INTERLINK NETWORKS

Technical Document

RAD-Series AAA Server Administrator's Guide

Version 9.0

August 2021

Interlink Networks Services, LLC
2531 Jackson Road, Suite 306
Ann Arbor, MI 48103-3818

Main - (734) 821-1200

Sales - (734) 821-1228

Fax - (734) 821-1235

Support - (734) 821-1222

Support E-mail support@interlinknetworks.com

Website - www.interlinknetworks.com

Table of Contents

About this Document

Welcome to RAD-Series Server software. This guide provides information about:

- Server components and operation
- Advanced server configuration, including:
    - EAP authentication
    - External data stores
    - Advanced Policy
- Server and accounting logging and monitoring
- Format and content of RAD-Series Server configuration files

Audience

This *Administrator’s Guide* is for Network and Systems Administrators who must configure and maintain the RAD-Series Server. It assumes that you:

- Are familiar with basic Unix commands
- Know the hardware and software profiles of the server machines and other devices used throughout the network
- Are familiar with LDAP configuration, if implemented
- For wireless networks, know the EAP methods and user name formats used

Notational Conventions

Text in this guide is marked in different styles to denote various things.

Text Marked...	Indicates...
`Fixed-width font`	Code, a command, a file name, or a file parameter. Enter exactly what is shown.
`Fixed-width italic`	A variable. Enter what is correct for your installation, not what is shown.
Normal italic	Title of a book or other publication.
Bold	Something noteworthy, so we emphasize it.
Blue underline	Hypertext link. Click the link to send e-mail or to open the Web page or document in your browser.

Introduction

Overview

See the *RAD-Series Getting Started Guide* for installation and basic configuration instructions.
The Interlink Networks RAD-Series Server software provides Authentication, Authorization, and Accounting services to secure wired or wireless networks. It uses the RADIUS protocol for LAN access control and supports:

- Password Authentication Protocol (PAP)
- Challenge-Handshake Authentication Protocol (CHAP)
- Microsoft® Challenge-Handshake Authentication Protocol (MS-CHAP), v1 and v2
- Extensible Authentication Protocol (EAP)

The RAD-Series Server fully implements the IEEE 802.1X standard for securing wireless LANs using Extensible Authentication Protocol (EAP). EAP authentication may be done using any of several different methods. See page 18 for a list of supported EAP methods.

The RAD-Series Server also supports:

- Proxying authentication and accounting messages to another RADIUS server
- Redundant LDAP user data stores, including load balancing and failover functions
- SNMP integration
- DHCP relay from configured IP address pools
- Token pools to control access to resources
- Advanced Policy to implement:
  - DNIS routing
  - Dynamic Access Control
  - Authorization based on logical groups
  - Customer-defined policies to apply to RADIUS requests

Software Components

RAD-Series Server software delivered to you includes a 64 bit application supporting RADIUS protocol.

RAD-Series AAA Server

RAD-Series AAA Server application is comprised of:

- Finite State Machine (FSM) and some associated routines
- Software modules
- Configuration files

When the RAD-Series Server is started, it loads and initializes the software modules and reads the configuration files.

Finite State Machine

At the core of the RAD-Series Server is the Finite State Machine (FSM) and associated state tables that define all processes for handling RADIUS requests. At application startup, the FSM reads instructions from a state table (by default the radius.fsm text file). The state table outlines what modules to call in response to certain events and in what order to call them.

The RAD-Series Server comes with a default state table capable of, with no modifications, handling standard RADIUS and EAP requests. However, you can extend or customize server function just by modifying the default state table or creating new state tables. For example, you can log interim accounting messages by calling the appropriate predefined module at a certain point in the accounting handling process.

The FSM also provides the flexibility to call custom plug-ins created with the Software Developers Kit (SDK) in lieu of the standard server modules.

Software Modules

RAD-Series Server software modules define the actions that the server performs in response to FSM events--such as authenticating requests, authorizing service, and logging. Built-in actions support various extensions, such as authenticating users with information retrieved from replicated data stores. A software module is also referred to as a “plug-in” or an “AATV”.

To customize the RAD-Series Server with a proprietary software module, you must build and compile your plug-ins using the Software Developers Kit (SDK), which is purchased separately.

Configuration Files

The configuration files store application-specific information used by the software modules to process requests. The configuration parameters can be modified by using a text editor. The files you’ll most commonly work with are:

- `aaa.config` - defines all RAD-Series Server properties.
- `authfile` - defines realm datastores.
- decision files - contain advanced policy information for user authorization and session control based on any logical group that can be defined with attribute-value (A-V) pairs.
- `dictionary` files - defines all attributes and values that may be used to build A-V pairs recognized by the RAD-Series Server. These A-V pairs convey information in requests and responses. These files also contains definitions for all the authentication types that the server recognizes.
- `EAP.authfile` - defines realm authentication actions.
- `las.conf` - enables session tracking and specifies some session timing and management values.
- `log.config` - defines accounting message logging behavior.
- `radius.fsm` - is the Finite State Machine table. You can edit this to reorder RAD-Series Server processing steps or call custom plug-ins.
- realm files - contain user profile entries, including check, deny, and reply items.
- `users` - defines user profiles which can be used for exceptions to the normal realm based configuration.The default users file contains only the test_user entry after an initial installation.
- `vendors` - contains optional entries for vendor names and numbers.

How the RAD-Series Server Works

RADIUS Message Exchange

The RAD-Series Server and its clients communicate through an exchange of RADIUS messages (data packets) that contain information related to a user request. Different types of RADIUS messages are exchanged throughout the AAA process.

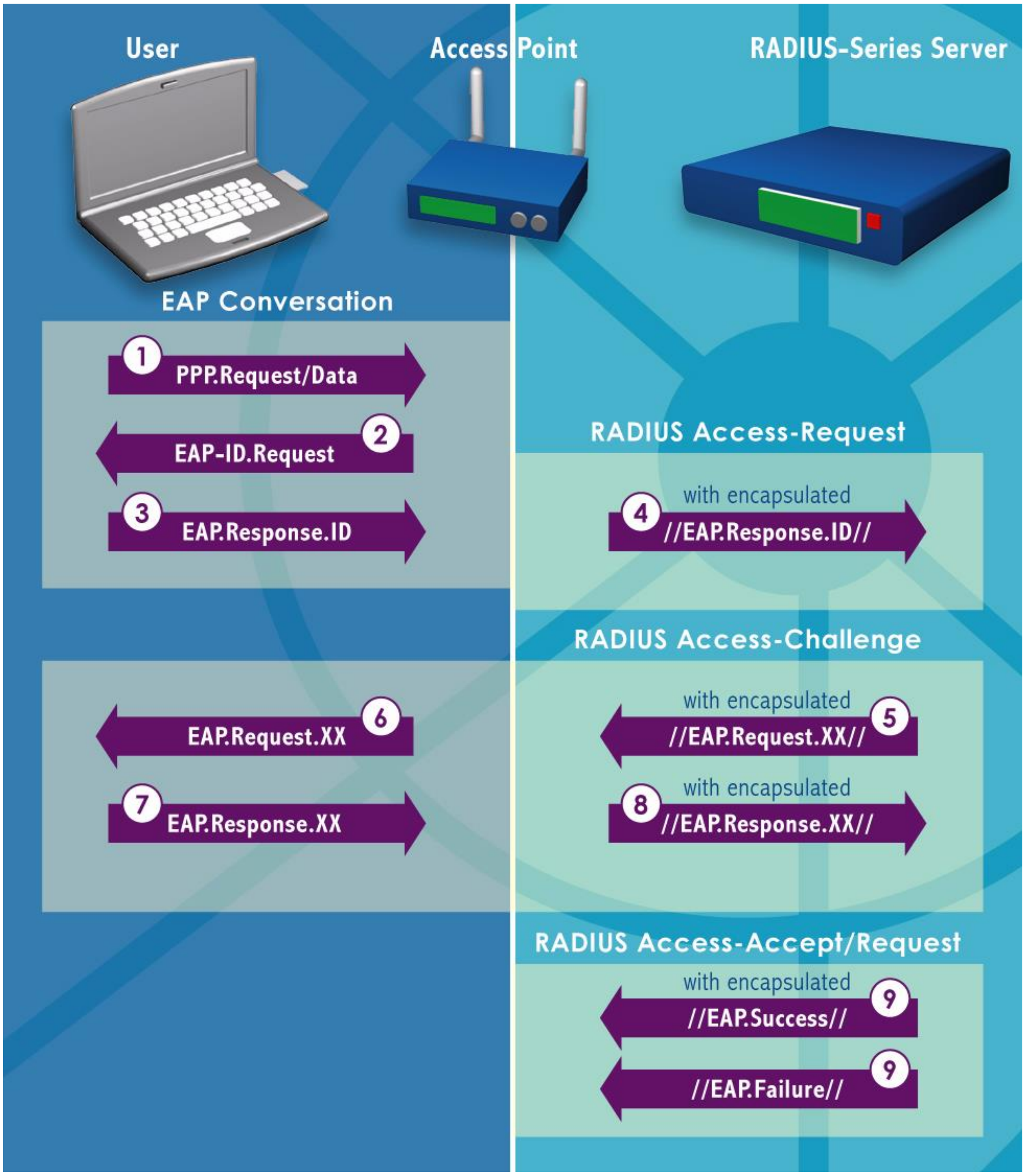
When the server receives a request, it first validates the access device that sent the request, then the user. If the access device is permitted to send requests to the server, the RAD-Series Server software takes information from the **Access-Request** message and attempts to match the request to a realm user profile store and a user profile. The realm policies and user profiles specify conditions that must be met to successfully authenticate and authorize a user. If an EAP authentication method is used, the server and the client exchange a series of **Access-Challenge** messages to verify the user's credentials.

If authentication is successful, the server goes on to authorization. Authorization may verify other information, such as the port number of the access device. If all conditions are met, the server sends an **Access-Accept** packet to the access device. An Access-Accept data packet includes attribute-value pairs that specify the type of user service and other session information, such as a timeout value that indicates when to disconnect the user.

If at any point conditions are not met, the server will send an **Access-Reject** message and the access device will disconnect the user.

When the access device receives an Access-Accept packet, it may send an **Accounting-Request** message to the server to start logging the session. The Accounting-Request data packet describes the type of service being delivered and the user that will use the service. The server responds with an **Accounting-Response** message.

During the session, the access point and the server may exchange Accounting messages that provide "`Interim-Updates`" to verify that the session is still active.



RADIUS message exchange with EAP authentication

RADIUS Message Format

RADIUS requests and replies are transported by UDP. By default, the RAD-Series Server listens on UDP port 1812 for Access-Requests and port 1813 for Accounting-Requests.

EAP-enabled access devices use the EAPOL (EAP Over LAN) standard format for transporting EAP messages within RADIUS messages. EAP requests and responses are encapsulated in the RADIUS `EAP-Message` attribute of the RADIUS messages. EAPOL also provides control functions such as start, logoff, and key distribution.

Password Encryption

The User-Password attribute within RADIUS messages are hidden using the RADIUS MD5 hashing algorithm.

Shared Secrets

RADIUS servers and their clients maintain a trust relationship through the use of a shared secret. A shared secret is a string up to 1023 characters long, with no spaces. The same secret is configured on both the server and the client device. A server may share a different secret with each of its clients.

Attribute-Value Pairs

RADIUS messages are primarily composed of attribute-value (A-V) pairs. A-V pairs represent a variable and one of the possible values that the variable can hold. A-V pairs are exchanged in RADIUS messages to:

Match values when authorizing an Access-Request (check and deny items)

Add provisioning instructions or other messages to an Access-Accept data packet (reply items)

In an RAD-Series Server configuration file, the A-V pairs usually follow the syntax:

`AttributeName=Value`

A-V pairs also appear in the server accounting logs (Merit format) where they follow the syntax:

`AttributeName:Type=Value`

Request Processing

When the RAD-Series Server receives any RADIUS message, it calls the FSM and defines a starting event according to the type of message. This first event will determine the first action. The following diagrams show how the default FSM calls actions to process requests for authentication, authorization, and accounting. This process can be modified by editing the FSM table or by creating custom plug-ins to replace the standard actions.

Authentication

The authentication process verifies that the access device and the user are legitimate.

1 The access device sends the RAD-Series Server an Access-Request containing the User-Name from which the Server get the user ID and realm.

2 The RAD-Series Server checks for the user's profile in the default users file and if found uses it to authenticate the user.

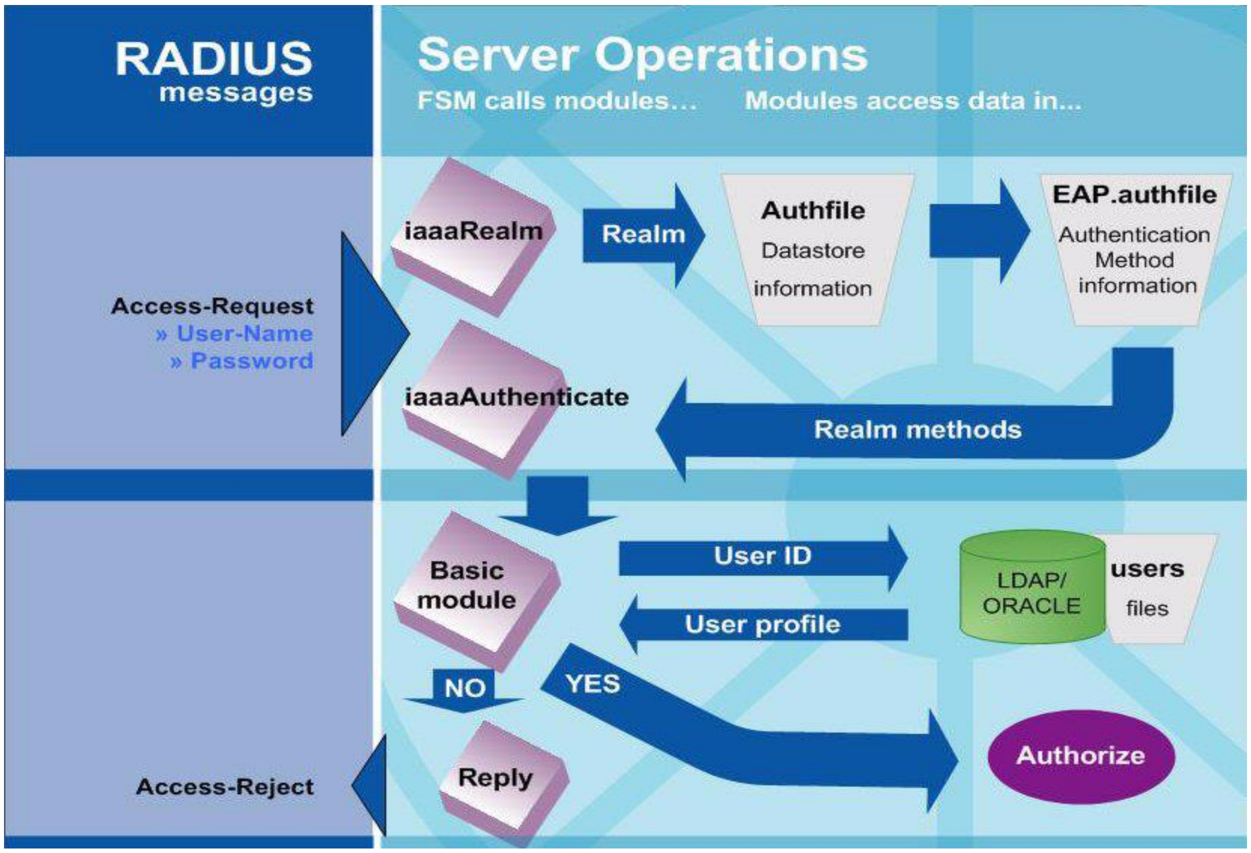
3 If the username is not found in the default users file then the RAD-Series Server finds the realm's user profile storage entry in the `authfile`.

