = 7746330615@vzw3g.com
      Calling-Station-Id = 111117742723726
      NAS-IP-Address = 10.211.16.30
      Acct-Status-Type = Stop
      Acct-Session-Id = 093C66E3
      3GPP2-Correlation-Id = 2LxCl6hC
      3GPP2-MEID =
      3GPP2-BSID = 001C000400D1
      3GPP2-RP-Session-ID = 1611188877
      3GPP2-Service-Option = HSPD
      3GPP2-User-Zone = 0
      Service-Type = Framed
      Framed-Protocol = PPP
      NAS-Port-Type = Wireless_Other
      3GPP2-Serving-PCF = 10.211.18.11
      Acct-Authentic = RADIUS
      SN1-Local-IP-Address = 66.174.120.30
      SN1-Primary-DNS-Server = 66.174.95.44
      SN1-Secondary-DNS-Server = 69.78.96.14
      SN1-Primary-NBNS-Server = 0.0.0.0
      SN1-Secondary-NBNS-Server = 0.0.0.0
      3GPP2-Always-On = Inactive
      Framed-MTU = 1500
      Framed-Compression = None
      SN1-PPP-Data-Compression = None
      SN1-PPP-Data-Compression-Mode = Normal
      SN1-VPN-Name = destination
      SN1-Rulebase = SIP
      SN1-Proxy-MIP = Disabled
      Class = 42 57 53 00 01 00 0E 2F 00 16 00 02 03 07 01 27 <more>
      SN1-Firewall-Policy =
      SN1-VPN-ID = 3
      3GPP2-IP-Technology = Simple-IP
      3GPP2-Comp-Tunnel-Indicator = No-Tunnel
      3GPP2-Release-Indicator-custom9 = PPP-Termination
      SN1-PPP-Progress-Code = Call-Simple-IP-Connected
      3GPP2-Forward-Mux-Option = 2337
```

```
      3GPP2-Reverse-Mux-Option = 2337
      3GPP2-Forward-Fundamental-Rate = 0
      3GPP2-Reverse-Fundamental-Rate = 0
      3GPP2-Forward-Traffic-Type = Primary
      3GPP2-Reverse-Traffic-Type = Primary
      3GPP2-Fundamental-Frame-Size = 20ms
      3GPP2-Forward-Fundamental-RC = 3
      3GPP2-Reverse-Fundamental-RC = 3
      3GPP2-Air-QOS = 12
      3GPP2-Airlink-Sequence-Number = 4
      3GPP2-Airlink-Record-Type = Active-Stop
      3GPP2-Bad-PPP-Frame-Count = 0
      3GPP2-Number-Active-Transitions = 1
      3GPP2-SDB-Input-Octets = 0
      3GPP2-SDB-Output-Octets = 0
      3GPP2-Num-SDB-Input = 0
      3GPP2-Num-SDB-Output = 0
      3GPP2-Num-Bytes-Received-Total = 1095
      Acct-Input-Octets = 1060
      Acct-Output-Octets = 1303
      3GPP2-Active-Time = 20
      Acct-Input-Packets = 7
      Acct-Output-Packets = 7
      3GPP2-Rsvp-Signal-In-Count = 0
      3GPP2-Rsvp-Signal-Out-Count = 0
      3GPP2-Rsvp-Signal-In-Packets = 0
      3GPP2-Rsvp-Signal-Out-Packets = 0
      SNA1-PPP-Unfr-data-In-Oct = 1060
      SNA1-PPP-Unfr-data-Out-Oct = 1303
      Acct-Session-Time = 305
      3GPP2-Session-Continue = False
      3GPP2-Last-Activity = 1223645493
      SN1-Disconnect-Reason = Remote-Disconnect
      Event-Timestamp = 1223645493
      3GPP2-Service-Reference-ID = 01 04 00 01 02 04 00 01
                SR-ID = 1
                Main-SI-Indicator = Main-SI
      Framed-IP-Address = 72.98.40.38
      Framed-IP-Netmask = 255.255.255.255
      SN1-Service-Type = PDSN
      Acct-Termination-Cause = User_Request
      NAS-Port = 81484


INBOUND>>>>>  13:31:33:157 Eventid:24900(6)
RADIUS ACCOUNTING Rx PDU, from 66.174.77.10:1646 to 10.211.16.30:32798 (20)
PDU-dict=custom9
 Code: 5 (Accounting-Response)
 Id: 221
 Length: 20
 Authenticator: B7 F4 4B 4A A9 FA B3 91 FD D6 E6 1F 36 75 88 30
```

Radius configuration and radius packet processing rules

The above scenario is the simplest, and if all requests worked like this, then there would be little to discuss. In reality, rejections take place, and these rejections may be legitimate based on the data being sent. Network congestion, dropped packets, overloaded radius server resulting in slow response, missing, extra, or wrong attributes or attribute values in the packets being exchanged, etc., are all possibilities.

To manage and plan for various scenarios related to packet exchange and timing, there are configurables to specify how many times to resend packets, the amount of time in between resends, how long to mark a server down, which server(s) to try and in what order, etc. The concepts apply the same for authentication as for accounting, but the configurables are slightly different for both (for example: "radius timeout" vs. "radius accounting timeout"), as you may want to have different behavior for both. In examples below, the default values are shown and the accounting version has parentheses around the word “accounting”.

Most of the configuration for radius will be in a specifically named group, and all contexts will have a default group which can be configured as follows. Many times configurations will have just one group, the default group.

[local]CSE2# config
[local]CSE2(config)# context destination
[destination]CSE2(config-ctx)# aaa group default
[destination]CSE2(config-aaa-group)#

If specific named aaa groups are used, they are pointed to by the following statement configured in a subscriber profile, which in turn is pointed to by a domain statement.

aaa group <group name>

Note: The system first checks the specific aaa group, and then checks the aaa group default for additional configurables not defined in the specific group.

Before moving on further with the details of various configurables, it is important to point out the command that summarizes all the values assigned to all the configurables in the various aaa group configurations. This allows quick viewing of all the configurables including default values without having to examine the configuration manually, and possibly help avoiding making mistakes when assuming certain settings. These commands report across all contexts:

show aaa group all
show aaa group name <group name>

The most important configurable will be the radius access and accounting servers themselves. Here is an example:

radius server 66.174.86.10 key testtesttesttest port 1645 priority 1

radius server 66.174.77.10 key testtesttesttest port 1645 priority 2
radius accounting server 66.174.86.10 key testtesttesttest port 1646 priority 1
radius accounting server 66.174.77.10 key testtesttesttest port 1646 priority 2

In addition, the NAS IP address is also required to be defined, which is the IP address on an interface in the context FROM which radius requests are sent. If not defined, requests are not sent and monitor subscriber traces may not post an obvious error (you just won't see any radius requests being sent without any indication why).

radius attribute nas-ip-address address 10.211.41.129

The following command can be used to determine if a NAS IP address exists. In this example, it does not:

[source]CSE01# show radius client status

RADIUS client status: DOWN (no active interface)

Active nas-ip-address: NONE
Configured primary nas-ip-address: NONE DOWN
Configured backup nas-ip-address: NONE DOWN

Note that because both authentication and accounting are usually handled by the same server, a different port number is used to differentiate authentication vs. accounting traffic on the radius server. For the STxx/ASR5K side, the source port number is not specified and is chosen by the chassis. The port number for any given

Normally multiple access and accounting servers will be specified for redundancy purposes. Either a round robin or prioritized order can be configured:

radius (accounting) algorithm {first-server | round-robin}

The first-server option will result in ALL requests sent to the server with the lowest-numbered priority. Only when retry failures occur, or worse, a server is marked down, will the server with the next priority be tried. More on this below.

When a radius (accounting or access) request is sent, a reply is expected. When a reply is not received within the timeout period (seconds):

radius (accounting) timeout 3

The request is resent up to the number of times specified:

radius (accounting) max-retries 5

This means that a request can be sent a total of max-retries + 1 times until it gives up on the particular radius server being tried. At this point, it will try the same sequence to the next radius server in order. If each of the servers have been tried max-retries + 1 times without response, then the PDSN will reject the call, assuming there is no other reason for failure up to that point.

As a sidenote, there are configurables that allow for users to have access even if authentication and accounting fail due to timeouts to all servers, though a commercial deployment would not likely want to implement this.

radius allow (accounting-) authentication-down

Also, there are configurables that can limit the absolute total number of transmissions of a particular request across all the configured servers, and these are disabled by default:

radius (accounting) max-transmissions 256

For example if this is set = 1, then even if there is a secondary server, it will never be attempted because only one attempt for a specific subscriber setup will ever even be attempted.

The "show radius counters all" command is very valuable in tracking success and failures on a server basis, and it is very important to understand the meaning of the various counters that compose this command, as it may not be obvious. In the following output, note "Access-Request Sent" = 1, while "Access-Request Retried" = 3. So, any given new request to a particular radius server is only counted once, and all the retries are counted separately. In this case, that is a total of 3 + 1 = 4 access requests sent. Note the counter "Access-Request Timeouts" = 1. A single timeout occurs only when ALL the retries fail, so in this case, 3 retries without a response result in 1 Timeout (not 4). This will happen across all of the configured servers until there is success, or all attempts have failed. So pay attention to the counters that are tracked for each server separately. Here is an example of this, where:

radius max-retries 3
radius server 192.168.50.200 encrypted key 01abd002c82b4a2c port 1812 priority 1
radius server 192.168.50.250 encrypted key 01abd002c82b4a2c port 1812 priority 2