4 The RAD-Series Server authenticates the user according to the protocol established by the realm's authenticator entry in the `EAP.authfile`.

If PAP, CHAP, or MS-CHAP is used, the RAD-Series Server searches the user data store for a matching user profile.

If an EAP method is used, authentication will be carried out according to the EAP method (MD5, PEAP, TLS, TTLS, AKA, SIM etc.).

5 If authentication succeeds, the server proceeds to authorization. If authentication fails, the server sends an Access-Reject message to the access device, who disconnects the user.



BASIC (non-EAP) authentication process

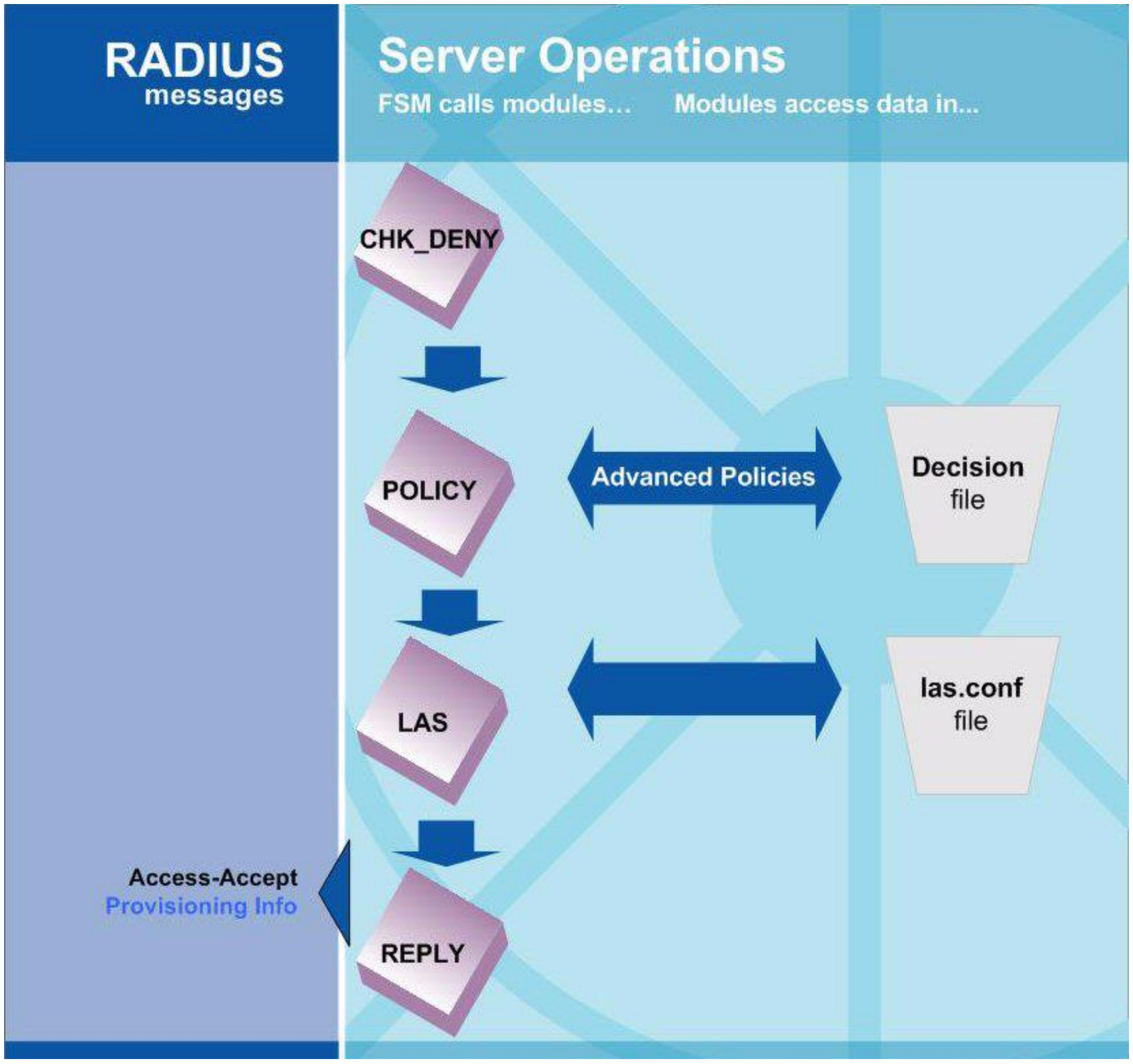
Authorization

The authorization process determines what services can be extended to the user. The RAD-Series Server can authorize users through several methods:

- On a user-by-user basis with check and reply items
- Based on realms through its Local Authorization Server (LAS) functions
- Based on other logical groups through stored POLICY decisions (advanced policy) that make use of more sophisticated checks and conditional reply items.

1. The RAD-Series Server evaluates the request against any check and deny items stored with the user profile. For example: a check item indicating that the request must be from port 1 on the access device must match a corresponding `NAS-Port=1` A-V pair in the request for the request to be accepted.
2. If there is a `Policy-Pointer` attribute from the user or realm profiles, the server evaluates the authorization policy specified by the `Policy-Pointer` attribute.
3. If all conditions are met, the server sends an Access-Accept message back to the access device.

   If any conditions are not met, the server sends an Access-Reject message back to the access device, which disconnects the user.



Authorization flow in default Finite State Machine

Accounting

During operation with session tracking enabled for the realm, the RAD-Series Server tracks information received in Accounting-Requests from the client in an active session table. When the session is stopped, the session record is written to the accounting log file. The predefined accounting logs follow the Interlink-MERIT format. To some degree, you can modify:

- Where and in what format the RAD-Series Server generates the logs by editing the `log.config` file.
- When the logging occurs by editing the FSM table.

1 The access device sends an Accounting-Request to the RAD-Series Server to start recording the session.

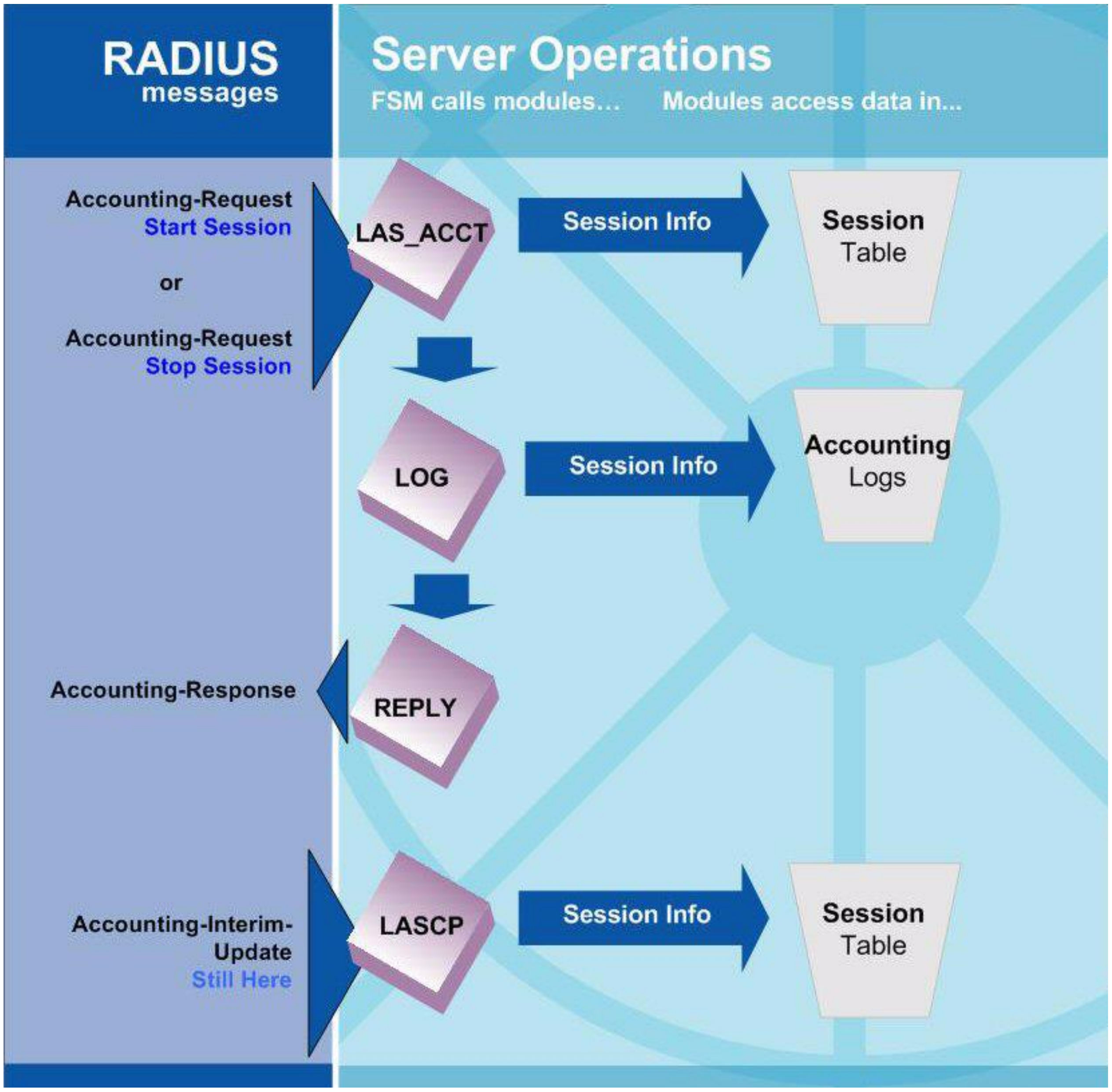
2 The server checks the `log.config` file for the realm's logging format.

3 The server begins tracking the session and sends an Accounting-Response to the access device to confirm this.

4 During the session, the access device and the server may exchange Accounting-Alive (Accounting-Interim-Update) messages to verify that the session is still active.

5 The access device sends an Accounting-Request message to the server to stop recording the session, which the server acknowledges with an Accounting-Response.

6 The server writes the session record to the accounting log.



Accounting Process

WLAN Security

With support for IEEE 802.1x functionality, the RAD-Series Server provides security framework to support EAP authentication mechanisms for Wireless Local Area Network (WLAN) users. The RAD-Series Server:

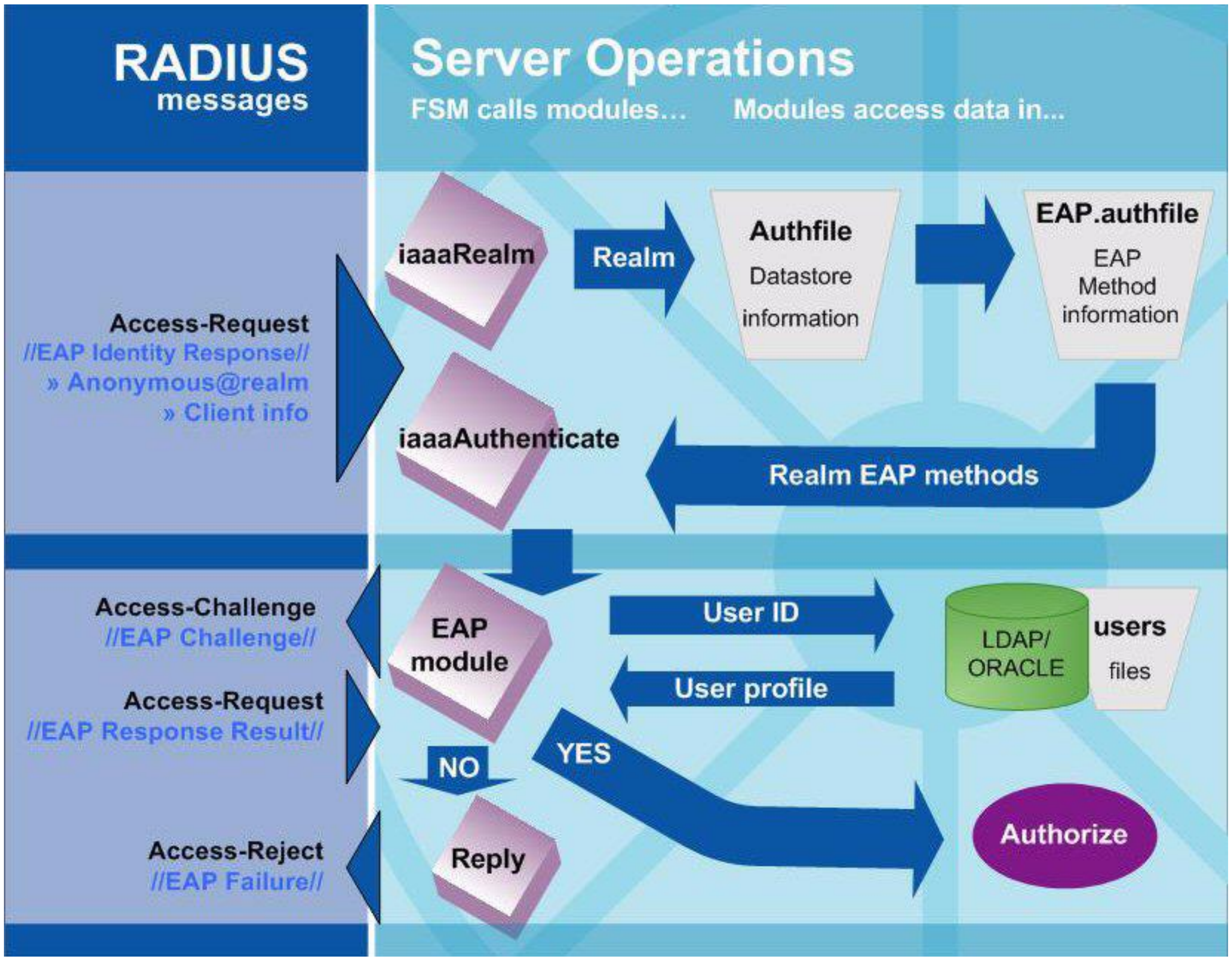
- Authenticates wireless users with password or non-password based mechanisms
- Supports dynamic key generation for data encryption between the access point and wireless stations

EAP Authentication

If the Access-Request message sent to the RAD-Series Server indicates an EAP authentication method for this user, an EAP conversation is encapsulated within the RADIUS exchange that allows mutual authentication to take place directly between the user and server without intervention by the access device. Mutual authentication may be achieved by challenges and responses or exchanging certificates.

1. The access device sends the RAD-Series Server an Access-Request containing the encapsulated EAP Identity Response message.
2. The Server checks for the user profile in the default users file and if found uses it for step 4.
3. If the username is not found in the default users file then the server finds the realm’s user profile storage entry in the `authfile`.
4. The server issues its EAP challenge to the user and verifies the response.
   - The EAP module searches the user data store for a matching user profile.
   - The server sends an Access-Challenge with an encapsulated EAP message to the user via the access point. There may be several such exchanges.
5. If authentication succeeds, the server proceeds to authorization.

   If authentication fails, the server sends an EAP-Failure to the supplicant and an Access-Reject message to the access device, which disconnects the user.



EAP authentication using a single-phase challenge

Tunneled EAP Authentication

The TTLS and PEAP methods establish a tunnel for secure message exchange. In these methods, the authentication process is divided into two phases:

Phase 1: Secure communication is established between the realm user and the RAD-Series Server Server.

Phase 2: Actual user authentication is performed.

Separating these operations also provides flexibility. For example, with TTLS you can proxy the authentication requests to a remote RADIUS server. This enables you to provide wireless access to users whose profiles are stored on a legacy RADIUS server that does not support EAP.

In TTLS and PEAP there is a server certificate but there is no client-side certificate to exchange, making it easier to administer a large number of users.

Tunneled methods often have an outer realm user, which is associated with the tunnel itself. This outer realm and “anonymous” user is configured with its own identity separate from the many inner realm users who are authenticated through the tunnel. In some cases (most PEAP implementations), the “outer” tunnel realm and the inner authentication realms bear the same name and there is no “anonymous” user.

1 The access device sends the server an Access-Request containing the encapsulated EAP Identity Response message. This message usually contains information for the outer realm user.

2 The server finds the tunnel realm’s authenticator entry in the `EAP.authfile`.

3 The server authenticates the outer realm user.

4 If authentication succeeds, the server establishes a tunnel to the user machine.

5 The user sends an Access-Request message through the tunnel containing the actual user information.

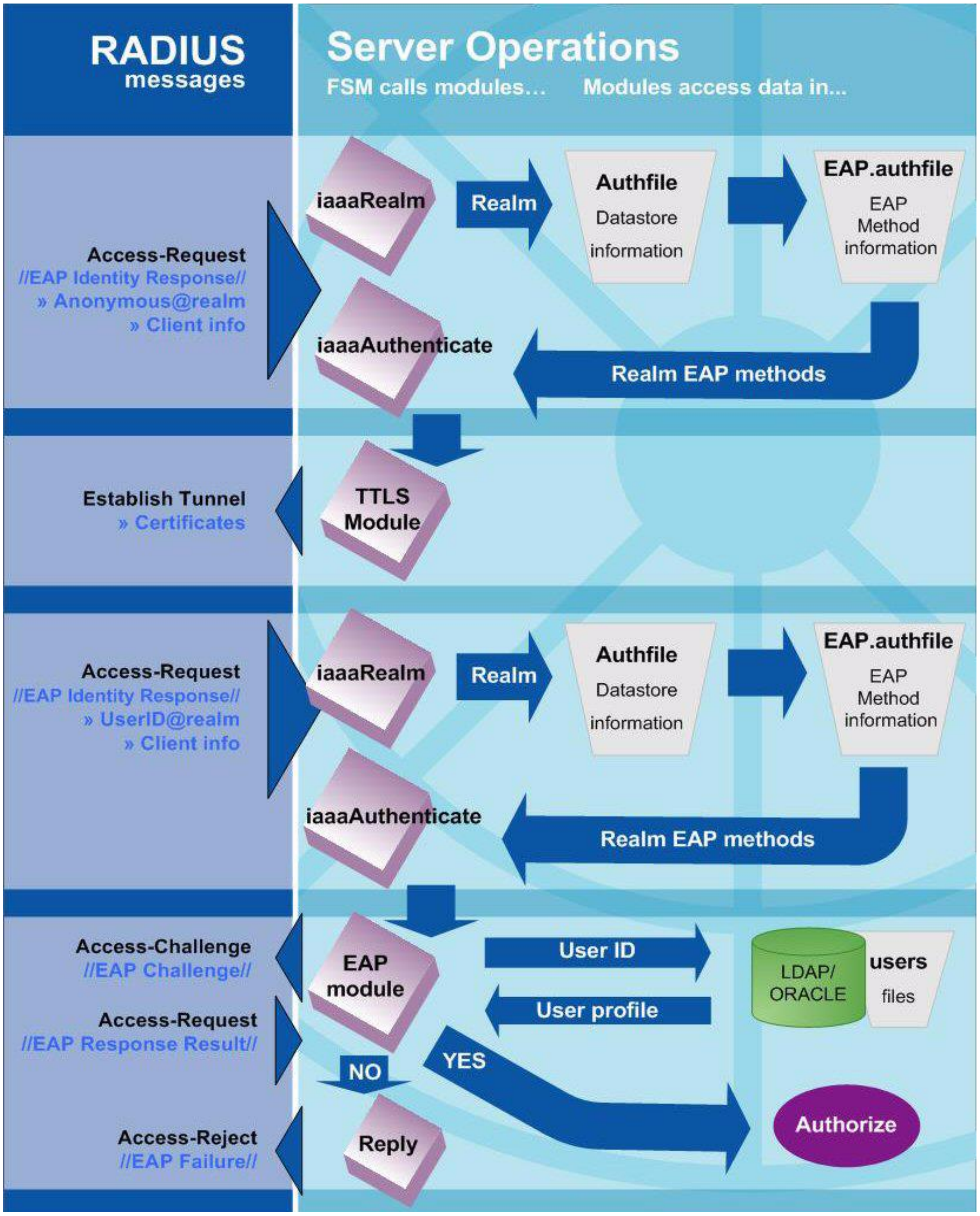
6 The server finds the inner realm’s user profile storage entry in the `authfile`.

7 The server authenticates the actual user. This may utilize a second EAP method, such as EAP- MD5.

   - The EAP module searches the user DataStore for a matching user profile.
   - The server and user exchange Access-Challenges with encapsulated EAP messages through the tunnel.

8 If authentication succeeds, the server proceeds to authorization.

   If authentication fails, the server sends an EAP-Failure to the supplicant and an Access-Reject message to the access point, which disconnects the user.



TTLS authentication

Preparing the WLAN

A WLAN requires you to synchronize items on the supplicant, access point, and the RAD-Series Server. The following table lists the items you need to synchronize on each node.

Item	Nodes	Notes
Shared Secret	Access Device Server	The shared secret configured on the access device and server must match for the two to communicate. Use the Access Devices link to configure this item on the RAD- Series Servers.
EAP Support	Access Device	Most access devices require you to enable EAP. You do not need to specify an EAP method, but you must enable support for EAP.
EAP Method	Client Supplicant Server	Verify that the supplicants support the EAP methods the RAD-Series Server supports. Enable EAP on the supplicants. Configure the same EAP method on the supplicant and the server. Use the Local Realms link to configure this item on servers.
EAP Tunnel Realm	Client Supplicant Server	Required for TTLS. Verify the supplicant has an anonymous user configured on it and configure a tunnel realm for that anonymous user on the RAD-Series Server. For example, if supplicant's anonymous user is: anonymous@tunnel.com, you should configure a realm for: tunnel.com. You must configure tunnel realms for TTLS. Configuring tunnel realms for PEAP is optional. Use the Local Realms link to configure this item on the server.
Users	Server	The RAD-Series Server must have access to a repository with information for each user. The server supports several different methods for retrieving user information.
Client Certificate	Client Supplicant	For TLS only. The digital certificate identifying the client
Client CA Certificate	Server	For TLS only. Used by RAD-Series Server to authenticate client certificates. Use the Server Properties link and select Certificate Path Properties. In the Certificate Authority Path field, configure the location of the client CA certificate on the server.

Item	Nodes	Notes
Server Certificate	Server	For TLS, TTLS, and PEAP only. The digital certificate identifying the RAD-Series Server. Use the Server Properties link and select Certificate Path Properties. In the Certificate Path field, configure the location of the client CA certificate on the server.
Server CA Certificate	Client Supplicant	For TLS, TTLS, and PEAP only. Used by clients to authenticate the server certificate.

Choosing an EAP Method

Choose EAP methods based on your security requirements and the clients you support.

First, create an inventory of the wireless clients you support. Wireless clients need specific supplicant software for each EAP method (WLAN access devices must only support EAP). You must use supplicants that support the hardware platforms, operating systems, and WLAN cards in your environment. Ideally, you should try to use client hardware and software that allows you to use one EAP method for all your wireless clients. This may mean avoiding solutions that are proprietary or support only a small variety of clients.

Next, determine which of the following features are important to you:

1. Dynamic Key Exchange—Distributes a user-specific encryption key to the client and access device during the authentication process. Without this feature, all clients must share the same static encryption key.
2. Mutual Authentication—Protects against unauthorized (rogue) access points by allowing clients to authenticate the network they are connecting to.
3. Password-based Authentication—Clients provide a password to authenticate to the network. Typically the password is sent to the server in a hashed (one-way encrypted) form. If you are integrating with an existing password storage format, be sure the EAP method you chose is compatible with the password storage format. For the most flexibility, choose an EAP method that allows the RAD-Series Server to access the password in clear text (for example, the PAP password format). Storing passwords in clear text requires you to use EAP methods that encrypt the channel between the client and the access point (like TTLS or PEAP).
4. Digital Certificate/Token Card-based Authentication—Uses a token card, smart card, or digital certificate assigned to each user for authentication. This feature must be deployed in an environment with supporting infrastructure—for example, an organization with a PKI and user-specific certificates.
5. Encrypted Tunnel—Establishes an encrypted channel to securely deliver authentication messages and encryption keys. The encrypted tunnel encapsulates another EAP method that provides the actual user authentication. Encrypted tunnels are good for securing authentication methods that are vulnerable when not encapsulated in an encrypted tunnel.

Supported EAP Methods

The following table lists the EAP methods the RAD-Series Server supports and which of the above features each method offers. Use the table and your inventory information to help decide which EAP method to use.

EAP Method	Feature	Description
TTLS • PAP • CHAP • MS-CHAP • MS-CHAPv2 • EAP-MD5	1, 2, 3, 5	**Tunneled TLS**: Can carry additional EAP or legacy authentication methods, like PAP and CHAP. Integrates with the widest variety of password storage formats and existing password-based authentication systems. Supplicants available for a large number of clients. May use PAP, CHAP, MS-CHAP, or EAP-MD5 for inner realm authentication.
PEAP, v0 and v1 • EAP-MD5 • MS-CHAPv2 • EAP-GTC	1, 2, 5	**Protected EAP**: Functionally very similar to TTLS, but does not encapsulate legacy authentication methods. May use EAP-MD5, MS-CHAPv2, or EAP-GTC for inner realm authentication.
TLS	1, 2, 4, 5	**Transport Layer Security**: Uses TLS (also known as SSL) to authenticate the client using its digital certificate. Note: some supplicants require specific extensions to support certificates for EAP.
EAP-MD5	3	**Message Digest 5**: Passwords are hashed using the MD5 algorithm. Can be deployed for protecting access to LAN switches where the authentication traffic will not be transmitted over airwaves. Can also be safely deployed for wireless authentication inside EAP tunnel methods (see feature 5 above).
EAP-GTC	4	**Generic Token Card**: Carries user specific token cards for authentication.
EAP-AKA	1, 2, 4	**Authentication and Key Agreement**: Authentication is based on the use of a USIM or (R)UIM cards.
EAP-SIM	1, 2, 4	**Subscriber Identity Module**: Authentication is based on the use of a SIM card.

Note: If you are using TLS, TTLS, or PEAP, be sure you configure the required digital certificates after you configure all your realms.

Process for Securing WLANs

Below is the general process for using the RAD-Series Server to secure your WLAN. See the following sections of this guide for procedures explaining each of the steps.

1. Identify the access devices that will send access requests to the RAD-Series Server. See “Defining Access Devices” on page 27.
2. Configure tunnel realms if you are using TTLS. See “Defining Realms” on page 34.
3. Configure user authentication realms and user profile data stores. See “Defining Realms” on page 34.
4. Configure user profiles to identify each user accessing services through the RAD-Series Server.
5. If you are using local realm files or the default users file, see “Defining Users” on page 48.
6. If you are using an LDAP directory, see “Using an LDAP Server” on page 104.
7. Configure digital certificates if you are using TLS, TTLS, or PEAP. See “Administering Digital Certificates” on page 44.
8. Deploy the configuration by stopping and starting the RAD-Series Server program. See “Server Administration” on page 20

Server Administration

This section shows how to issue commands to:

- Start and stop the RAD-Series Server
- Change the server's startup options
- Restart the server after changing configurations, without stopping the service
- Report on server status

Starting the Server

To start one or more RAD-Series Servers:

1. Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).
2. Run `radiusd.sh start`

If the process fails to start:

1. Check the `lofgile` (in `/var/opt/aaa/logs` by default).

   If you see the message "(E): setupsock: Terminating. Could not bind socket. Interface(0.0.0.0) port(1813) errno(98='Address already in use')" the default port 1813 is already being used by another process.
2. To identify the process currently using port 1813, at your shell prompt, enter:

   ```
   lsof -i :1813
   ```
3. Either change the Server ports, or kill the process using the ports.

Note: For other error messages, see "Error Messages" on page 101

Stopping the Server

To stop one or more RAD-Series Servers:

1. Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).
2. Run `radiusd.sh stop`

Restarting or Reloading the Server

The reload option causes selected RAD-Series Server to reload a new configuration without first being stopped and restarted. This option sends a HUP signal to the radiusd process. This allows the server to begin processing request again faster than a restart, which first stops radiusd and then starts it up agian.

1 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

2 Run `radiusd.sh reload`
`Or`
Run `radiusd.sh restart`

You must use the `restart` option to stop and start the RAD-Series Server to apply all server startup options and the following server properties:

- Server Certificate Path
- Certificate Revocation List Path
- Server Private Key Path
- Client Certificate Authority Path
- Random Seed Path
- Hold Replies

You must use the `restart` option to stop and start the RAD-Series Server to re-read the following configuration files:

- dictionary files
- vendors
- radius.fsm
- log.config

Debug Level

The debug level specifies the amount of detail written to the `radius.debug` file. You can set the debug level to a value between 0 and 4. Each level includes the information in the levels before it. Higher levels write more information to the debug file, slowing performance. Debugging is intended for limited time use for testing purposes.

Level	Description
0	No debugging (default)
1	Brief trace
2	High-level FSM output, some function tracing, A-V pairs, etc.
3	Full function tracing
4	Low-level FSM and configuration file output

1 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

2 Run `radiusd.sh debug <delta>`

   `<delta>` is the number of debug levels to increment from the current level (not the number of the level you want). The debug level will not increase past level 4. "`off`" or 0 turns debugging off.

Setting Server Start Options

To change various server statrup option you will have to use a text editor. In a text editor, open the file `radiusSetup.sh` (in `/opt/aaa/bin` by default).

UDP Ports

You can change the default port the RAD-Series Server uses to listen for authentication or accounting requests from the UDP standard 1812 and 1813.