```
[destination]CSE2# show radius counters all

   Server-specific Authentication Counters
   ---------------------------------------

   Authentication server address 192.168.50.200, port 1812:
     Access-Request Sent:                                  1
     Access-Request with DMU Attributes Sent:              0
     Access-Request Pending:                               0
     Access-Request Retried:                               3
     Access-Request with DMU Attributes Retried:           0
     Access-Challenge Received:                            0
     Access-Accept Received:                               0
     Access-Reject Received:                               0
     Access-Reject Received with DMU Attributes:           0
```

```
     Access-Request Timeouts:                                  1
     Access-Request Current Consecutive Failures in a mgr:     1
     Access-Request Response Bad Authenticator Received:       0
     Access-Request Response Malformed Received:               0
     Access-Request Response Malformed Attribute Received:     0
     Access-Request Response Unknown Type Received:            0
     Access-Request Response Dropped:                          0
     Access-Request Response Last Round Trip Time:             0.0 ms
…

  Authentication server address 192.168.50.250, port 1812:
     Access-Request Sent:                                      1
     Access-Request with DMU Attributes Sent:                  0
     Access-Request Pending:                                   0
     Access-Request Retried:                                   3
     Access-Request with DMU Attributes Retried:               0
     Access-Challenge Received:                                0
     Access-Accept Received:                                   0
     Access-Reject Received:                                   0
     Access-Reject Received with DMU Attributes:               0
     Access-Request Timeouts:                                  1
     Access-Request Current Consecutive Failures in a mgr:     1
     Access-Request Response Bad Authenticator Received:       0
     Access-Request Response Malformed Received:               0
     Access-Request Response Malformed Attribute Received:     0
     Access-Request Response Unknown Type Received:            0
     Access-Request Response Dropped:                          0
     Access-Request Response Last Round Trip Time:             0.0 ms
```

Note also that timeouts are NOT counted as failures, the result being that the number of Access-Accept received and Access-Reject received will not add up to Access-Request Sent if there are any timeouts.

For MIP, it can be even more complicated than just described, because as the authentications are failing, there is no MIP RRP being sent, and the mobile may continue to initiate new MIP RRQs because it has not received a MIP RRP. Each new MIP RRQ causes the PDSN to send a new Authentication request which itself can have its own series of retries. This can be seen in the Id field at the top of a packet trace – it will be unique for each set of retries. The result is that the counters for Sent, Retried, and Timeout can be much higher than expected for the number of calls received. There is an option that can be enabled to minimize these extra re-tries, and it can be set in the FA (but not on the HA) service: "authentication mn-aaa <6 choices here> optimize-retries"

Note: Over an un-monitored time period, it is difficult to draw any conclusions from the counter values or the relationships amongst counters. To make accurate conclusions, the best approach is to reset the counters and monitor them over a period of time when the issue you are troubleshooting is occurring.

Radius Server State Machine Behavior

There are actually two different models/algorithms/approaches to choose from to determine the status of a radius server and when to try a different server if failures are occurring:

The original approach involves keeping track of the number of failures that have occurred in a row for a particular aaamgr process. A aaamgr process is responsible for all radius message processing and exchange with a radius server, and many aaamgr process will exist in a chassis, each paired with sessmgr processes (which are main processes responsible for call control). (You can view all the aaamgr processes with the "show task resources" command) A particular aaamgr process will therefore be processing radius messages for many calls, not just a single call, and this algorithm involves tracking how many times in a row a particular aaamgr process has failed to get a response to the same request it has had to resend - an "Access-Request Timeout" as described above. The respective counter "Access-Request Current Consecutive Failures in a mgr" is incremented when this occurs, and the "show radius accounting (or authentication) servers detail" command will indicate the timestamps of the radius state change from Active to Not Responding (but no SNMP trap or logs will be generated). For example:

```
[source]txhoupdsn1> show radius accounting servers detail
Friday November 28 23:23:34 UTC 2008

+-----Type:       (A) - Authentication   (a) - Accounting
|                 (C) - Charging         (c) - Charging Accounting
|                 (M) - Mediation        (m) - Mediation Accounting
|
|+----Preference: (P) - Primary          (S) - Secondary
||
||+---State:      (A) - Active           (N) - Not Responding
|||               (D) - Down             (W) - Waiting Accounting-On
|||               (I) - Initializing     (w) - Waiting Accounting-Off
|||               (a) - Active Pending   (U) - Unknown
|||
|||+--Admin       (E) - Enabled          (D) - Disabled
||||  Status:
||||
||||+-Admin
||||| status      (O) - Overridden       (.) - Not Overridden
||||| Overridden:
|||||
vvvvv IP              PORT  GROUP
----- --------------- ----- ----------------------
aPNE. 172.28.220.5    1813  default

Event History:
2008-Nov-28+23:18:36       Active
2008-Nov-28+23:18:57       Not Responding
2008-Nov-28+23:19:12       Active
2008-Nov-28+23:19:30       Not Responding
2008-Nov-28+23:19:36       Active
2008-Nov-28+23:20:57       Not Responding
2008-Nov-28+23:21:12       Active
2008-Nov-28+23:22:31       Not Responding
2008-Nov-28+23:22:36       Active
2008-Nov-28+23:23:30       Not Responding
```

If this counter reaches the value configured (Default = 4) without ever being reset:

radius (accounting) detect-dead-server consecutive-failures 4

Then this server will be marked "Down" for the period (minutes) configured:

radius (accounting) deadtime 10

An SNMP trap and logs will be triggered as well, for example, for authentication and accounting respectively:

```
Fri Jan 30 06:17:19 2009 Internal trap notification 39
(AAAAuthSvrUnreachable) server 2 ip address 66.174.86.10
Fri Jan 30 06:22:19 2009 Internal trap notification 40 (AAAAuthSvrReachable)
server 2 ip address 66.174.86.10

Fri Nov 28 21:59:12 2008 Internal trap notification 42 (AAAAccSvrUnreachable)
server 6 ip address 172.28.221.178
Fri Nov 28 22:28:29 2008 Internal trap notification 43 (AAAAccSvrReachable)
server 6 ip address 172.28.221.178

2008-Nov-28+21:59:12.899 [radius-acct 24006 warning] [8/0/518 <aaamgr:231>
aaamgr_config.c:1060] [context: source, contextID: 2]  [software internal
security config user critical-info] Server 172.28.221.178:1813 unreachable

2008-Nov-28+22:28:29.280 [radius-acct 24007 info] [8/0/518 <aaamgr:231>
aaamgr_config.c:1068] [context: source, contextID: 2]  [software internal
security config user critical-info] Server 172.28.221.178:1813 reachable
```

Also, the "show radius accounting (or authentication) servers detail" command will indicate the timestamps of the state changes:

```
vvvvv IP              PORT  GROUP
----- --------------- ----- ----------------------
aSDE. 172.28.221.178  1813  default

Event History:
2008-Nov-28+21:59:12       Down
2008-Nov-28+22:28:29       Active
2008-Nov-28+22:28:57       Not Responding
2008-Nov-28+22:32:12       Down
2008-Nov-28+23:01:57       Active
2008-Nov-28+23:02:12       Not Responding
2008-Nov-28+23:05:12       Down
2008-Nov-28+23:19:29       Active
2008-Nov-28+23:19:57       Not Responding
2008-Nov-28+23:22:12       Down
```

If there is only one server configured, then it will not be marked down, as that would be critical for successful call setup.

Worth mentioning is that there is another parameter that can be configured on the detect-dead-server config line called “response-timeout”. When specified, a server will be marked down only when the consecutive failures and response-timeout conditions are both met. The response-timeout specifies a period of time when NO responses are received to ALL the requests sent to a particular server. (Note that this timer would be continually reset as responses are received.) This condition would be expected when either a radius server or the network connection is completely down, vs. partially compromised/degraded.

The use case for this would be a scenario where a burst in radius traffic causes the consecutive failures to trigger, but marking a server down immediately as a result is not desired. Rather, the server will only be marked down after a specific period of time passes where no responses are received, effectively representing true server un-reachability.

This method just discussed of controlling radius state machine changes is dependent on looking at all aaamgr processes and finding one that triggers the condition of failed retries. This method is subject to some degree to some randomness of failures, and so may not be the ideal algorithm to detecting failures.

Another method of detecting radius server reachability is using dummy keepalive test messages. This involves the constant sending of fake radius messages instead of monitoring live traffic. Another advantage of this method is that it is always active, vs. with the aaamgr approach, where there could be periods where no radius traffic is sent, and so there is no way to know if a problem exists during those times, resulting in delayed detection when attempts do start occurring. Also when a server is marked down, these keepalives continue to be sent so that the server can be marked up as soon as possible.

Here are the various configurables relevant to this approach:

radius (accounting) detect-dead-server keepalive
radius (accounting) keepalive interval 30
radius (accounting) keepalive retries 3
radius (accounting) keepalive timeout 3
radius (accounting) keepalive consecutive-response 1
radius (accounting) keepalive username Test-Username
radius keepalive encrypted password 2ec59b3188f07d9b49f5ea4cc44d9586
radius (accounting) keepalive calling-station-id 000000000000000
radius keepalive valid-response access-accept

The command “radius (accounting) detect-dead-server keepalive” turns on the keep-alive approach instead of the consecutive failures in a aaamgr approach. In the example above, the system will send a test message with username Test-Username and password Test-Username every 30 seconds, and will retry every 3 seconds if no response is received, and will retry up to 3 times, after which it will mark the server down. Once it gets its first response, it will mark it back up again.

Here is an example authentication request/response for the above settings:

```
<<<<OUTBOUND  17:50:12:657 Eventid:23901(6)
RADIUS AUTHENTICATION Tx PDU, from 192.168.50.151:32783 to
192.168.50.200:1812 (142) PDU-dict=starent-vsa1
 Code: 1 (Access-Request)
 Id: 16
 Length: 142
 Authenticator: 51 6D B2 7D 6A C6 9A 96 0C AB 44 19 66 2C 12 0A
      User-Name = Test-Username
      User-Password = B7 23 1F D1 86 46 4D 7F 8F E0 2A EF 17 A1 F3 BF
      Calling-Station-Id = 000000000000000
      Service-Type = Framed
      Framed-Protocol = PPP
      NAS-IP-Address = 192.168.50.151
      Acct-Session-Id = 00000000
      NAS-Port-Type = HRPD
      3GPP2-MIP-HA-Address = 255.255.255.255
      3GPP2-Correlation-Id = 00000000
      NAS-Port = 4294967295
      Called-Station-ID = 00


INBOUND>>>>>  17:50:12:676 Eventid:23900(6)
RADIUS AUTHENTICATION Rx PDU, from 192.168.50.200:1812 to
192.168.50.151:32783 (34) PDU-dict=starent-vsa1
 Code: 2 (Access-Accept)
 Id: 16
 Length: 34
 Authenticator: 21 99 F4 4C F8 5D F8 28 99 C6 B8 D9 F9 9F 42 70
      User-Password = testpassword
```

The same SNMP traps are used to signify the unreachable/down and reachable/up radius states as with the original approach:

```
Fri Feb 27 17:54:55 2009 Internal trap notification 39
(AAAAuthSvrUnreachable) server 1 ip address 192.168.50.200
Fri Feb 27 17:57:04 2009 Internal trap notification 40 (AAAAuthSvrReachable)
server 1 ip address 192.168.50.200
```

The “show radius counters all” has a section for keeping track of the keepalive requests for authentication and accounting as well – here are the authentication counters:

```
   Server-specific Keepalive Auth Counters
   ---------------------------------------
     Keepalive Access-Request Sent:                          33
     Keepalive Access-Request Retried:                       3
     Keepalive Access-Request Timeouts:                      4
     Keepalive Access-Accept Received:                       29
     Keepalive Access-Reject Received:                       0
     Keepalive Access-Response Bad Authenticator Received:   0
     Keepalive Access-Response Malformed Received:           0
     Keepalive Access-Response Malformed Attribute Received: 0
     Keepalive Access-Response Unknown Type Received:        0
     Keepalive Access-Response Dropped:                      0
```

Radius Probe feature

There is another keep-alive feature that also exists, but working in the opposite direction. It is called authentication probe and is implemented only on Home Agent chassis. In the radius authentication server configuration line, you enable this feature by specifying a username and password for authentication test messages – similar messages as with the keep-alive feature:

radius server 66.174.77.10 key testkey port 1645 priority 1 probe probe-username dave@wireless.com probe-password testpassword

There are configurables associated with this feature as well:

radius probe-interval 30
radius probe-timeout 3
radius probe-max-retries 5

The difference is that the STxx/ASR5K does not care if it does not receive replies to the messages. More importantly, the radius server is expecting these requests with the specific username and password, and uses this information to track whether or not this HA is available (up and running) or not. If it stops getting requests, then it will mark the HA down, and will no longer provide that particluar HA's ip address in Radius access accept messages to any PDSNs where New calls are trying to authenticate and get an HA IP address with which to complete MIP calls. Existing calls will continue to remain up though. This is the same as the keepalive feature in that the STxx is initiating the request, but different in that the radius server is the one that cares about the request.

There are also counters for the radius probe that are part of the radius counters command:

```
Server-specific Probing Counters
--------------------------------
State:                                              Active
Number of transactions issued:                      109
Number of successful transactions:                  109
Number of failed transactions:                      0
Last successful transaction time:             Fri Feb 06 15:52:41 2009
Last failed transaction time:                       -
Last roundtrip time:                                75.1 ms
```

The radius probe feature is also used as part of another major product feature that provides for HA chassis redundancy called Inter Chassis Session Recovery (ICSR), where there is a sister chassis that is in standby mode and which is keeping track of all the sessions in the active chassis. If the active chassis goes down, then the standby chassis takes over and becomes active instantaneously with all the session intact as if nothing ever happened. One of the ways that an active chassis is monitored for being in a good state is using the AAA probe feature. If the radius probes are not responded to, then the chassis, which is monitoring for this, switches to standby state and the chassis that was standby becomes active at the same time. The SNMP trap that gets triggered is SRPAAAUnreachable, and the trap indicating change to standby state is SRPStandby. The SRPAAAUnreachble will trigger when all the monitored AAA Probes are

marked down. Here is an example where one can follow the timeline dictated by the configurables and match with the switchover time:

```
Tue Jan 31 09:17:30 2012 Internal trap notification 125
(SRPAAAUnreachable)  vpn SRP ipaddr 66.174.98.10 port 1645
Tue Jan 31 09:17:30 2012 Internal trap notification 125
(SRPAAAUnreachable)  vpn SRP ipaddr 66.174.99.10 port 1645
Tue Jan 31 09:17:30 2012 Internal trap notification 126
(SRPSwitchoverInitiated)  vpn SRP ipaddr 10.222.151.212
Tue Jan 31 09:17:31 2012 Internal trap notification 121 (SRPStandby)  vpn SRP
ipaddr 10.222.151.212 rtmod 4
```

```
     State:                                          Down
     Number of transactions issued:                  172858
     Number of successful transactions:              172856
     Number of failed transactions:                  2
     Last successful transaction time:               Tue Jan 31 09:16:42 2012
     Last failed transaction time:                   Tue Jan 31 09:17:30 2012
     Last roundtrip time:                            19.7 ms
```

Every 30 seconds a probe is sent, and if there is no response it will try every 3 seconds for a total of 5 retries (or 6 total attempts) which means 6 * 3 = 18 seconds, so 9:16:42 + 30s + 18s = 9:17:30 which exactly matches the failed transaction time AND the SNMP traps indicating AAA unreachable and SRP switchover to standby.

Troubleshooting basic Radius connectivity issues

An earlier part of this article was devoted to explaining exactly how radius authentication and accounting works with regards to ensuring that requests get responded to. It is important to understand those explanations if you are going to be troubleshooting connectivity issues. Note that a redundant design should help minimize impact if connectivity to a specific server is compromised, but if connectivity to all servers is compromised, then you should be ready to act quickly to take the necessary diagnosis and troubleshooting steps.

You may need to look at some of the STxx radius configurables and modify them to work better with your network and radius servers. Certainly you don't want to hide problems by increasing timers, retries, etc., some of which have been described in detail earlier, but you should at least consider if the values currently being used are too stringent.

Some ideas include increasing the timeout or number of retries or consecutive failures in a aaamgr

If the radius server is getting overloaded, you could decrease the load by decreasing the value (default 256) configured for "radius (accounting) max-outstanding", which sets a limit on the number of outstanding (unanswered) requests for any given aaamgr process. If the limit is reached, logs may indicate this: "Failed to assign message id for radius authentication server x.x.x.x:1812". There is no way to specifically rate-limit authentication/accounting messages.

Sometimes it is not a problem of connectivity but of increased accounting traffic, which is not a problem with radius at all, but pointing to another area, such as increased ppp renegotiations which are causing more accounting starts and stops. So you may need to troubleshoot outside of radius to find a cause for the symptoms being observed.

One feature for accounting that may be turned on is archiving, enabled with the command "radius accounting archive". This allows for accounting packets that have failed to be responded to, to be re-queued and sent at a later time when the aaamgr processes have free cycles not busy processing authentication requests. The idea is that timeliness of accounting packets is not as important as authentication, though this may not be true in all situations. If the radius system is overloaded, this might be a feature you could do without, and you could turn off the archiving. Note though there is no way to flush out existing archived messages. The show radius counter "Accounting-Request Pending" will show the number of archived accounting messages, which could increase over time if the accounting server gets bogged down. On the other hand, the "Access-Request Pending" counter for Access-Requests will decrement when it has given up trying to authenticate for a given call, which will always be a limited period of time (vs. accounting which could theoretically try forever).

If during the troubleshooting process it has been decided to remove a radius authentication or accounting server from the list of live servers for whatever reason, there is a (non-config) command that will take a server out of service indefinitely until it is desired to put it back in service. This is a cleaner approach than having to remove it from the configuration manually:

{disable | enable} radius (accounting) server x.x.x.x

```
[source]CSE2# show radius authentication servers detail

+-----Type:       (A) - Authentication   (a) - Accounting
|                 (C) - Charging         (c) - Charging Accounting
|                 (M) - Mediation        (m) - Mediation Accounting
|
|+----Preference: (P) - Primary          (S) - Secondary
||
||+---State:      (A) - Active           (N) - Not Responding
|||               (D) - Down             (W) - Waiting Accounting-On
|||               (I) - Initializing     (w) - Waiting Accounting-Off
|||               (a) - Active Pending   (U) - Unknown
|||
|||+--Admin       (E) - Enabled          (D) - Disabled
||||  Status:
||||
||||+-Admin
||||| status      (O) - Overridden       (.) - Not Overridden
||||| Overridden:
|||||
vvvvv IP              PORT  GROUP
----- --------------- ----- ----------------------
APNDO 192.168.50.200  1812  default
```

If you want to change the priority of the existing servers, you must first remove the server that you want to change the priority for, and then add it back in. When removing a server, you don't need to specify the password/key, but you do need to specify the port, for example:

no radius server 192.168.50.200 port 1812
radius server 192.168.50.200 encrypted key 01abd002c82b4a2c port 1812 priority 3

Data collection

There may be circumstances where packet captures need to be done between the STxx NAS IP address and the radius server(s) to confirm where delays or packet drops are occurring. This information along with logs from the radius server and STxx counters could help paint the picture of what is going on.

There may be circumstances also where logging for aaamgr, aaa-client, radius-auth, and radius-acct facilities need to be turned up beyond the default error level. You should only do this upon recommendation from Cisco Support, as increasing too high may put too much stress on the system and impact subscribers. Certainly in a test lab environment you could do this, especially when troubleshooting functional issues that don't require any load to reproduce. The logging configuration command in the local context:

logging filter runtime facility <aaamgr | aaa-client | radius-auth | radius-acct> level <warning | unusual | info | trace | debug>

Logging monitor for a specific subscriber for which an issue is reproducible might also be useful in order to see underlying messages that would not be displayed by monitor subscriber.

With regards to testing a specific behavior without having to set it up in your radius server, you can do so by forcing local authentication by creating a subscriber template with the exact name of the subscriber – radius authentication will not be attempted in that case. You can then confirm successful authentications with the command: "show aaa local counters".

Traps and Alarms

There are a number of ways that you may be alerted of radius related issues. The aforementioned SNMP traps for servers being marked down is one of the most obvious, and should be investigated immediately. There are also alarms and traps that can be triggered for failed authentication rates. The values for the triggers should be planned carefully, as there will always be failures, legitimate ones, and you don't want the failures to constantly cause false alarms.

The same triggers are available for authentication and accounting. A poll interval rate is specified as the period over which the entity being monitored is monitored. The difference between aaa-auth-failure/aaa-acct-failure and aaa-auth-failure-rate/aaa-acct-failure-rate is that the former measures an absolute number of failures within the specified time, while the latter measures percentage failures out of the total attempts. Taking the percentage approach is often

the most logical for measuring rates because it does not depend on traffic load which varies during the day, night, weekends and holidays,and also over time as the subscriber base grows.

Note that failures includes the sum of rejections (implies failure response received) and no responses at all.

For a given metric, you need to specify a poll interval, threshold and clear values, and whether to enable monitoring or not. For example:

threshold poll aaa-auth-failure interval 600
threshold poll aaa-auth-failure-rate interval 900
threshold poll aaa-acct-failure interval 300
threshold poll aaa-acct-failure-rate interval 300
threshold poll aaa-retry-rate interval 300

threshold aaa-auth-failure 1000 clear 500
threshold aaa-auth-failure-rate 25 clear 20
threshold aaa-acct-failure 1000 clear 500
threshold aaa-acct-failure-rate 25 clear 20
threshold aaa-retry-rate 25 clear 20

threshold monitoring aaa-auth-failure
threshold monitoring aaa-acct-failure
threshold monitoring aaa-retry-rate

Here are example snmp traps and logs for aaa-auth-failure for alarm trigger and clearing:

```
Fri Jun 27 01:00:02 2008 Internal trap notification 216 (ThreshAAAAuthFail) threshold
1000 measured value 2370
Fri Jun 27 01:10:02 2008 Internal trap notification 217 (ThreshClearAAAAuthFail)
threshold 500 measured value 25