Note: The listen ports configured in any `radius_socket()` blocks override these default values.

The authentication and accounting relay ports are used when forwarding requests to machines using different port numbers than the standard 1812 and 1813.

1 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

2 Locate the following lines in `radiusSetup.sh`:

```
radAuthPort="-p 1812"
radAcctPort="-q 1813"
radAuthRelayPort="-pp 1812"
radAcctRelayPort="-qq 1813"
```

3 Change "1812" and "1813" to the ports you wish to use.

4 Save and close `radiusSetup.sh.`

5 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

6 Stop and restart the AAA Server by running `radiusd.sh restart`

Enable Errorlog

When enabled, the server will log messages into the errorlog file. When disabled, the server will not log messages into the errorlog file. The messages selected for inclusion in the errorlog file are those as specified by the `errorlog-level` parameter of the "General Server Properties" on page 171. The server appends to the errorlog file, and rolls over the errorlog file based on file size, per the `maximum-errorlog-file-size` parameter of the "General Server Properties" on page 171. Defaults to Enabled.

Reset Logfile and Reset Session Table

You can set the RAD-Series Server's log file to be renamed to a file with the extension `.`*`yyyymmdd`*`.old` and to start with an empty log file each time the server is started. The debug file is always cleared on each startup. The server can also be configured to ignore reading the session table file (`session.las`) when the server is started

---

Note: Resetting the session table is only recommended for testing environments.

To Change Server Start Options

1. Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).
2. Locate the following line in `radiusSetup.sh`:

   ```
   radOptions=""
   ```
3. Change the contents to "-z" to reset the logfile on every startup. This is only recommended for testing environments.
4. Change the contents to "-n" to reset the session table on every startup. This is only recommended for testing environments.
5. Save and close `radiusSetup.sh.`
6. Stop and restart the AAA Server using the command: `./radiusd.sh restart`.

Reporting Server Status

Use the radcheck program to check the operational status of the selected RAD-Series Server.

Note: To see extended output, a localhost entry in the server’s Access Devices or Proxies list is required.

1. Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).
2. Run `radcheck.sh localhost`

The extended status message contains this additional information

Example Message Lines	Description
Current-Time(2016-11-17/11:00:50) Start-Time(2016-11-17/10:59:36) Uptime(00:01:14)	The current time, server start time, and server uptime
Version 8.5.0 (Linux), Debug-Level(0), errorlog(enabled), License-End-Date(permanent license)	The servers's version, OS type, current debug level, errorlog state, and license end date.
Process-ID(19354) inetd-startup(no) #HUPs(1) Last-HUP-Time(2016-11-17/11:04:45)	The server's process-id, inetd startup indication, and number of HUPs the server has processed since startup.
Auth Requests: Cur(1)/HWM(1)/Max(40000) rcvReq(11) xmtAcc(8) xmtChal(0) xmtRej(2) rcvDup(0) rcvBad(0)	RADIUS authentication request statistics: queue current size, high water mark, and maximum; received requests, transmitted accepts, transmitted challenges, transmitted rejects, duplicate requests, bad requests.
Acct Requests: Cur(0)/HWM(0)/Max(40000) rcvReq(10) xmtRsp(10) rcvDup(0) rcvBad(0)	RADIUS accounting request statistics: queue current size, high water mark, and maximum; received requests, transmitted responses, duplicate requests, bad requests.
Sessions: States: Active(3) Finished(4). Sublists: Timer4(4/5) Total(7/7).	Session tracking statistics. Will report 0 for all statuses unless Session Tracking is enabled for some realm. Session statistics: Number of sessions in each state, and number of sessions on each session timer list.
Sessions: Total(7)/TotalHWM(7). Licensed(3)/ LicensedHWM(3)/LicensedMax(5000).	Current number of sessions and high water mark, following by licensed session statistics: current number, high water mark, and license limit.

Example Message Lines	Description
authfile(x), clients(x), users(x), fsmid[default-8.3-0] dictid[8.3-0] vendid[8.3-0]	Record counts for the authfile, clients, and users configuration files; followed by version information for the finite state table, dictionary and vendors files. You can modify this information by changing the lines like: *%FileID Version-String* Where *FileID* is: • FSMID in a state table (.fsm) • DICTID in the dictionary file • VENDORSID in the vendors file *Version-String* is the version identifier you wish to assign to that file.
"t25(1812)" is responding	A success message. If radcheck fails, one of the following messages will appear: • No reply from server *Name (UDP-port*) • No such server: *Name*
Exit Codes	0 - Successful completion -2 or 254 - Remote server had errors -1 or 255 - Local errors 1 - Timeout errors

Changing radcheck Command Options

To make a temporary change to the command options, just add them to the run command.

```
Radcheck.sh -r 1 -x localhost
```

To change the values used in the radcheck command permenantly:

1. Locate the following line in `radcheck.sh`:
2. `OPT="-d $radConfig_path $radAuthPort"`
3. Add "`-r` *`<count>`*" to change the number of retries.
4. Add "`-t` *`<seconds>`*" to change the timeout value in seconds.
5. To change the port number, replace "`$radAuthPort`" with "`-p <port>`" if you need to change the port to match the port used by a different RADIUS server.
6. Save and close `radcheck.sh.`

Edit Configuration

This section contains instructions on configuring the RAD-Series Server’s:

- Access devices — client devices sending Access-Requests and Accounting Requests to the server
- Proxies — other servers to which requests are forwarded and from which responses are received
- Authentication realms — user realms authenticated by this server
- User data stores — files, directories, and databases from which the server will retrieve user profiles

There are also procedures for setting up:

- The DHCPv4 Relay
- SNMP
- Server Properties

See “Using External Data Stores” on page 104 to configure the RAD-Series Server for use with an LDAP directory or other external data store.

Defining Access Devices

An access device is any other network device—Network Access Server (NAS) or wireless Access Point (AP)— from which the RAD-Series Server will receive RADIUS service requests (aka its clients). It does not include servers we will proxy to or servers we will receive proxied requests from, which are configured separately under Proxies. The server configuration must include an entry for all the access devices that will communicate with the server.

To do this you will have to modify the server’s `clients` file. See “Clients Entry Syntax” on page 224 for more detailed information about the clients file format and options. See “Defining Proxies” on page 30 for instructions on defining proxy servers.

Using Wildcards for IP Addresses

When specifying a client device name, you can use the Classless Inter-Domain Routing (CIDR) subnet notation to specify a set of contiguous IPv4 or IPv6 addresses. You can optionally use the *-style wildcard to replace any portion of the IPv4 address. For example, if a device name `15.0.0.0/8` is entered, then all clients in the ‘15.’ subnet can access the RAD-Series Server. If a device name `2001:3::/64` is entered, then any client with an IPv6 address in the range 2001:3::0 through 2001:3::ffff:ffff:ffff:ffff can access the RAD-Series Server. If a device name of `::/0` is entered, then any IPv6 client knowing the shared secret can access the RAD-Series Server or if device name of `*` or `0.0.0.0/0` is entered, then any IPv4 client knowing the shared secret can access the RAD-Series Server.

The following wildcard patterns are allowed, where `h` is a hexadecimal digit and `w, x, y`, or `z` are decimal numbers less than or equal to 255:

```
0::/0,     0000:0000::/0,   0000:0000:0000::/0,  0000:0000:0000::0000/0,
hhhh::/16, hhhh:0000::/16,  hhhh:0000:0000::/16, hhhh:0000::0000:0000/16
hhhh:hhhh::/32, hhhh:hhhh:0000::/32, hhhh:hhhh::0:0/32
hhhh:hhhh:hhhh::/48, hhhh:hhhh:hhhh::0000/48
0.0.0.0/0,        *,           *.*,         *.*.*,      *.*.*.*
w.0.0.0/8,        w.*,         w.*.*,       w.*.*.*
w.x.0.0/16,       w.x.*,       w.x.*.*
w.x.y.0/24,       w.x.y.*
```

All of the entries on each of the above lines represent the same wildcard pattern, since they will match the same set of IP addresses.

No non-zero bits may follow the wildcard specification. The following are examples of invalid wildcards:

```
2003:24::/16
2003::1/16
15.24.*.42
```

The more specific entries are given more precedence, when matching the source IP address of a request to a client entry. For example: 15.12.0.0/16 will have more precedence than 15.0.0.0/8 when matching a source IP address of, say, 15.12.1.2. The entries are internally arranged to make this precedence effective.

Adding an Access Device

1. Change directory to `/<server configuration path>` (`/etc/opt/aaa/` by default).
2. In a text editor, add a line for each Access Device to the `clients` file of the following form:

   Name Secret **type=*Vendor*:NAS**[*Flags*] *Options* [*Optional-Fields*]

3. For the *Name* enter the device's fully-qualified domain name, IP address, a wildcard IP address as described above.
4. The shared secret, *Secret*, is the text that will be used to establish a trust relationship between the access device and the server. The shared secret can't exceed 1023 characters or contain spaces.
5. The *Vendor* field specifies which set of vendor-specific attributes to return in RADIUS message sent to this device. Use a "+" to specify multiple vendors. Usually, it is sufficient to select the hardware vendor. Use "`none`" if no vendor specific attributes should be sent to the NAS.

6. The `Flags` section is optional and can contain one or more of the Flags below. Unless you have special requirements, you probably do not need any options.

Flags	Description
+Debug	Dump packets into the server's debug output file. To use this option, you must set the server's debug level to > 0 in Server Start Options.
+NoEncaps	Do not encapsulate vendor response.This option is useful if the client requires nonencapsulated A-V pairs.
+OldCHAP	Use pre-RFC CHAP with this client.

7. Save and close the `clients` file.
8. Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).
9. Stop and restart the AAA Server by running `radiusd.sh restart`

Defining Proxies

To proxy authentication or accounting requests with the RAD-Series Server, identify both:

- Proxy servers that this server may receive requests from
- Remote servers that this server may proxy requests to
- Which realms will proxy to which remote servers

Self-referring Client Entry (localhost)

If the RAD-Series Server has a self-referring client entry named **localhost**, the server Status program (`radcheck`) will return extended information, such as authentication and accounting request statistics. This entry is automatically made in the Proxies list when the server is installed. Deleting it will change the Status command output to a single line, "*host* is|is not responding," but won't alter server functioning. See "Reporting Server Status" on page 25 for a description of the extended output.

Receiving Requests from a Proxy Server

To configure proxy servers from which the RAD-Series Server will receive authentication requests:

1. Change directory to `/<server configuration path>` (`/etc/opt/aaa/` by default).
2. In a text editor, add a line for each remote server to the `clients` file of the following form:

   ***Name Secret* type=*Vendor*:proxy**[*Flags*]   *Options* [*Optional-Fields*]
3. For the *Name* enter the remote server's fully-qualified domain name, IP address or a wildcard IP address as described above.
4. The shared secret, *Secret*, is the text that will be used to establish a trust relationship between the remote server and the RAD-Series AAA server. The shared secret can't exceed 1023 characters or contain spaces.
5. The *Vendor* field specifies which set of vendor-specific attributes to return in RADIUS message sent to this remote server. Use a "+" to specify multiple vendors. Usually, it is sufficient to select the hardware vendor. Use "`none`" if no vendor specific attributes should be sent to the remote server.

6 The `Flags` section is optional and can contain one or more of the Flags below. Unless you have special requirements, you probably do not need any options.

Option	Description
+Debug	Dump packets into the server's debug output file. To use this option, you must set the server's debug level > 0 in Server Start Options.
+No Append	If checked, the proxy server expects that all attributes which were sent in a transmitted Access-Request will be echoed in the received response. If unchecked the proxy server expects that all new attributes added to the response will follow the proxy server's Proxy-State attribute.
+Prune	Forces pruning as if the response were being returned to an access device. Using Prune with the Generic vendor option prunes all vendor-specific attributes before a message is returned to the proxy server.

7 Save and close the `clients` file.

8 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

9 Stop and restart the AAA Server by running `radiusd.sh restart`

Proxying Requests to a Remote Server

To configure RAD-Series Server as a proxy to a remote server:

1 Change directory to `/<server configuration path>` (`/etc/opt/aaa/` by default).

2 In a text editor, add a line for each remote server to the `clients` file of the following form:

```
Name:[AuthPort:AcctPort] Secret type=Vendor:PROXY[Options] [srcip=[SrcIP]:SrcPort]]
```

3 For the `Name` enter the remote server's fully-qualified domain name or IP address.

4 To send authentication requests to a port other than the RAD-Series Server's default relay port, enter the port number in *AuthPort*.

This number overrides any relay port you set up under the RAD-Series Server's start options, the content of `/etc/services`, or the RAD-Series Server's default of 1812.

Note: The current RADIUS default ports are 1812 and 1813. Older RADIUS servers may listen for requests on ports 1645 and 1646.

5 To send accounting requests to a port other than the RAD-Series Server's default relay port, enter the port number in `AcctPort`.

This number overrides any relay port you set up under the RAD-Series Server's start options, the content of `/etc/services`, or the RAD-Series Server's default of 1813.

6 The shared secret, `Secret`, is the text that will be used to establish a trust relationship between the remote server and the RAD-Series AAA server. The shared secret can't exceed

1023 characters or contain spaces.

7 The *`Vendor`* field determines which vendor-specific attributes are to be forwarded to the remote server in request messages. Generally you would use "`none`", but you can choose a set that applies if there are special requirements. Use a "+" to specify multiple vendors.

8 The *`Options`* section is optional and can contain one or more of the options below. Unless you have special requirements, you probably do not need any options.

Option	Description
+Debug	Dump packets into the server's debug output file. To use this option, you must set the server's debug level > 0 in Server Start Options.
+No Append	If checked, the proxy server expects that all attributes which were sent in a transmitted Access-Request will be echoed in the received response. If unchecked the proxy server expects that all new attributes added to the response will follow the proxy server's Proxy-State attribute.
+Prune	Forces pruning as if the response were being returned to an access device. Using Prune with the Generic vendor option prunes all vendor-specific attributes before a message is returned to the proxy server.

9 To forward requests from a specified source IP address, add the `srcip=` option and enter the IP address in the *`SrcIP`* field. This value will override the Default Local IP Address for Proxy Socket parameter.

10 To forward requests from a specified port, add the `srcip=` option and enter the port number in the *`SrcPort`* field. If not specified, the server will choose an ephemeral port number.

11 Save and close the `clients` file.

12 Change directory to `/<`*`server utilities path`*`>` (`/opt/aaa/utl` by default).

13 Stop and restart the AAA Server by running `radiusd.sh restart`

Using Wildcards for Realms

When specifying a proxy realm name, you can use the wildcard syntax, `*.realm`.

This syntax provides a shorthand for handling several realms the same way. For example, a company may have several branches, `eastern.company.com, western.company.com`, and `central.company.com`. Using the wildcard `*.company.com`, all three realms would be forwarded to the same server.

It does not matter in what order you enter regular or wildcard realm names in the `authfile`. When the RAD-Series Server reads the realms it sorts the entries so that:

- Non-wildcard entries come first
- Wildcard entries of length *n* come before wildcard entries of < *n*.

This ensures that wildcard entries are used only when there is no exact match among the regular entries and that when there are several wildcards that may apply, the "best-fit" entry is selected.