2008-Jun-27+01:10:02.511 [alarmctrl 65200 info] [8/0/494 <evlogd:0>
alarmctrl.c:273] [software internal system critical-info] Alarm cleared: id
50020c643b920000: <23:aaa-auth-failure> has reached or exceeded the
configured threshold <1000>, the measured value is <2370>. It is detected at
<System>.
2008-Jun-27+01:00:02.549 [alarmctrl 65201 info] [8/0/494 <evlogd:0>
alarmctrl.c:180] [software internal system critical-info] Alarm condition: id
50020c643b920000 (Minor): <23:aaa-auth-failure> has reached or exceeded the
configured threshold <1000>, the measured value is <2370>. It is detected at
<System>.
```

The alarm would look something like this:

```
[source]DO-PDSN# show alarm outstanding
Sev Object     Event
--- ---------- ------------------------------------------------------------
MN  Chassis    <23:aaa-auth-failure> has reached or exceeded the configured
threshold <1000>, the measured value is <2370>. It is detected at <System>.
```

Radius counters and status

Whenever running radius commands, you need to be in the context where the aaa group that you are troubleshooting is located.

show radius counters {all | summary}

show radius {accounting | authentication} servers detail

These have already been discussed in detail in earlier sections. A few things to note:
- show support details does not contain "show radius (accounting | authentication) servers detail"
- "show radius servers detail" only saves the last ten radius state changes – so you may want to run the command multiple times to get a complete history of troubleshooting over an extended period.
- clearing the radius counters is a good idea if monitoring the counters over a time period, otherwise it takes more time to make the math calculations and do the comparisons each time it is run.
- The summary version of show radius counters groups together all the counters of a specific type (accounting, authentication, probe, etc.), and may or may not offer the level of granularity or accuracy required for troubleshooting the issue
- Trying to look for exact correlations of various counters, even over a monitored period, can be tricky. You are best looking at general patterns, as getting too specific may not be feasible.

Call Setup and Teardown statistics

show session disconnect-reasons

This is the general command to see reasons for call failures (setup and teardown). This command will tell you why calls disconnected or failed to setup. In the list you could look for reasons that could be related to authentication failures, but beware, it may not be obvious. For example, for MIP, failed authentications result in the PDSN sending a MIP RRP with code x43 = Mobile Node Failed Authentication, which should result in the mobile sending a PPP LCP Term-Req, which gets counted as PPP-LCP-Remote-Disconnect in show session disconnect-reasons. For SIP, this is via the PDSN sending a PPP CHAP failure and PPP LCP Term-Req, which gets counted as PPP-Auth-failed in disconnect reasons. The point here is that the disconnect reasons by themselves may not always point to failed radius authentication as the cause.

show mipfa statistics (on FA chassis)

show mipha statistics (on HA chassis)

For MIP calls, run these command(s) to get more detailed information on possible call setup/re-registration/teardown failures. Make sure to be in the contexts where the respective FA and HA services are located when running the commands.

Radius-related areas to look under for "show mipfa statistics" include:

MIP AAA Authentication:
    Attempts – incremented for each set of authentication messages sent with specific id
    Success – incremented for each success response
    Total Failures – includes Failures due to failure response (rejection) or no response
    Actual Auth Failures – incremented for each failure response from Radius
    Misc. Auth Failures – incremented ONCE for each set (each set of messages will have a unique Id field) of requests that had no response. A set will include all the attempted retries to a particular server for a particular call.

Registration Request Received:
    Auth Failed Reg
    * HA Denied Reg
    Init RRQ Auth Failed
    * Init RRQ Denied by HA
    Renew RRQ Auth Failed
    *Renew RRQ Denied by HA
    Dereg RRQ Auth Failed
    * Dereg RRQ Denied by HA
Denied by FA
    MN Auth Failure
    AAA Authenticator
    HA Auth Failure
Denied by HA
    MN Auth Failure

Similar though less detailed information is available with "show mipha statistics":

MIP AAA Authentication – same as mipfa
Registration Request Received:
    Initial Reg Requests:
        * Denied
    Renew Reg Request:
        * Denied
    DeReg Requests:
        * Denied
Registration Reply Sent:
    MN Auth Failure

show ppp statistics

For SIP calls, additional information can be seen by running this command. Radius-related areas include:

Session Failures
    Authentication failures
Session Authentication
    * CHAP auth failure
    * PAP auth failure
    * MSCHAP auth failure
Session Disconnect Reason
    PPP auth failures

* may or may not be Radius-related

Testing communication with a Radius Server

There are a few commands that can be run to test connectivity to a Radius server.

ping

An ICMP Ping will test basic connectivity to see if the AAA server can be reached or not.

```
[source]CSE2# ping 192.168.50.200
PING 192.168.50.200 (192.168.50.200) 56(84) bytes of data.
64 bytes from 192.168.50.200: icmp_seq=1 ttl=64 time=0.411 ms
64 bytes from 192.168.50.200: icmp_seq=2 ttl=64 time=0.350 ms
64 bytes from 192.168.50.200: icmp_seq=3 ttl=64 time=0.353 ms
64 bytes from 192.168.50.200: icmp_seq=4 ttl=64 time=0.321 ms
64 bytes from 192.168.50.200: icmp_seq=5 ttl=64 time=0.354 ms

--- 192.168.50.200 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4000ms
rtt min/avg/max/mdev = 0.321/0.357/0.411/0.037 ms
```

radius test authentication server x.x.x.x port yyyy <user> <password>

radius test accounting server x.x.x.x port yyyy <user>

This command sends a basic authentication request or accounting **start** and **stop** requests and waits for a response. For authentication, you can use any username and password, in which case you will get a reject response, confirming that Radius is working as designed, or you could use a known working username/password, in which case you should get an accept response.

Sidenote: This command uses the aaamgr process running on the SPC/SMC card to process the messages. Normal subscriber radius traffic is handled by aaamgr processes running on the PAC/PSC cards. This applies to the aaa test command discussed in the next section also.

Here is an example output from monitor protocol and running the authentication version of the command on lab chassis:

```
[source]CSE2# radius test authentication server 192.168.50.200 port 1812 test
test

Authentication from authentication server 192.168.50.200, port 1812
Authentication Success: Access-Accept received
Round-trip time for response was 12.3 ms

<<<<OUTBOUND  14:53:49:202 Eventid:23901(6)
RADIUS AUTHENTICATION Tx PDU, from 192.168.50.151:32783 to
192.168.50.200:1812 (58) PDU-dict=starent-vsa1
 Code: 1 (Access-Request)
 Id: 5
 Length: 58
 Authenticator: 56 97 57 9C 51 EF A4 08 20 E1 14 89 40 DE 0B 62
      User-Name = test
      User-Password = 49 B0 92 4D DC 64 49 BA B0 0E 18 36 3F B6 1B 37
      NAS-IP-Address = 192.168.50.151
      NAS-Identifier = source

INBOUND>>>>>  14:53:49:214 Eventid:23900(6)
RADIUS AUTHENTICATION Rx PDU, from 192.168.50.200:1812 to
192.168.50.151:32783 (34) PDU-dict=starent-vsa1
 Code: 2 (Access-Accept)
 Id: 5
 Length: 34
 Authenticator: D7 94 1F 18 CA FE B4 27 17 75 5C 99 9F A8 61 78
      User-Password = testpassword

Live chassis example:

<<<<OUTBOUND  12:45:49:869 Eventid:23901(6)
RADIUS AUTHENTICATION Tx PDU, from 10.209.28.200:33156 to 65.175.85.10:1645
(72) PDU-dict=custom150
 Code: 1 (Access-Request)
 Id: 6
 Length: 72
 Authenticator: 67 C2 2B 3E 29 5E A5 28 2D FB 85 CA 0E 9F A4 17
      User-Name = test
      User-Password = 8D 95 3B 31 99 E2 6A 24 1F 81 13 00 3C 73 BC 53
      NAS-IP-Address = 10.209.28.200
      NAS-Identifier = source
      3GPP2-Session-Term-Capability =
Both_Dynamic_Auth_And_Reg_Revocation_in_MIP


INBOUND>>>>>  12:45:49:968 Eventid:23900(6)
RADIUS AUTHENTICATION Rx PDU, from 65.175.85.10:1645 to 10.209.28.200:33156
(50) PDU-dict=custom150
 Code: 3 (Access-Reject)
 Id: 6
 Length: 50
 Authenticator: 99 2E EC DA ED AD 18 A9 86 D4 93 52 57 4C 2F 84
```

```
      Reply-Message = Invalid username or password
```

As another example, here is a failure due to the wrong secret configured on the Radius server (or PDSN):

```
[source]CSE2# radius test authentication server 192.168.50.200 port 1812 test
test

Response Failure: Bad Authenticator from RADIUS sever, probably mismatched
key
Round-trip time for response was 6.3 ms
```

Note that this would be counted as "Access-Request Response Bad Authenticator Received" in the radius counters.

Here is an example output from running the accounting version of the command. A password is not needed.