Defining Realms which will Proxy to Remote Server

1 Change directory to `/<server configuration path>` (`/etc/opt/aaa/` by default).

2 In a text editor, add a line for each realm to be proxied to the `authfile` of the following form:

```
RealmName RADIUS RemoteServer
```

3 For the *`RealmName`* enter the realm name used in incoming requests that should be forwarded to the remote server. Realm names match the domain component of the authentication string input by the user. For example, defining a proxy realm named example.com will cause requests for user@example.com (in NAI format) to be forwarded to the specified RADIUS server.

4 For the *`RemoteServer`* enter the remote server's fully-qualified domain name or IP address.

Proxying Accounting Requests to a Central Server

By modifying the Finite State Machine table (`.fsm`), you can forward all received accounting messages to a central server. This configuration will disable all local accounting for all realms normally handled by the RAD-Series Server.

To forward accounting for a single realm, follow the steps in "Proxying Requests to a Remote Server" on page 31.

To proxy all accounting:

1 Change directory to `/<server configuration path>` (`/etc/opt/aaa/` by default).

2 Complete the steps to add the server to the `clients` file as a Proxy. See "Proxying Requests to a Remote Server" on page 31.

3 In a text editor, open radius.fsm and locate the lines:

```
AcctWait:
   *.*.ACK       ACCT_SWITCH   AcctLog
   *.*.ACCT_DUP  ACK           ReplyHold
```

4 Replace them with:

```
ACCTwait:
   *.*.ACK       RAD2RAD   ReplyHold    Xstring="central.server"
   *.*.ACCT_DUP  RAD2RAD   ReplyHold    Xstring="central.server"
```

Where "`central.server`" is the fully-qualified domain name or IP address of the central accounting server which must be found in the `clients` file.

5 To forward the accounting to a port other than the RAD-Series Server's default, modify the `clients` file for that port, see "Proxying Requests to a Remote Server" on page 31.

6 Save and close radius.fsm.

7 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

8 Stop and restart the AAA Server by running `radiusd.sh restart`

Defining Realms

Generally, a realm is a group of users who share a common characteristic, such as being employees of the same company, division, department, or subscribers to the same internet service. All users of a given realm are:

- Stored in the same location
- Authenticated by the same protocol
- Identified by their `User-Name` attribute value (e.g.: userid@realm)

Define a **separate** realm for each group of users who:

- Use different authentication protocols
- Use different domain names
- Are stored in different locations (different data stores or files)

You may also need multiple realms if using two-phase EAP methods, such as TTLS. For more information on configuring RAD-Series Servers for WLAN and 802.1X environments see "Configuring TTLS Authentication" on page 36, "Configuring PEAP Authentication" on page 38, and "Configuring TLS Authentication" on page 41.

These procedures require you to modify the RAD-Series Server's `authfile` and `EAP.authfile` using a text editor.

About Realms

Realm vs. Domain

A domain refers to a fully-qualified domain name registered with DNS. The realm name may or may not be a domain name. For example, the NULL realm exists to accommodate users who are not associated with any domain name, but who still require authentication.

In general, we recommend that if users in a realm are required to supply a domain name upon login, the domain name be used as the realm name—for example, yourcompany.com. Similarly, any flat file used to store user profiles for this realm should also bear the realm name.

NULL Realm

The NULL realm is defined in the default RAD-Series Server configuration. This is beneficial to those people who manually edit the las.conf file to add the NULL realm and store the users in the default users file.

The NULL realm can be reconfigured to handle authentication requests where the user doesn’t supply a domain name, such as for Windows domains. Any time a user name does not contain the realm portion of the NAI format *userid@realm*, the server automatically considers the user part of the NULL realm and applies the authentication method defined for NULL.

The default users file, found in fresh installations of the server, is used for testing the server installation and is set to perform password authentication on the default test_user. Remove this user from the file when you are done testing the initial installation.

There can be only one NULL realm per server. However it can exist once with each of the possible protocol options.

Wildcards

The wildcard * can be used in authentication realm names, as well as for proxy realms. See “Using Wildcards for Realms” on page 32 for more information.

Configuring TTLS Authentication

If you are using TTLS to authenticate users, you will need to define at least two realms for the group:

- One realm to establish the tunnel to secure a channel for another authentication method to authenticate the user. This is refered to as outer realm where the user configured on the supplicant is something like: anonymous@tunnelrealm.com.
- Inner authentication realms that indicate where user profiles are stored and the authentication method that will be used to authenticate actual users.

When defining a TTLS tunnel realm, the name must match what has been configured on the client device as the anonymous user’s realm name. Usually, this is the primary domain for the organization. When defining the inner authentication realms, the name should be the group’s own logon domain name or else use the NULL realm.

TTLS can be used to authenticate users on legacy RADIUS servers, once the tunnel has been established to secure the connection between the user and the RAD-Series Server.

The process for configuring TTLS realms is:

1. Configure your TTLS clients.
2. Add a tunnel realm, so the RAD-Series server knows the realm of the anonymous user.
3. Either:

   Add authentication realm(s) for the inner users.

   Or

   Configure the RAD-Series Server as a proxy to the legacy server where users will be authenticated. See “Proxying Requests to a Remote Server” on page 31.
4. Generate and install server-side certificates and keys. See “Administering Digital Certificates” on page 44.

Add a Tunnel Realm

To add a TTLS tunnel realm:

1 Change directory to `/<server configuration path>` (`/etc/opt/aaa/` by default).

2 In a text editor, add these lines for each TTLS tunnel/outer realm to the `EAP.authfile` of the following form:

```
Realm            -EAP      EAP     ""
{
   EAP-Type      TTLS
}
```

3 Replace *`Realm`* with the anonymous user's realm name as set up on the user machines.

4 Save and close `EAP.authfile`.

Add Authentication Realms

Follow the steps in "

Configuring Non-EAP Authentication Realms" on page 43 to add the authentication realm.

1 Change directory to `/<server configuration path>` (`/etc/opt/aaa/` by default).

2 For TTLS with PAP or CHAP, in a text editor, add these lines for each TTLS authentication/inner realm to the `authfile` of the following form:

```
Realm          -BIN  Auth-type  Auth-type-specific
```

Replace *`Realm`* with the real user's realm name as set up on the user machines. See "User Profile Storage" on page 48 for profile storage steps.

3 For TTLS with EAP, in a text editor, add these lines for each TTLS authentication/inner realm to the `authfile` of the following form:

```
Realm          -BIN  EAP   Auth-type-specific
{
   EAP-Type    EAP-name
}
```

Replace *`Realm`* with the real user's realm name as set up on the user machines and replace *`EAP-name`* with `MD5_Challenge` for MD5 or `MS_EAP` for MSCHAP. The **`EAP-Type`** line can be repeated to allow multiple EAP Types. See "User Profile Storage" on page 48 for profile storage steps.

4 Save and close `authfile`.

5 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

6 Stop and restart the AAA Server by running `radiusd.sh restart`

Configuring PEAP Authentication

Most PEAP supplicants do not support a separate realm for tunnels. Configuring a realm for the tunnel is optional - you can use one realm for both the inner user authentication and the outer tunnel, even though it is a two-phase authentication method.

You need only one authentication realm for each domain name.

The process for configuring PEAP realms is:

1 Configure your PEAP clients, usually access points.

2 Configure the outer/tunnel realm.

3 Add authentication realm(s) for the inner users.

4 Generate and install server-side certificates and keys. See "Administering Digital Certificates" on page 44.

Configure PEAP Outer/Tunnel and Inner Realms with different realm names

1 Change directory to `/<server configuration path>` (`/etc/opt/aaa/` by default).

2 In a text editor, add these lines for each PEAP realm to the `EAP.authfile` of the following form:

```
Realm-Outer     -EAP            EAP     ""
{
   EAP-Type     PEAP-Ver
}

Realm-Inner     -EAP    -BIN    EAP     ""
{
   EAP-Type     PEAP-Ver
   {
      EAP-Type  Peap-Inner-Type

   }
}
```

3 Replace `Realm-Outer` with the anonymous user's realm name as set up on the user machines.

4 Replace `Realm-Inner` with the realm portion of the user's authentication name as set up on the user machines.

5 Replace `PEAP-Ver` with PEAP for PEAP protocol version 1 or PEAPv0 for PEAP protocol version 0. If you are using any third party supplicants that do not support PEAP version 1, you will need to select "PEAPv0". Cisco devices often need version 0.

6 Replace `Peap-Inner-Type` with the inner EAP protocol to be used for authentication Repeat this line if multiple authentication protocols are needed.

7 Save and close `EAP.authfile`.

8 In a text editor, define the authentication real by adding these lines for each PEAP inner realm to the `authfile` of the following form:

```
Realm-Inner/PEAP  -EAP   -BIN   Profile-Storage-Type   ...
```

9 Replace `Realm-Inner` with the realm portion of the user's authentication name as set up on the user machines.

Replace `Profile-Storage-Type` with the profile storage that contains the user's profile. Such as a realm file (`iaaaFile`) or LDAP (`ProLDAP`). See "User Profile Storage" on page 48 for further details.

Note: The "`/PEAP`" after the ralm name is not mentioned in the "User Profile Storage" section since it is unique to PEAP but don't forget to add it.

10 Save and close `authfile`.

11 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

12 Stop and restart the AAA Server by running `radiusd.sh restart`

Configure PEAP Outer/Tunnel and Inner Realms with the same realm names

1 Change directory to `/<server configuration path>` (`/etc/opt/aaa/` by default).

2 In a text editor, add these lines for each PEAP inner realm to the `EAP.authfile` of the following form:

```
Realm            -EAP     -BIN      EAP     ""
{
   EAP-Type      PEAP-Ver
   {
      EAP-Type   Peap-Inner-Type

   }
}
```

3 Replace *`Realm`* with the realm portion of the user's authentication name as set up on the user machines.

4 Replace *`PEAP-Ver`* with PEAP for PEAP protocol version 1 or PEAPv0 for PEAP protocol version 0. If you are using any third party supplicants that do not support PEAP version 1, you will need to select "PEAPv0". Cisco devices often need version 0.

5 Replace *`Peap-Inner-Type`* with the inner EAP protocol to be used for authentication Repeat this line if multiple authentication protocols are needed.

6 Save and close `EAP.authfile`.

7 In a text editor, define the authentication real by adding these lines for each PEAP inner realm to the `authfile` of the following form:

```
Realm/PEAP         -EAP   -BIN   Profile-Storage-Type   ...
```

8 Replace *`Realm`* with the realm portion of the user's authentication name as set up on the user machines.

Replace *`Profile-Storage-Type`* with the profile storage that contains the user's profile. Such as a realm file (`iaaaFile`) or LDAP (`ProLDAP`). See "User Profile Storage" on page 48 for further details.

NOTE: The "/PEAP" after the ralm name is not mentioned in the "User Profile Storage" section since it is unique to PEAP but don't forget to add it.

9 Save and close `authfile`.

10 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

11 Stop and restart the AAA Server by running `radiusd.sh restart`

Configuring TLS Authentication

TLS does not access a data store for user authenication. Instead, this method requires certificates to be installed on both the RAD-Series Server and on each user workstation. You can optionally use a data store to configure a user’s profile which is used to specify attributes to be sent in the Access Accept (Authorization information).

The process for configuring TLS realms is:

1 Add authentication realm(s).
2 Generate and install server and client-side certificates. See “Administering Digital Certificates” on page 44.
   - For the server, you will need two certificates: a CA (certificate authority) certificate and a server-specific certificate.
   - For each user workstation, you will need the same CA certificate and a client-specific certificate. Client certificates must have the User-Name in the one of the three vailid fields of the certificate. See “Client User Name Attribute” on page 58.

Add Authentication Realm Entry

Follow these steps to add the TLS authentication realm entry.

1 Change directory to `/<server configuration path>` (`/etc/opt/aaa/` by default).

2 In a text editor, add these lines for each TLS realm to the `EAP.authfile` of the following form:

```
Realm           -EAP     EAP      ""
{
    EAP-Type    TLS
}
```

3 Replace `Realm` with the domain portion of the “Client User Name Attribute” as specified on page 58.

4 Save and close `EAP.authfile`.

5 Change directory to /<*server utilities path*> (/opt/aaa/utl by default).

6 Stop and restart the AAA Server by running radiusd.sh restart

Add Authorization Realm Entry - Optional

Follow these steps to add the TLS authorization realm entry.

1 Change directory to `/<server configuration path>` (`/etc/opt/aaa/` by default).

2 In a text editor, add these lines for each TLS realm to the `authfile` of the following form:

```
Realm          -EAP  -BIN   Profile-Storage-Type   ...
```

3 Replace `Realm` with the domain portion of the "Client User Name Attribute" as specified on page 58.

Replace `Profile-Storage-Type` with the profile storage that contains the user's profile. Such as a realm file (`iaaaFile`) or LDAP (`ProLDAP`). See "User Profile Storage" on page 48 for further details.

4 Save and close `authfile`.

5 Change directory to /<*server utilities path*> (/opt/aaa/utl by default).

6 Stop and restart the AAA Server by running radiusd.sh restart

Configuring Non-EAP Authentication Realms

Follow these steps to define each user realm to be authenticated by the RAD-Series Server.

1 Change directory to `/<server configuration path>` (`/etc/opt/aaa/` by default).

2 In a text editor, add this line for each authentication realm to the `authfile` of the following form:

`realm-name -flags auth-type auth-parameter`

3 Replace *`realm-name`* with the domain portion of the user name.

4 The auth-type defines the user profile storage, choose the type of data store you're using for this realm.

- o Realm File - `iaaafile` - flat file identified by name in `auth-parameter`
- o Users - - profile is stored in the default "`users`" file. This file is searched if no other type is defined for the realm.
- o Allow - `Allow` - no authentication, allow all users in this realm
- o Deny - `Deny` - no authentication, deny all users in this realm
- o OS Security Database - `Unix-PW` - UNIX password storage
- o RSA SecurID/ACE server - `SecurID` - RSA SecurID identification and authentication. Requires special licensing option and is compatible with RSA Authentication Manager versions 6.1.2 and later, 7.1 SP2, 7.1 SP3 and 8.1 SP2 and later.
- o LDAP - `ProLDAP` or `ProLDAPX` – Find user profile in a LDAP directory

  See "ProLDAP Authentication Type" on page 211 for configuring ProLDAP. For some environments it would be better to use ProLDAPX which can yield higher performance in a multi CPU server. For this, see "ProLDAPX Authentication Type" on page 216.

See "Authfile Entry Syntax" on page 207 for *`-flags, auth-type`* and *`auth-parameter`* values.

5 Save and close `authfile`.

6 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

7 Stop and restart the AAA Server by running `radiusd.sh restart`

Administering Digital Certificates

TLS, TTLS, and PEAP use certificates for authentication. To configure realms using these methods you must:

- Generate certificates and the corresponding private keys for the RADIUS server.
- Install signed certificates and private keys in the RAD-Series Server's directories.
- For TLS, you must also generate certificates and private keys for each user workstation that will access the network.

If you are supporting multiple realms, configure digital certificates after you've added all of your realms.

You can deploy digital certificates in an environment with supporting infrastructure—for example, an organization with a PKI and user-specific certificates.

Defining Digital Certificate Locations

The RAD-Series Server uses the default self-signed certificates by default for TLS, TTLS, and PEAP. If you want to use your own certificates, you must define where the required certificates reside on the server.

1 Change directory to `/<server configuration path>` (`/etc/opt/aaa/` by default).

2 In a text editor, modify lines in `aaa.config` file in the `security.paths {}` for the four paths related to certificates.

3 Change `Certificate-Path` from the current path to the full path to your RAD-Series Server certificate in `.pem` or `.cer` format.

4 Change `Key-Path` from the current path to the full path of your file in `.pem` or `.cer` format that contains the private key used to generate the RAD-Series Server certificate. This file cannot be encrypted.

5 Change `CA-Path` from the current path to the full path to your CA certificate for the client certificate. This is used by the RAD-Series Server to authenticate client certificates. The CA certificate for the client certificate must be in `.pem` format.

6 Change `Random-Path` from the current path to the full path of your random seed file which is used to generate keys.

7 Save and close `aaa.config`.

8 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

9 Stop and restart the AAA Server by running `radiusd.sh restart`

Using the “Self-Signed” Digital Certificates

The RAD-Series Server is installed with a sample set of self-signed” digital certificates. You can use the self-signed certificates in test environments for TLS, TTLS and PEAP.

The self-signed server certificates are in `/<server configuration path>/security/` (`/etc/opt/aaa/security/` by default). They are:

- `rsa_cert.pem` — server certificate
- `rsa_key.pem` — server key
- `ca_list.pem` — list of client CA certificates
- `sampleclientcert.p12` — sample client certificate

For TTLS and PEAP

If you are using TTLS or PEAP, the default certificates are safe to deploy in your production environment. The RAD-Series Server is its own Certificate Authority. If you are managing multiple RAD-Series Servers, you must have the same set of digital certificates on each server in your configuration. Pick one of your RAD-Series Servers and copy the set of self-signed digital certificates to every RAD-Series Server in the configuration. You should save each RAD-Series Server's original self-signed certificates for future use.

For TLS

If you are using TLS, use the default certificates to simulate and troubleshoot TLS certificate administration before you deploy your own enterprise certificates.

1. Copy `/<server configuration path>/security/rsa_cert.pem` to the CA storage on the supplicant.
2. Copy `/<server configuration path>/security/sampleclientcert.p12` to the user’s PC and add it to the certificate storage of the supplicant:
   - The pass phrase is blank for `sampleclientcert.p12`
   - The user name for `sampleclientcert.p12 is`: sampleclient@samplecompany.com
3. Configure a TLS realm for samplecompany.com on the RAD-Series Server

Using Your Own Digital Certificates and Keys

If you don't use the default self-signed RAD-Series Server certificates, you must generate certificates and the corresponding private keys for the RADIUS server and (when using TLS) each workstation that will access your network services. To acquire a server or client-specific certificate, submit a certificate signature request (CSR) to a certificate authority (CA), such as VeriSign or Microsoft.

Note: For TLS, the same Certificate Authority must be used to sign certificates for both the RAD-Series Server and the user workstation.

Obtaining Certificates and Creating Keys

Follow these steps to create a CSR and corresponding key:

1 Create or choose a directory for the certificates and your private key. Because the private key is stored unencrypted, it is very important to restrict access to the directory.

2 Use the OpenSSL Certificate Request Generator or another utility to create two files:

- the key, `rsa_key.pem`
- the CSR, `rsa_cert.pem`

To create these files with openssl, enter the following at the command line prompt:

```
openssl req -new -nodes -out req.pem -keyout Key-location
```

For Key-location specify the `rsa_key.pem` file, including the path of the directory you have chosen to store the server- or client-specific key. For example: `/etc/opt/aaa/security/rsa_key.pem`.

Note: Do not protect the key with a passphrase.

3 You will be prompted to enter information that will be used to generate the private key file and CSR file. Complete the information according to the following table:

Field	Enter
Country code	Two-letter ISO code for your country. The code for the United States is US.
Organizational unit name	Name of the division, department, or other organizational unit that will be authenticated using this certificate.
Organization name	Name of your organization. Verisign may require any host names to belong to a domain registered to this organization.
E-mail address	The e-mail address that should be used to receive the certificate.

Field	Enter
Locality name (city)	Name of your city or town. If you operate with a license granted by a city, this field is required; enter the name of the city that granted your license.
State name	Name of the state or province in which your organization operates. Do not abbreviate.
Common name	Fully-qualified domain name of the server or client that will use this certificate.

4 Submit the CSR to VeriSign or another certificate authority. Request the certificate in Base-64 format.

You will receive the server or client-specific certificate through the e-mail address you specified when you used openSSL to generate the CSR or the certificate may be downloaded from a web page.

How the certificate is submitted and delivered varies according to the certificate authority that you use. For example, a CSR is submitted to Microsoft and the certificate is downloaded from a web page.

Installing Server Certificates and Keys

1 Add the CA certificate to the `ca_list.pem` file (found in `/etc/opt/aaa/security/` by default) by copying and pasting the contents of the certificate into the file.

2 Copy the files for the server-specific certificate and key contents into the RAD-Series Server security directory (`/etc/opt/aaa/security/` by default).

Installing Client Certificates and Keys

For each user workstation where you will make a TLS connection:

1 Install the CA certificate in the Trusted Root Certification Authorities store.

2 Install its user-specific (or computer-specific) key and certificate:

- Install computer-specific certificates in the Local Computer certificate store. Set the Subject Alternative Name property to match the FQDN of the wireless client computer account, which should be the same as the User-Name (userid@realm).
- Install user-specific certificates in the Current User certificate store. Set the Subject Alternative Name property to match the universal principal name (UPN) or common name of the user account, which should be the same as the User-Name (userid@realm).

User Profile Storage

Defining Users

User profiles may be stored locally (in realm flat files or the default users file) or in an external source, like an LDAP directory. Use the procedures in this section to add users to realm files for local storage. Otherwise, use the interface to your data repository to add user profiles. If you have a small number of users from different realms, you can store all of their profiles in the default users file.

In following procedures you will modify the realm file you specified when configuring the realm.

For information on using LDAP storage, see “Using External Data Stores” on page 104 and “Using LDAP Directories” on page 49.

User A-V Pairs

User profile information is stored as a set of Attribute-Value, A-V, pairs. These A-V pairs mostly fall into two primary groups:

- Check/deny items--for authorization (simple policy). This includes checking for Service Type, NAS/Login ID, Caller/Calling Station ID, etc.
- Reply items--for provisioning. This includes any session data returned to the access device, such as session control limits, filters, callback numbers, IP addresses, port limits, and reply messages.

All user A-V pairs put into the file, represent a **one-to-one match** between the attribute and a specified value. You cannot specify a list of values for any single user attribute.

Note: Wireless access points that adhere to the 802.1X standard should support the reply item attributes. Verify the capabilities of your access points with the access point vendor.

See “Configuration Files” on page 164 for user A-V pair syntax and a description of Interlink-specific user attributes. Standard RADIUS attributes are defined in the RAD-Series Server’s `dictionary` and `dictionary.*` files, by default in `/etc/opt/aaa`.

User Name Formats

A user name in Network Access Identifier (NAI) format should look like *userid@realm*, for example: "you@yourcompany.com.” When adding users to local realm files, supply only the `userid` part of their NAI format login string. When adding users to deualt users file, supply their full NAI login string.

Adding User Profiles

Generally it is wise to hash passwords for security reasons but if you do, then some authentication methods will not work. For PEAP, and TTLS, choose a password hashing mechanism compatible with the inner realm authentication method. It is up to you to generate the hash of the password before entering into the file.

Authentication Method	Hashing Mechanisms
PAP	Any
MS-CHAP	Plain Text or NT Hash
EAP-GTC	Any
EAP-MD5	Plain Text or MD5 Hash
EAP-TLS	None

Here is a list of the supported hashes and there prefix:

Hash	Prefix Added to Identify the Hash
NT	{x-nthash}
LM	{x-lmhash}
MD5	{md5}
SHA	{sha}
SSHA	{SSHA}
Crypt	{crypt}

Using LDAP Directories

Authentication realms that use LDAP needs additional parameters to allow the RAD-Series server to access the LDAP directory. These parameters let you identify the LDAP directories from which the RAD-Series Server will retrieve user profiles. They also specify how the lookup is done. See "ProLDAP Authentication Type" on page 211 for configuring ProLDAP. For some environments it would be better to use ProLDAPX which can yield higher performance in a multi CPU server. For this, see "ProLDAPX Authentication Type" on page 216.

Using Local Storage – Realm Files

To add new users to a realm file:

1 Change directory to `/<server configuration path>` (`/etc/opt/aaa/` by default).

2 In a text editor, open "`Realm.users`", where `Realm` is the realm name used in the `authfile`. Add the user to the file using the following format:

```
User-Name check-items
   reply-item,
   reply-item,
   ...
```

See "Realm files (.users) and default users file" on page 228 for a complete description of this entry.

Note: For realm files the one difference is that the User-Name is replaced with the User-Id. The "`@realm-name`" is omitted

---

3 Follow the procedures in "Controlling and Provisioning Sessions" on page 52 to complete the user profile if necessary.

4 Save and close the file. For realm files it is not necessary to reload the server.

Also see "Local Storage – iaaaFile" on page 210.

Using Local Storage – Default Users File

To add new users to the default users file:

1 Change directory to `/<server configuration path>` (`/etc/opt/aaa/` by default).

2 In a text editor, open the "`users`" file. Add the user to the file using the following format:

```
User-Name@Realm-Name check-items
   reply-item,
   reply-item,
   ...
```

See "Realm files (.users) and default users file" on page 228 for a complete description of this entry.

Note: For the default users file, the one difference is that the "`@Relam-Name`" is required unless the realm name is NULL.

---

3 Follow the procedures in "Controlling and Provisioning Sessions" on page 52 to complete the user profile if necessary.

4 Save and close the file.

5 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

6 Stop and restart the AAA Server by running `radiusd.sh reload`

Also see "Local Storage in Default users File" on page 210.

Using Local Storage – UNIX Password

To add a user to the UNIX password database, just use the normal UNIX tools on the system. Also see "Unix Password" on page 210.

Using RSA SecurID

See "SecurID" on page 210.

Controlling and Provisioning Sessions

Complete the Definig Users section above to enter provisioning information, such as:

- Filters
- Session limits
- Point of access

Session control attributes appear as check, deny, or reply items in the realm file or LDAP.

Specify Login Service Type

To configure users for login service types (such as dumb-terminal access):

1. Complete the steps to add a new user.
2. Add the check item Service-Type = "Login".
3. Add any other check items needed to restrict the user origin or destination.

Specify Framed Service Type

To configure dial-up users for framed service types:

1. Complete the steps to add a new user.
2. Add the check item Service-Type = "Framed".
3. Add the check item for Framed-Protocol = "PPP".
4. Optionally, add any other Framed check items.

Set Timeout Values

1. Complete the steps to add a new user.
2. Add the reply items:
   - Session-Timeout — how many consecutive seconds user can access the service.
   - Idle-Timeout — how many consecutive seconds of idle connection time can pass before the session is terminated.

Note: The RAD-Series Server does not enforce either timer, it just sends then to the NAS for it to use.

Establish Filters

If you’ve defined filters on your access devices, you can specify which filter to apply to the user. You can only associate one filter with the user.

1 Complete the steps to add a new user.

2 Add the reply item `Filter-Id = "`*`filter-name`*`"`, where *`filter-name`* exactly as set up on your device in Filter ID.

Establish Callback Number

If you’re setting up a callback service type.

1 Complete the steps to add a new user.

2 Add the reply item `Callback-Number = "`*`number`*`"`, where *`number`* is the exact string of numbers to dial.

or

Add the reply item `Callback-ID = "`*`ID`*`"`, where *`ID`* is the name of a place and number to be interpreted by the access device.

Control Access by Device

To limit users to access only through specific devices:

1 Complete the steps to add a new user.

2 Add the check item `NAS-IP-Address = "`*`IP`*`"`, where *`IP`* is the NAS IP address.

3 Optionally, enter:

- `NAS-Port = "`*`number`*`"`, where *`number`* is the port number that must appear in an Access-Request for authorization to succeed
- `NAS-Identifier = "`*`ID`*`"`, where *`ID`* is the NAS identifier that must appear in an Access-Request for authorization to succeed
- `NAS-Port-Type = "`*`Type`*`"`, where *`Type`* is the NAS Port Type, if NAS differentiates among its ports

Assign a Static IPv4 Address

1 Complete the steps to add a new user.

2 Add the reply item `Framed-IP-Address = "`*`IP`*`"`, where *`IP`* is the IPv4 address.

3 If the user is a router to a network, also enter `NAS-IP-Netmask = "`*`IP`*`"`, where *`IP`* is the Framed IP Netmask.

Define DHCP IPv4 Address Pool

Use this to associate an address pool with the user if you're using the RAD-Series Server as a DHCPv4 relay. Be sure to also enable DHCP in Server Properties. See "Using DHCP" on page 67.

1 Complete the steps to add a new user.

2 Add the reply item: `Address-Pool = "`*`name-of-pool`*`"`.

Control Access by Dial-Up Number/MAC Address

This configuration limits the user to calling from or to the specified dial-up number or, in the case of a wireless network, to the station's MAC address.

To deny access to specific numbers/addresses, such as 800 numbers, see "Deny Access" on page 55.

1 Complete the steps to add a new user.

2 To allow the user to only call **from** a specific phone number or machine, add the check item: `Calling-Station-Id = "`*`Calling-ID`*`"`, where *`Calling-ID`* is the number or MAC address.

3 To allow the user to only call **to** a specific phone number or access point, add the check item: `Called-Station-Id = "`*`Called-ID`*`"`, where *`Called-ID`* *is* the number or MAC address.

Set Reauthentication Session Timeout

For wireless users, you can set how often (in seconds) the access point will attempt to reauthenticate the user. If authentication fails, the access point terminates the session.
To do this procedure you must enable Session Tracking for the user's realm. See "Configuring Non-EAP Authentication Realms" on page 43.

Note: Some access points do not properly support session tracking with RADIUS accounting messages.

1 Complete the steps to add a new user.

2 Add the reply item:

```
Terminate = "1"
```

3 Add the reply item:

`Session-Timeout = "`*`Number of seconds`*`"`

Override Concurrent Session Limit

To do this procedure you must enable Session Tracking for the user’s realm. See “Configuring Non-EAP Authentication Realms” on page 43.

Note: Enabling Session Tracking sets the number of individual concurrent sessions to the global Simultaneous Use value set in Session Properties under the Server Properties page. You can override this number by defining a Simultaneous-Use value for an individual user in the user profile sessions (zero denies all sessions for the user, while -1 imposes no session limits).

1 Complete the steps to add a new user.

2 Add the check item:

`Simultaneous-Use = “`*`Max-number-sessions`*`”`

Set Port Limit

If you’re using Multilink PPP, you can set the maximum number of ports that may be assigned to the user for a single session.