```
[source]CSE2# radius test accounting server 192.168.50.200 port 1813 test

RADIUS Start to accounting server 192.168.50.200, port 1813
Accounting Success: response received
Round-trip time for response was 7.9 ms

RADIUS Stop to accounting server 192.168.50.200, port 1813
Accounting Success: response received
Round-trip time for response was 15.4 ms

<<<<OUTBOUND  15:23:14:974 Eventid:24901(6)
RADIUS ACCOUNTING Tx PDU, from 192.168.50.151:32783 to 192.168.50.200:1813
(62) PDU-dict=starent-vsa1
 Code: 4 (Accounting-Request)
 Id: 8
 Length: 62
 Authenticator: DA 0F A8 11 7B FE 4B 1A 56 EB 0D 49 8C 17 BD F6
      User-Name = test
      NAS-IP-Address = 192.168.50.151
      Acct-Status-Type = Start
      Acct-Session-Id = 00000000
      NAS-Identifier = source
      Acct-Session-Time = 0


INBOUND>>>>>  15:23:14:981 Eventid:24900(6)
RADIUS ACCOUNTING Rx PDU, from 192.168.50.200:1813 to 192.168.50.151:32783
(20) PDU-dict=starent-vsa1
 Code: 5 (Accounting-Response)
 Id: 8
 Length: 20
 Authenticator: 05 E2 82 29 45 FC BC D6 6C 48 63 AA 14 9D 47 5B


<<<<OUTBOUND  15:23:14:983 Eventid:24901(6)
```

```
RADIUS ACCOUNTING Tx PDU, from 192.168.50.151:32783 to 192.168.50.200:1813
(62) PDU-dict=starent-vsa1
 Code: 4 (Accounting-Request)
 Id: 9
 Length: 62
 Authenticator: 29 DB F1 0B EC CE 68 DB C7 4D 60 E4 7F A2 D0 3A
      User-Name = test
      NAS-IP-Address = 192.168.50.151
      Acct-Status-Type = Stop
      Acct-Session-Id = 00000000
      NAS-Identifier = source
      Acct-Session-Time = 0


INBOUND>>>>>  15:23:14:998 Eventid:24900(6)
RADIUS ACCOUNTING Rx PDU, from 192.168.50.200:1813 to 192.168.50.151:32783
(20) PDU-dict=starent-vsa1
 Code: 5 (Accounting-Response)
 Id: 9
 Length: 20
 Authenticator: D8 3D EF 67 EA 75 E0 31 A5 31 7F E8 7E 69 73 DC
```

In all of the above example the radius test message was sent using the aaamgr located on the SPC/SMC card. There is a more specific version of the command that can be run in hidden mode that allows specifying a specific aaamgr instance that you suspect an issue with, for example in the following example no response was received for instance 92 but was received for instance 93:

```
[source]CSE2# radius test instance 92 authentication server 65.175.95.10 port
1645 test test

Authentication from authentication server 65.175.95.10, port 1645
Communication Failure: No response received

[source]CSE2# radius test instance 93 authentication server 65.175.95.10 port
1645 test test

Authentication from authentication server 66.175.5.10, port 1645
Authentication Failure: Access-Reject received
Round-trip time for response was 38.0 ms
```

aaa test {authenticate <user> <password> | accounting username <user>}

These are similar commands as radius test authentication/accounting commands, though a bit more information is included in the requests, and the port number is fixed according to the configuration.

```
[source]CSE2# aaa test authenticate test test

Authentication Successful
  User level: unknown(2)
```

```
  User Privs: CLI

<<<<OUTBOUND  15:02:59:055 Eventid:23901(6)
RADIUS AUTHENTICATION Tx PDU, from 192.168.50.151:32783 to
192.168.50.200:1812 (76) PDU-dict=starent-vsa1
 Code: 1 (Access-Request)
 Id: 8
 Length: 76
 Authenticator: 2E 2D 1F A9 19 FD 1A 6A 4E C0 AF 8A 04 C4 77 45
      User-Name = test
      NAS-IP-Address = 192.168.50.151
      NAS-Identifier = source
      Service-Type = Authenticate_Only
      Event-Timestamp = 1235851379
      NAS-Port = 4573834
      User-Password = D9 BF 75 9D BC 9B 45 00 F8 DB A1 6F A1 30 1D 3C


INBOUND>>>>>  15:02:59:059 Eventid:23900(6)
RADIUS AUTHENTICATION Rx PDU, from 192.168.50.200:1812 to
192.168.50.151:32783 (34) PDU-dict=starent-vsa1
 Code: 2 (Access-Accept)
 Id: 8
 Length: 34
 Authenticator: EF A4 11 21 9B 59 DF 50 77 BF 66 59 F9 D7 B7 73
      User-Password = testpassword


[source]CSE2# aaa test accounting username test

Accounting Successful

<<<<OUTBOUND  15:18:38:850 Eventid:24901(6)
RADIUS ACCOUNTING Tx PDU, from 192.168.50.151:32783 to 192.168.50.200:1813
(74) PDU-dict=starent-vsa1
 Code: 4 (Accounting-Request)
 Id: 6
 Length: 74
 Authenticator: 90 87 2F 57 14 5E 08 29 74 6A 16 0C 82 C3 CF 01
      User-Name = test
      NAS-IP-Address = 192.168.50.151
      Acct-Status-Type = Start
      Acct-Session-Id = 00000000
      NAS-Identifier = source
      Service-Type = Authenticate_Only
      Event-Timestamp = 1235852318
      NAS-Port = 4573839

INBOUND>>>>>  15:18:38:857 Eventid:24900(6)
RADIUS ACCOUNTING Rx PDU, from 192.168.50.200:1813 to 192.168.50.151:32783
(20) PDU-dict=starent-vsa1
 Code: 5 (Accounting-Response)
 Id: 6
 Length: 20
 Authenticator: BC 75 DF 03 E1 BB CE 10 7B 70 85 2B 17 32 B0 8B
```

```
<<<<OUTBOUND  15:18:38:862 Eventid:24901(6)
RADIUS ACCOUNTING Tx PDU, from 192.168.50.151:32783 to 192.168.50.200:1813
(80) PDU-dict=starent-vsa1
 Code: 4 (Accounting-Request)
 Id: 7
 Length: 80
 Authenticator: 76 D8 55 DC F4 16 25 D6 6C B1 5A F0 50 60 DF E1
      User-Name = test
      NAS-IP-Address = 192.168.50.151
      Acct-Status-Type = Stop
      Acct-Session-Id = 00000000
      NAS-Identifier = source
      Service-Type = Authenticate_Only
      Event-Timestamp = 1235852318
      Acct-Session-Time = 0
      NAS-Port = 4573839


INBOUND>>>>>  15:18:38:876 Eventid:24900(6)
RADIUS ACCOUNTING Rx PDU, from 192.168.50.200:1813 to 192.168.50.151:32783
(20) PDU-dict=starent-vsa1
 Code: 5 (Accounting-Response)
 Id: 7
 Length: 20
 Authenticator: 47 F2 35 A5 86 07 12 EB 48 BC 49 C4 39 79 25 43
```

Advanced commands

aaamgr processes handling authentication/accounting requests are running on all active PAC/PSCs in the system, and their instance IDs as well as memory and cpu usage can be viewed with:

show task resources facility aaamgr all

Each aaamgr will be paired with and work for an associated sessmgr process (responsible for overall call handling) and located on a different PSC but using the same instance ID. Also in this example output note the special aaamgr instance 231 running on the SPC which does NOT process subscriber requests but it used for radius test commands (see below section for more detail on that) AND for CLI login processing.