1 Complete the steps to add a new user.

2 Add the reply item: `Port-Limit = "`*`max`*`"`, where *`max`* is the maximum number of ports allowed.

Deny Access

To deny a user access through a specific connection point:

1 Complete the steps to add a new user.

2 Add any or all of the following check items:

- `NAS-Port != “`*`Port-number`*`”`
- `NAS-Identifier != “`*`value`*`”`
- `Calling-Station-ID != “`*`AP-MAC-address`*`”`
- `Called-Station-ID != “`*`dial-up number`*`”`

Note: Spaces before and after the “!=” are required.

Policy-Pointer

Advanced policy can be used apply special criteria for deciding if the user should be allowed in or which reply items to return. It can be invoked by defining the `policy-pointer` attribute with the decision file name as its value. See “User/Realm policy” on page 113 for information on using this attribute.

1 Complete the steps to add a new user.

2 Add the check item:

```
Policy-Pointer = “file.name”
```

Defining Server Properties

These properties require modification of the RAD-Series Server's `aaa.config` and/or `las.conf` files.

Modifying Server Properties

1. Change directory to `/<server configuration path>` (`/etc/opt/aaa/` by default).
2. In a text editor, add or modify lines for each server property in the `aaa.config` as needed. See "Server Variable Syntax" on page 170 for the complete set of properties and the format for each entry.
3. Save and close `aaa.config`.
4. Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).
5. Stop and restart the AAA Server by running `radiusd.sh restart`

See the topics below for a list of configurable properties.

Certificate Properties

These properties specify the location of certificates used in TLS, TTLS, and PEAP authentication. Any security files that were not installed with the server must be manually installed in these directories.

Server Certificate Path (optional)

Used in EAP-TLS, TTLS, and PEAP message processing. Full path to the RAD-Series Server certificate in `.pem` or `.cer` format. The default is `/etc/opt/aaa/security/rsa_cert.pem`, which is a self-signed certificate created at installation time.

Server Private Key Path

The full path to the private key file associated with the RAD-Series Server certificate for EAP-TLS, TTLS, and PEAP. This file cannot be encrypted. The default is `/etc/opt/aaa/security/rsa_key.pem`.

Client Certificate Authority Path (optional)

Used in EAP-TLS, EAP-TTLS, and EAP-PEAP message processing. Full path to the CA certificate used to sign the RAD-Series Server certificate. The default is `/etc/opt/aaa/security/ca_list.pem`, which is created with the self-signed server certificate at installation time.

Certificate Revocation List Path (optional)

Used in EAP-TLS message processing. Full path to a list of prohibited client certificates in `.pem` or `.cer` format. No default is set. If this path is specified but no CRL file is found, the server will not authenticate.

Random Seed Path (optional)

Used in EAP-TLS, EAP-TTLS, and EAP-PEAP message processing. Full path to the random seed file. This file may contain any type of random data. The default is `/etc/opt/aaa/security/random.rnd`, which was created during server installation.

Client User Name Attribute

Used only for TLS authentication. The name in the client certificate used to validate the User Name from the TLS Access-Request. Choose which field and attribute to match:

- **Subject:CommonName** (default) — Use the CommonName (CN) from the Subject field.
- **Subject:EmailAddress** — Use the Email Address (E) from the Subject field.
- **SubjectAltName:RFC822Name** — Use the RFC822Name from the SubjectAltName field of the certificate's subject alternative name extension.
- **Check all attributes** — Search all of the above three fields.

SSL Debug Level (optional)

The minimum RAD-Series server debug level required so that SSL debug output is produced in the radius.debug file. A value of 0 disables SSL debug output.

ECDSA Certificate Path, ECDSA Key Path, ECDHE Parameter Curve (optional)

These parameters are described in "EAP-TLS Server Properties" on page 181.

DHCPv4 Relay Properties

These properties are used to configure the RAD-Series Server as a DHCPv4 relay agent. For more information, see "Using DHCP" on page 67.

DNS Update Properties

These properties determine how the RAD-Series Server resolves IP addresses with DNS.

DNS Refresh Interval

Interval in seconds at which to refresh the IP addresses for Access Devices and Proxies that have been configured by host name. Default is 3600.

DNS Refresh Time Frame

When a DNS entry for a configured client expires (needs refreshing), all other clients that would normally be refreshed within this number of seconds are refreshed immediately. Default is 60.

IP/UDP Properties

These properties configure IP/UDP operations.

Enable IPv6 Communications

This parameter indicates whether the RAD-Series Server will engage in IPv6 communications, i.e. listening on IPv6 interfaces or sending RADIUS packets to/from IPv6 addresses. The default value is NO.

UDP receive buffer size for proxy sockets

This parameter indicates the requested UDP buffer size for a proxy socket. The minimum value is 8192 bytes (8kB); the maximum is 8388608 bytes (8 MB). The default value is 0 which lets the operating system set the UDP receive buffer size.

Default local IPv4 address for IPv4 proxy socket

This parameter specifies the local IPv4 address the RAD-Series Server will use when proxying RADIUS requests via IPv4. The default value is 0.0.0.0, the IPv4 ANY address, which indicates the system will chose the local IPv4 address when transmitting a RADIUS request. This value can be overridden by configuring a Proxy-specific source IP address.

Default local IPv6 address for IPv6 proxy socket

This parameter specifies the local IPv6 address the RAD-Series Server will use when proxying RADIUS requests via IPv6. The default value is ::, the IPv6 ANY address, which indicates the system will chose the local IPv6 address when transmitting a RADIUS request. This value can be overridden by configuring a Proxy-specific source IP address.

RADIUS Listen Socket Properties

These properties configure the sockets on which the RAD-Series Server listens for RADIUS requests.

The following parameters are configured for each listen socket.

IP Address

The IP address can be an IPv4 specific address, an IPv6 specific address, the IPv4 ANY address (0.0.0.0), or the IPv6 ANY address (::). This parameter is required. If the IPv4 ANY address is specified, the server will listen on all IPv4 interfaces. If the IPv6 ANY address is specified, the server will listen for IPv4 and IPv6 messages on all interfaces. There is no default value.

Authentication port

The UDP port number on which to receive authentication requests. The minimum value is 0 and the maximum value is 65535. There is no default value. If set to the special value of zero, the RAD-Series Server will execute a hierarchy of steps to determine the authentication port to listen on:

- If there is an Authentication Port specified on the Administration: Start Options screen, use that value, else
- If the environment variable RAD_AUTH_PORT is defined, use that, else
- If a RADIUS authentication port is configured in the /etc/services file, use that, else
- Use 1812, as defined by the RADIUS RFC.

If the Authentication Port is not configured, the server will not open an authentication listen socket for the given IP Address.

Accounting port

The UDP port number on which to receive accounting requests. The minimum value is 0 and the maximum value is 65535. There is no default value. If set to the special value of zero, the RAD-Series Server will execute a hierarchy of steps to determine the accounting port to listen on:

- If there is an Accounting Port specified on the Administration: Start Options screen, use that value, else
- If the environment variable RAD_ACCT_PORT is defined, use that, else
- If a RADIUS accounting port is configured in the `/etc/services` file, use that, else
- Use 1813, as defined by the RADIUS RFC.

If the Accounting Port is not configured, the server will not open an accounting listen socket for the given IP Address.

Authentication port UDP receive buffer size

This optional parameter indicates the requested UDP buffer size for an auth listen socket. The minimum value is 8192 bytes (8kB); the maximum is 8388608 bytes (8 MB). The default value is 0 which lets the operating system set the UDP receive buffer size.

Accounting port UDP receive buffer size

This optional parameter indicates the requested UDP buffer size for an accounting listen socket. The minimum value is 8192 bytes (8kB); the maximum is 8388608 bytes (8 MB). The default value is 0 which lets the operating system set the UDP receive buffer size.

Message Handling Properties

These properties determine how the RAD-Series Server performs RADIUS message handling.

Hold Replies

Number of seconds the RAD-Series Server holds a request after replying to it in case a retransmission is necessary. The default is 6. The value should be twice the default retransmission period of the access devices involved. This does not apply to packets that are forwarded to another server. When set to 0, a special behavior is invoked where the RAD-Series Server does not change the hold time for a request.

Note: Using the special value of 0 or a hold time greatly in excess of the retransmission policy of an access device may cause the authentication and accounting queues to grow large, degrading server performance. Tailor this value by the *total* and *holding* values reported on a per-request basis.

Global Retry Limit

Maximum number of retransmissions allowed before a RETRY event occurs (a RETRY event is similar to a TIMEOUT event and is handled by the default FSM). The purpose of this is to catch an authentication request and perform some action when a certain number of retransmissions from an access device occur. The default is 0 and no limits are imposed.

Maximum Accounting Requests

The maximum number of active accounting requests to be handled by the RAD-Series Server within the Hold Replies time. The default is 40,000. When this limit is exceeded, the server drops the request and logs the event.

Maximum Authentication Requests

The maximum number of active authentication requests to be handled by the RAD-Series Server within the Hold Replies time. Default is 40,000. When this limit is exceeded, the server sends an Access-Reject message and logs the event.

Maximum Send Message Size

The maximum size in bytes for an outbound RADIUS packet. The minimum value is 4096. The default value is 16536.

This property is primarily intended for supporting a customized server configuration that might transmit very large packets. Limiting it to be the UDP MTU for the network will prevent excessively large packets from being forwarded (or replied to) in certain circumstances.

Maximum Receive Message Size

The maximum size in bytes for an inbound RADIUS packet. The minimum value is 4096. The default value is 16536.

This property is primarily intended for supporting an access client that might transmit very large packets.

Miscellaneous Properties

These properties are used for performance tuning and other RAD-Series Server behavior.

CUI Encryption Secret

This parameter configures the secret used for encryption of the real user identity into a generated Chargeable-User-Identity (CUI) when needed. The CUI Encryption Secret can be from 0 to 127 characters long. It can contain any printable character except for quotes. If not configured or if configured as the empty string, then a default internal secret will be used to generate the CUI. The default value is an empty string.

If a CUI is configured in a user's profile, then this configured CUI will be used, and the user's CUI will then change only as often as his configuration changes. If a CUI is not configured for a user, then the server will generate a CUI when needed, and the user's generated CUI will change weekly if no CUI Encryption Secret is configured and if one is configured then it changes weekly or as often as the CUI Encryption Secret is changed, whichever is sooner.

Logfile Compression

This parameter indicates if the old logfile should be compressed when rolled over (by date or by size) to a new logfile. Enabled (Yes) is the default.

Maximum logfile File Size

The maximum size in bytes of the server log file and accounting log file. The minimum value for this parameter is 65536; the maximum is 2147483647 (default).

Maximum logfile Line Length

The maximum length, in bytes, of the server's logfile lines. The minimum value for this parameter is 1024; the maximum is 16384, and the default is 4096. Lines longer than this are truncated.

Maximum errorlog File Size

The maximum size in bytes of the server's errorlog file. The minimum value for this parameter is 65536; the maximum (default) is 2147483647. When errorlog reaches this size, it is rolled over.

errorlog Message Logging Levels (ACEWN)

The levels of logfile messages to include in the errorlog file. 'A' represents LOG_ALERT, 'C' represents LOG_CRIT, 'E' represents LOG_ERR, 'W' represents LOG_WARNING, and 'N' represents LOG_NOTICE. The default is 'ACEWN' i.e. include all non LOG_INFO messages. This parameter is only relevant if writing to the errorlog file is enabled.

Microsoft Host-Based Authentication

If enabled (Yes), the server will strip “host/” from the EAP-Identity sent by Microsoft clients configured to authenticate as computer on a wireless connection. When using TLS authentication, the remaining string is compared to the selected Client User Name Attribute field in the client certificate. When using PEAP/MSCHAPv2 authentication, the remaining string is used as the inner-realm userid. Enabled (Yes) is the default.

Tunneled EAP MTU Reduction (optional)

The number of bytes by which to reduce the Framed-MTU AVP value when a EAP-PEAP/EAP-TTLS inner (tunnel) request is created. EAP-PEAP and EAP-TTLS must have enough room in the outer packet to contain the inner (tunneled) EAP conversation plus any attributes (such as Reply-Message) that must be sent outside the tunnel during the exchange. This parameter specifies how much of the outer FramedMTU value is reserved for these non-tunneled attributes when constructing an inner reply.

ProLDAP Properties

These properties specify how the RAD-Series Server interacts with LDAP servers. They are global properties used for all LDAP servers. The parameters are described in “LDAP Connection Properties” on page 185.

RSA SecurID Properties

The RSA SecurID authentication supports connections to RSA SecurID Authentication Manager versions 6.1.2 and later, 7.1 SP2, 7.1 SP3 and 8.1 SP2 and later. These properties specify how the RAD-Series Server interacts with RSA SecurID servers and logging of the RSA SecurID authentication subprocess. They are global properties used for all RSA SecurID servers. They are described in “RSA SecurID Server Properties” on page 203.

Session Properties

These properties determine how the RAD-Series Server manages sessions. These parameters are located in the `las.conf` file. The complete description of them can be found in “LAS Session Configuration” on page 233.

SNMP Properties

The properties for SNMP can be found in “SNMP Server Properties” on page 181. SNMP is disabled by default.

For more information, see “Using SNMP” on page 66.

Tunneling Properties

These properties configure the RAD-Series Server’s handling of tunneling hints. For more information, see “Tunneling” on page 69.

Using SNMP

The RAD-Series Server can exchange information with any Simple Network Management Protocol (SNMP) master agent software that supports the AgentX protocol (see RFC 2741 for more technical information about this protocol).

At startup the server automatically activates its SNMP subagent if SNMP is enabled and the subagent registers the application with the SNMP master agent.

Enabling and Disabling SNMP

See "SNMP Server Properties" on page 181 to see how to configure SNMP in the aaa.config file.

MIB Objects

Information exchanged through SNMP is represented by objects in the Management Information Base (MIB). The MIB includes extensions for RADIUS authentication and accounting servers that are supported by the RAD-Series Server. The MIB objects that the RAD-Series Server will interpret to the SNMP master agent are defined in the files:

- RADIUS-ACC-SERVER-MIB.txt
- RADIUS-AUTH-SERVER-MIB.txt
- RADIUS-ACC-CLIENT-MIB.txt
- RADIUS-AUTH-CLIENT-MIB.txt

These files can be found in the server's configuration directory, by default /etc/opt/aaa/.

Since the RAD-Series Server performs both authentication and accounting functions, some of the MIB objects return duplicate information. The RAD-Series Server acts as a client when proxying requests

All of the MIB object requests that are sent by the management workstation to the RAD-Series Server in SNMP requests are read-only, except for radiusAuthServConfigReset and radiusAcctServConfigReset, which allow a write operation.

Note: When you check the RAD-Series Server status, the server will increase the radiusAuthServTotalAccessRequests count but will not increase radiusAuthServAccessRequests for any client. This behavior will result in a total authentication request count that will not equal the sum of requests received by individual clients.

See RFCs 4668 thorough 4671 for descriptions of the MIB objects for RADIUS authentication and accounting servers and clients. These MIBs support IPv4 and IPv6 addresses. The older server MIBs (RFCs 2619 and 2621) have been deprecated and are no longer supported by the RAD-Series Server.