```
                    task   cputime        memory     files      sessions
cpu facility      inst used allc   used  alloc used allc  used  allc S status
---------------------- --------- ------------- --------- ------------- -----
1/0 sessmgr        107 1.6% 100% 119.6M 155.0M   26  500    83  6600 I   good
13/1 aaamgr        107 0.3%  94%  30.8M  77.0M   18  500    --    -- -   good
8/0 aaamgr         231 0.1%  30%  11.6M  25.0M   19  500    --    -- -   good
```

These next commands are extremely valuable in advanced troubleshooting for looking at very detailed data on every (or the specified) aaamgr process/instance. If a aaamgr process gets into a

bad state where it is no longer processing requests, you should be able to see this from the output of this command.

show session subsystem facility aaamgr {all | instance <number>}

show session subsystem facility sessmgr {all | instance <number>}

While it is beyond the scope of this article to examine the superfluous output from this command, you should take some time to go through the output to see what kind of valuable information can be gleaned should you need it one day. Like any other troubleshooting, comparing the output from what you believe to be good versus bad aaamgr instances will often reveal obvious differences in the values reported. This could be reflected in the total number of requests, failure/success rate, auth cancelled, etc. Be sure to clear the session subsystem (you can't clear just one instance stats) so as to eliminate any history which could potentially provide a clouded picture of the current state.

Note that sometimes you may first make an observation with regards to some sessmgrs, for example not taking as many calls (used sessions column in show task resources) as the rest of the sessmgrs on the chassis, only to realize that it is the associated aaamgrs where the actual issue lies as evidenced by the output of this command.

There are also some special commands that can only be run in hidden mode:

show npu flow record min-flowid <aaa min flowid> max-flowid <aaa max flowid>

As already discussed, Radius requests are sent and received by individual aaamgrs. Each of these aaamgrs has a flow associated with it in order for authentication RESPONSES being received by the chassis to be routed to the respective aaamgr expecting a response. Each aaamgr will be assigned a flow ID, so in a fully loaded (depends on what types of PSC/PAC cards of course) chassis of 192 sessmgr/aaamgr pairs, this would mean 192 aaamgr flows, one for each aaamgr to handle incoming traffic. Each flow would need to exist in the NPU flow database on each PSC/PAC that has aaamgrs (all of them) and the range of flows for aaamgrs can be retrieved from the flow space database which is a fixed table stored every PSC (slot) (so you could specify any slot when running the command)

```
[local]HA# show npu flow space slot 16
Flow Range Table Information
----------------------------
fi-lkup-tab-sz=115400
  range-type|min-flowid|max-flowid| total-cnt|  facility|  max-inst|max-per-in
     session|         1|  20324352|  20324352|   sessmgr|       384|     52928
         vpn|  20324609|  20326721|      2113|    npumgr|        16|         0
    aaaproxy|  20326722|  20326785|        64| <UNKNOWN>|         0|         0
     ipsgmgr|  20326786|  20327809|      1024| <UNKNOWN>|         0|         0
         aaa|  20327810|  20636609|    308800|    aaamgr|       386|       800
        port|  20636610|  20702273|     65664|    npumgr|        16|         0
```

You could dump the data of all of these flows by dumping the entire range allocated (308,800) for aaamgrs, but ony the IDs that have been assigned flows will be displayed. Specifying the particular slot where the aaamgr is located is important if you are suspecting corruption of the flow database for that aaamgr on the slot where it is located.

show npu flow record min-flowid 20327810 max-flowid 20636609 slot 4 verbose

Here is an example of one of the 192 flows that get outputted from the command, for aaamgr instance 188, on slot 4. Note the following:

da: destination address of all Authentication Response packets for that aaamgr, which actually is common to all aaamgrs, as it will be the NAS IP address from the aaa group
udp dst prt: UDP port used by the particular aaamgr instance
vpnid: refers to context where aaa is configured
dbia=0:3 : aaamgr is located on PSC 4 (3+1):

```
flow meta data for flowid 20477422:
                addr=0x26778d30, f/i=aaamgr(65000)/188, size=64, hash=0x0123079d,
hash-collision=no
                meta-flow-addr=0x48764cc0, npu-xref-flow-addr=0x48764cc0
        ipv4udp-flow binary record @ 0xc9764cc0 (virt):
                0xffffffff8 0x50010002 0x42aea9c0 0x00000000 0x00000000 0x00006970
0x00000000 0x00000000
                0x413875ee 0x00000000 0x10000002 0x00000000 0x000f4240 0x01040e00
0x03000000 0x0301c297
        Decoding ipv4udp-flow[1]: vpnid=2 keylen=5 type=fixed-sz-flow
        Key:
                da         : 66.175.170.192
                udp dst prt: 26992 (0x6970)
        common-portion:
                valid=1, type=0, ifentry=0, pooldrop=0, sessbia=0, flowidx=20477422
                post-decrypt-data/csum-delta=0x00000000
                cp-qos=0, encode=0, proxy-arp=0, valid=1, prio=0, dpti=0
                decypt-data-type=0, ipsec-idx=0, skip-ipv4=0, proxy-flow=0, vpnid=2
                stat-id/out-sa/sa-index=0x00000000
                sess-id/out-da/policy-id=0x000f4240
        ppb:
                strip-data-link-ppb: header-len=14, rtp-info=0
                policy-fwd-ppb: dbia=0:3:1:49815
```

```
     task                             parent
cpu facility      inst    pid pri  facility      inst    pid
---- ------------------------------  -------------------------
4/0 aaamgr         188   2528   0  sessctrl         0   2887
```

A faster way (compared to scrolling through the dump of all aaamgr flow ids as just described) of determining the flow ID for a given aaamgr is via the command:

show radius info instance <instance ID>

```
[source]HA# show radius info instance 188
```

```
Context source:
-----------------------------------------
  AAAMGR instance 188:  cb-list-en: 1 AAA Group:
  -----------------------------------------
  socket number: 420895072
  socket state: ready
  local ip address: 66.175.170.192
  local udp port: 26992
  flow id: 20477422
  use med interface: yes
  VRF context ID: 2
```

And below is a bug where one aamgr instance (92) in the system has its flow missing from ONLY ONE of the PSC's (11) flow database, but existing in the rest.

```
[source]HA# show radius info instance 92
Context source:
-----------------------------------------

  AAAMGR instance 92:  cb-list-en: 1 AAA Group:
  -----------------------------------------
  socket number: 420419808
  socket state: ready
  local ip address: 66.175.170.192
  local udp port: 25456
  flow id: 20400611
  use med interface: yes
  VRF context ID: 2

[source]HA# show npu flow record min-flowid 20400611 max-flowid 20400611 slot
11 verbose
Dumping flowid range [20400611 - 20400611] flowtype: unspecified[-1]
```

Troubleshooting other types of Radius issues

Radius connectivity issues are probably the most urgent types of issues that need addressing, as has been discussed earlier.

But there are many other types of issues that can occur.

Issues may involve a specific subscriber or groups of subscribers, and/or specific attributes being passed back and forth in the requests and responses.

Sometimes the unexpected behavior occurs for a subscriber, and does not match the behavior expected for the attribute being returned by an authentication accept. This could be a bug, or it could be a lack of understanding of what the attribute value is supposed to mean.

Sometimes there are STxx configurables that need to be specified in order to send particular attributes in authentication/accounting messages. For example, to send the mobile country code

and mobile network code in an accounting message, you would need to configure as follows in the global space: “system carrier-id mcc 123 mnc 567”. Or the reverse may be true – particular attributes are being sent when they should not be, or are not expected.

Sometimes unexpected values for attributes are passed in accounting requests, and in some cases more than one value may be legitimately passed because the specification is not completely clear.

Another common point of confusion is that the radius attribute name returned in an Accept Accept does not always match the name used in the STxx configuration. See the AAA Interface Reference Guide for a list of all possible AAA attributes, their definitions, and in which messages they could appear. But unfortunately there is no one document that maps STxx configurables with Radius attributes.

Timing may be an issue as well – if accounting messages are not sent in a timely manner, it may affect subscribers ability to run certain applications.

The aaa configuration has an option for specifying a dictionary to use, via the configurable “radius dictionary <dictionary name>”. This dictionary contains a list of all the possible attributes that may be sent and received in access and accounting messages. There is a standard dictionary, as well as custom dictionaries that have been developed uniquely for each customer. The code necessary to support all dictionaries is already included in the STxx build – no extra files are required for loading.

Most of the issues that involve dictionaries have to do with the PDSN not including an attribute in an access or accounting request when it should, including an attribute when it shouldn’t, or including an unexpected or wrong value for an attribute.

The area of intra-pdsn and inter-pdsn handoffs and the accounting associated with that also is a complex area with lots of possible things that can happen, and is worth studying carefully.

Whenever troubleshooting issues related to values of attributes being assigned to a subscriber and the resultant behavior, you should capture a monitor subscriber verbosity 2, along with “show subscriber aaa-configuration <username | ip-address | …> <id of user per previous argument>”, which is the chassis’s understanding of the subscriber’s radius attributes.

As mentioned earlier, taking capture traces and turning on the appropriate logging may also be needed depending on the issue being troubleshot.

Explanation of typical authentication and accounting attributes

While it is beyond the scope of this article to discuss every attribute and combination of attributes, it may be worthwhile to become familiar with some of the more common attributes and the list of values that can be assigned to these attributes (some are not list-based but rather could be any numerical value, such as number of bytes, timestamp, etc.). Understand that this

will for sure be different for each vendor, so it is impossible to give a definitive list, but many of the ones listed below will be found in most requests, so it is a good starting point.

There are useful tables in the AAA Interface Reference Guide that explain which attributes can be sent/expected in which kind of message. This includes entries for PDSN/FA and HA, requests and responses (further distinguishing rejects vs. accepts), authentication vs. accounting (further distinguishing start, stop, interim)

Authentication (Access) Requests/Responses

PDSN-based requests (SIP and MIP)

- Calling-Station-id – assigned to the mobile by the RAN during call setup.
- User-Name – programmed into the mobile device
- CHAP-Password (and CHAP-Challenge) – hashed items including user password
- NAS-IP-Address – ip address of interface used to send radius messages
- 3GPP2-Service-Option – HSPD (High Speed Packet Data) (1X) = 33; HRPD (High Rate Packet Data) (Rev A) = 59; Wireless-Other
- Service-Type – service attempting to be used: Framed = 2 is for normal subscriber sessions
- Framed-Protocol – protocol to be used: PPP = 1 is only one supported
- SN1-Service-Address – IP address of **PDSN** or HA service handling the call
- SN1-VPN-Name – destination context name
- NAS-Port-Type – speed/type of data connection: Wireless_Other = 18; HRPD = 24
- 3GPP2-Serving-PCF – PCF IP address
- SN1-Service-Type – type of service being accessed: **PDSN = 1**, chassis management = 2, HA = 3, LNS = 5
- 3GPP2-IP-Technology – Simple-IP = 0; Mobile-IP = 1
- NAS-Port – assigned resource number for the session

Additional fields for PDSN-based requests for MIP:

Note: In order to minimize the amount of authentication requests on the FA, there is a configurable in the FA service, "authentication mn-aaa renew-and-dereg-noauth" that allows for disabling authentication for re-registrations and de-registrations.

- 3GPP2-Correlation-Id – correlation id for this leg of the call
- Framed-IP-Address and -Netmask – either an existing address already assigned to the mobile, or 0.0.0.0 to request an address
- 3GPP2-FA-Address – FA service IP address
- 3GPP2-MIP-HA-Address – requesting this IP address for HA service; usually = 255.255.255.255 for new calls

PDSN-based responses (SIP):

- Service-Type, Framed-Protocol
- Framed-IP-Address and Netmask – either ip address and netmask to be assigned for static subscriber, or 255.255.255.254 for dynamic user who will be assigned address out of a public pool
- Framed-MTU – Maximum transmission unit (size) for data packets
- Framed-Compression – the compression type to be attempted to be negotiated during remaining ppp negotiation phase: none = 0; VJ = 1 (VJ_TCP_IP_header_compression); LZS = 3
- Session-Timeout – total allowable life of session on PDSN
- Idle-Timeout
- Acct-Interim-Interval – frequency to send an interim accounting message (seconds)
- Framed-Pool – group pool name to get ip address from

PDSN-based responses (MIP):

- 3GPP2-MIP-HA-Address – HA to send MIP request to
- Session-Timeout, Idle-Timeout, Acct-Interim-Interval – same as for SIP
- Note: no pool information returned as pools are on the HA
- Note: no Framed-Compression, as compression has already been negotiated unlike with SIP

HA-based requests (MIP only obviously):

The same configurable for the FA service for disabling unnecessary authentication as described above exists in the HA service.

- NO Calling-Station-id – the HA is not dealing directly with the mobile device
- NO Framed-Protocol – PPP not done o006E the HA
- NO 3GPP2-FA-Address – No FA service on the HA
- NAS-Identifier – used to identify this HA from others for the purposes of the Radius server initiating message exchange with an HA (as opposed to the HA/PDSN initiating which is the norm)
- SN1-Service-Type – type of service being accessed: PDSN = 1, chassis management = 2, **HA** = 3, LNS = 5
- 3GPP2-MIP-Lifetime – timeout by which mobile needs to re-register MIP RRQ (could be present on PDSN/FA also)
- 3GPP2-MN-HA-SPI – SPI passed to radius to get secret returned – normally specified in the configuration line “mn-ha-spi”

HA-based responses (MIP only obviously):

- 3GPP2-MN-HA-Shared-Key – key must match with the authenticator received in the Mobile Home Authentication Extension in the MIP RRQ

Accounting Requests

Accounting Requests (Acct-Status-Type = Start)

- The following have already been described above:
  User-Name, Calling-Station-id, NAS-IP-Address, 3GPP2-Correlation-Id, 3GPP2-Service-Option, Service-Type, Framed-Protocol, NAS-Port-Type, 3GPP2-Serving-PCF, Framed-MTU, Framed-Compression, 3GPP2-IP-Technology, Framed-IP-Address/-Netmask, SN1-Service-Type, SN1-VPN-Name; MIP only: 3GPP2-MIP-HA-Address, 3GPP2-FA-Address
- Acct-Status-Type – what type of accounting message: Start, Stop, Interim
- Acct-Session-Id – unique alphanumeric string identifying the accounting session
- SN1-Local-IP-Address – For SIP, the destination context IP address
- SN1-Primary-DNS-Server, SN1-Secondary-DNS-Server – primary and secondary DNS servers set during PPP negotiation; may be 0.0.0.0
- 3GPP1-Always-On – Active, Inactive
- 3GPP2-Airlink-Record-Type: Active-Start, Active-Stop
- 3GPP2-Beginning-Session: new accounting session or existing one: True, False
- SN1-IP-Header-Compression: None, …

Accounting Requests (Acct-Status-Type = Stop):

- Similar as Start, except:
- No 3GPP2-Beginning-Session
- 3GPP2-Release-Indicator – reason for the stop (handoff, PPP-termination, etc.)
- SN1-PPP-Progress-Code – last state of call (Call-Simple-IP-Connected, Call-Mobile-IP-Connected, etc.)
- 3GPP2-Session-Continue – whether or not this session will be continued – True or False
- Acct-Session-Time – length of the accounting session (seconds)
- SN1-Disconnect-Reason – (Remote-Disconnect, etc.)
- Acct-Termination-Cause – User_Request
- Many other attributes related to throughput

Accounting Responses: as mentioned earlier, there is nothing significant returned in an accounting response.