Using DHCP

The RAD-Series Server can be configured to act as a DHCPv4 relay to request IPv4 address assignments from a DCHP server and pass them to an access device. You can associate DHCP address pools with either individual users or realms.

The RAD-Series Server does **not** support:

- DHCP server failover
- Relay agent information option
- Dynamic DNS updates
- The DHCPv6 protocol, or the assignment of IPv6 addresses
- Communication with the DHCP server via IPv6. Only IPv4 communications are supported.

See RFC 2131 for more technical information about the DHCP protocol.

Required DHCP Server Features

Required Features	Recommended Features
Assign addresses from its IPv4 address pools based on the User Class or Vendor Class Identification attribute.	Assign IPv4 addresses outside the network it resides in. Many RADIUS/DHCP deployments will require this capability. Send to ports above the well-known port range (0- 1023). Without this capability the RAD-Series Server will not be able to run as a non-root process.

Process for Using DHCP

To use the RAD-Series Server as a DHCP relay, you’ll need to:

1 Configure the RAD-Series Server for DHCP.

2 Set up realms for DHCP.

3 Define DHCP address pools for realms or specific users.

4 Configure the DHCP server to synchronize with the RAD-Series Server’s DHCP properties.

5 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

6 Stop and restart the AAA Server by running `radiusd.sh restart`

Configuring the RAD-Series Server for DHCP

See "DHCP Server Properties" on page 201 for parameters you can configure.

Setting Up Realms for DHCP

For each realm that will get IP address assignments via DHCP:

1 Change directory to `/<server configuration path>` (`/etc/opt/aaa/` by default).

2 In a text editor, add this line for each realm that is going to do DHCP in the `las.conf`.

```
Realm    Realm-name
```

Where ***Realm-name*** is the realm name of a realm requiring DHCP. This turns on session tracking which is needed to be able to return the IP address afer a session is closed.

3 Save and close `las.conf`.

4 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

5 Stop and restart the AAA Server by running `radiusd.sh restart`

Note: Enabling Session Tracking sets the number of individual concurrent sessions to the global Simultaneous Use value set in Session Properties under the Server Properties page. You can override this number by defining a Simultaneous-Use value for an individual user in the user profile.

Defining Address Pools for Specific Users

If the user profile is stored in local storage:

1 Change directory to `/<server configuration path>` (`/etc/opt/aaa/` by default).

2 In a text editor, open *`realm`*`.users` for *`realm`* users stored in a realm file or open `users` for users stored in the default users file.

3 Enter the reply item for the users: `Address-Pool = "name-of-pool"`

4 Save and close the opened file.

6 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

7 Stop and restart the AAA Server by running `radiusd.sh restart`

If the user profile is stored in an LDAP directory, add the reply-item attribute to your LDIF file:

```
aaaReply: Interlink:Address-Pool=name-of-pool
```

Configuring the DHCPv4 Server

Be sure the following properties on the DHCP server do not conflict with the RAD-Series Server’s DHCP properties:

- The DHCP server’s DHCP Lease value must be greater than the RAD-Series Server’s Session-Kill and Session-Clear values.
- The DHCP server must be configured to match the DHCP Send User Class setting configured on the RAD-Series Server.

Tunneling

The RAD-Series Server establishes tunnels, such as compulsory VPN tunnels, by returning standard RADIUS and vendor-specific tunneling attributes to the client device. See “Tunneling Properties” on page 205 for the parameter settings.

Establishing a Tunnel for a User

If the user profile is stored in local storage, you can add tunneling attributes.

1 Change directory to `/<server configuration path>` (`/etc/opt/aaa/` by default).

2 In a text editor, open *`realm`*`.users` for *`realm`* users stored in a realm file or open `users` for users stored in the default users file.

3 Enter the tunneling reply items one per line for the user:

```
Tunneling-Attribute = :Tag-no:Value,
```

For example:

```
user Password = “topsecret”,
   session-timeout = 43200,
   idle-timeout = 3600,
   Tunnel-Type = :0:PPTP,
   Tunnel-Medium-Type = :0:IPv4,
   Tunnel-Client-Endpoint = :0:192.168.127.1,
   . . .
```

4 Save and close the opened file.

8 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

9 Stop and restart the AAA Server by running `radiusd.sh restart`

If the user profile is stored in an LDAP directory, you can use an LDIF file to add the tunneling attributes to the profile as reply item attributes. Use the syntax:

```
aaaReply: Tunneling-Attribute = :Tag-no:Value
```

Tunneling Attributes

Tunneling attributes are returned as reply items in a RADIUS Access-Accept message. These tunneling attributes are supported by the RAD-Series Server:

Attribute	Type	Description
Tunnel-Medium-Type	tag-int	Indicates the transport medium to use when establishing the tunnel. See the `dictionary` file for the defined/supported values.
Tunnel-Type	tag-int	Indicates the tunneling protocol to use when establishing the tunnel. See the `dictionary` file for the defined/supported values.
Tunnel-Client-Endpoint	tag-str	Address of the client that initiated the tunnel.
Tunnel-Server-Endpoint	tag-str	Address of the server that provides the tunnel to the user.
Tunnel-Password	tag-str	Password for access to the machine specified by Tunnel-Server-Endpoint. This is not the password used for authentication.
Tunnel-Private-Group-ID	tag-str	Group identifier for a private session. Private groups may be used to associate a tunnel with a particular group of users, for example, to route unregistered IP addresses through a particular interface.
Tunnel-Assignment-ID	tag-str	Indicates what tunnel to use to provide the appropriate level of service. Data transfer for users that share the same assignment are multiplexed over a shared tunnel.
Tunnel-Preference	tag-int	When using tagged tunnel attributes, indicates each tunnel’s relative level of preference. Specified as an ordinal number: first, second, etc.
Tunnel-Client-Auth-ID	tag-str	Name used by the client during the authentication that occurs between Tunnel-Client-Endpoint and Tunnel-Client-Server.
Tunnel-Server-Auth-ID	tag-str	Name used by the server during the authentication that occurs between Tunnel-Client-Endpoint and Tunnel-Client-Server.

Tagged Tunneling Attributes

The AAA software supports tagged attributes that can be used to specify tunneling alternatives, in the event that the access device cannot establish the preferred tunnel configuration.

To specify a tagged attribute, use the syntax:

Tunneling-Attribute = :Tag-no:Value,

The order in which the access device should consider the tunnel alternatives is specified with the `Tunnel-Preference` attribute. For example, if Tag set 2 is actually the preferred tunnel, the entry would be:

Tunnel-Preference = :2:1,

Some access devices do not support tagged attributes. We recommend that when you return multiple tunnel definitions to a client, you should have at least one set of attributes that is untagged or tagged with a 0 value, so that there is a tunnel definition available to a client that does not support tags.

This example shows two sets of tagged values for the same tunneling attributes.

```
Tunnel-Type = :1:PPTP,
Tunnel-Medium-Type = :1:IPv4,
Tunnel-Client-Endpoint = :1:192.168.127.1,
Tunnel-Server-Endpoint = :1:192.155.111.1,
Tunnel-Password = :1:Michigan,
Tunnel-Private-Group-ID = :1:engineering,
Tunnel-Assignment-ID = :1:management,
Tunnel-Preference = :1:1,
Tunnel-Client-Auth-ID = :1:NET,
Tunnel-Server-Auth-ID = :1:Michigan,
Tunnel-Type = :0:L2TP,
Tunnel-Medium-Type = :0:IPv4,
Tunnel-Client-Endpoint = :0:192.168.127.1,
Tunnel-Server-Endpoint = :0:192.170.130.1,
Tunnel-Password = :0:California,
Tunnel-Private-Group-ID = :0:engineering,
Tunnel-Assignment-ID = :0:management,
Tunnel-Preference = :0:2,
Tunnel-Client-Auth-ID = :0:NET,
Tunnel-Server-Auth-ID = :0:California
```

Maintenance

These sections explains the RAD-Series Server functions for:

- Reading the server log file
- Reading accounting logs
- Reporting on active sessions
- Debugging the server

Viewing Server Log File

The RAD-Series Server log file contains a history of server messages generated in response to:

- Server starts/stops
- Internal errors
- Access-Requests and Accounting-Requests

Server logs are, by default, automatically compressed and stored each day in a different file in the server's log file directory (`/var/opt/aaa/logs` by default). Logs follow the file naming convention `logfile.`*yyyymmdd*.

To view server log file messages, use your favorite UNIX text tool like less, more, grep, etc.

Message Type	Description
Error/Alert/Critical message	Messages indicating a RAD-Series Server internal error, a serious problem with the configuration files or other severe issues. These should be corrected promptly as they may affect the server operation or ability to provide AAA services. These messages are displayed as message types Error "(E)", Critical "(C)", or Alert "(A)" in the physical logfile.
Warning/Notification message	Messages, flagged as "(W)" in the logfile, indicate a less serious problem, often produced when processing a received request, and can indicate a problem on the client's end. They should be corrected as they may impact the ability of some users to obtain AAA services, though the correction may well involve the NAS rather than the RAD-Series Server. Notification messages, flagged as "(N)" in the logfile, are indications of noteworthy events, such as the receipt of a HUP signal by the server.
Information	All messages that are not one of the above types. They are usually confirmation messages about server setup or usage. Informational messages, flagged as "(I)" in the logfile, are generated for start up messages and messages generated for each Accounting-Request message that is received. Other messages flagged as "(I)" in the logfile are not reported in this category but have their own category, such as authentication requests. By default, these types are not displayed.

Reporting User Accounting Records

The RAD-Series Server records session information in an active session table record. When the server receives an Accounting-Request to stop the session, this information is written to the server's accounting log file.

By default, accounting log files are written in MERIT standard format in `/var/opt/aaa/acct/session.`*`yyyy-mm-dd`*`.log`. See "Accounting Record Format" on page 73 for a description of log file information.

You can control the session logging behavior by changing when the server's Finite State Machine (FSM) calls the LOG action. See "Modifying Accounting Logging Behavior" on page 78.

Note: Not all wireless access points support RADIUS account logging. You should verify support with the access point vendor.

Viewing the Accounting Log

To view accounting log records, use your favorite UNIX text tool like less, more, grep, etc.

Accounting Record Format

In the default MERIT format, the first line of an accounting record contains 14 tab delimited fields that represent the user session information. If a value does not exist, NA appears as the value's placeholder.

The first line of an accounting record contains:

```
LAS-Start Session-State LCL-Time LAS-Duration LCL-Duration User-Name
   Authenticated-User-Name Session-ID Token Session-Timeout NAS/Port
   Service-Class Filter-ID Service-Type
```

After the first line, each A-V pair in the Accounting-Request-Stop message is listed, preceded by the characters "##."

Note: The default format is specified by the "log_v2_0" setting for the AATV parameter in the `log.config` file. Alternate formats, including Livingston Call Detail Records, may be specified

Accounting Attributes

These attributes appear on the first line of the MERIT accounting record:

Attribute	**Description**
LAS-Start (epoch)	Time when session started (in seconds), relative to 1/1/1970.
Session-State	State of the session when record was logged. Codes are: • `ACTIVE` • `FINISHED` • `UNCONFIRMED` • `MIA` • `DROPPED` • `COLLISION` • `No-Session` • `Not-Local`
LCL-Time (r-epoch)	Time when record was logged, relative to LAS-Start.
LAS-Duration (integer)	Duration of session (in seconds) according to the RAD-Series Server.
NAS-Duration (integer)	Duration of session (in seconds) according to the NAS, per the NAS's Acct-Session-Time attribute.
User-Name (special)	User-Id "@" User-Realm from RADIUS User-Name attribute. If realm uses a tunneled authentication method, this field will show the outer user identity.
Authenticated-User-Name (special)	If realm uses a tunneled authentication method, this field will show the authenticated Inner-Identity value. Otherwise, always shows the RADIUS User-Name value. You must enable session tracking for the outer realm to see this value. Doing so will also limit each user in this realm to the number of concurrent sessions you have defined with the Simultaneous-Use parameter of `las.conf`.
Session-ID (string)	Session ID, as assigned by the RAD-Series Server (is unique to each session).
Token (string)	Token pool name for the first generic tokenpool used by this session, else NA.
Session-Timeout (integer)	Session Time limit.
NAS/Port (special)	NAS-Identifier "/" NAS-Port-Number.
An obsolete field	Always NA.
Filter-ID (string)	Filter name associated with session.

Attribute	Description
Service-Type (special)	May be: • FRAMED/PPP • FRAMED/PPP/ip-addr • FRAMED/SLIP • FRAMED/SLIP/ip-addr • LOGIN • LOGIN/TELNET • LOGIN/TELNET/PROMPT • LOGIN/TELNET/PROMPT • LOGIN/TELNET/ip-addr • LOGIN/TELNET/ip-addr/tcp-port

Accounting Record Extensions

Standard Accounting RFC extensions may appear in a RAD-Series Server accounting record in addition to the basic session information. These extensions are the A-V pairs sent from the client in the Accounting-Request-Stop message. See the `dictionary` file for a list of valid A-V pairs.

Access Device Values

Vendor-specific attributes (VSAs) related to the access device may appear among the accounting record extensions. These represent attribute values that describe the access device used for authentication and authorization. Valid attributes are defined in the `dictionary` and `dictionary.*` files. You may extend the VSA list by adding the `vendor` to the vendors file and the attributes and their values to the `dictionary.custom` file.

Example Accounting Record

```
1353084227 FINISHED 4 4 NA t2002l@pap.com t2002l@pap.com
'AAA.50a66d43.0001' NA NA t32.iln.com/2002 NA NA NA
## User-Name:0='t2002l@pap.com'
## NAS-IP-Address:2=192.168.3.32
## Acct-Input-Octets:1=16
## Acct-Output-Octets:1=32
## Acct-Input-Packets:1=2
## Acct-Output-Packets:1=4
## Acct-Authentic:1=RADIUS Acct-Delay-Time:1=0
## Acct-Session-Id:0='t32-2-50a66d43'
## Acct-Status-Type:1=Stop
## Session-Tracking-Status:1=Have-Session
## Session-Start-Time:3='2016-11-16 16:43:47'
## Session-End-Time:3='2016-11-16 16:43:51'
```

Modifying the Accounting Log

Certain features of the standard RAD-Series Server accounting log format can be reconfigured through the `log.config` file.

Writing CDR Accounting Records

You can configure the server to write Accounting log records in the Livingston Call Detail Record (CDR) format, rather than, or in addition to, the default MERIT format.

If you change radius.fsm to call ACCT rather than LOG for each logged message event, logs will be stored in an alternate directory (`/var/opt/aaa/radacct`).

1 In a text editor, open the `log.config` file found in the server’s configuration file directory. Locate the following line, which should be found near the beginning of the file:

```
aatv log_v2_0
```

2 Change `aatv log_v2_0` to `aatv log_acct`

3 Save and close the file.

4 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

5 Stop and restart the AAA Server by running `radiusd.sh restart`

Changing the Accounting Log Filename

1 In a text editor, open the `log.config` file found in the server’s configuration file directory. Locate the following line, which should be found near the beginning of the file:

```
filename session.%Y-%m-%d.log
```

2 Change `session.%Y-%m-%d.log` to the filename syntax you wish to use.

3 Save and close the file.

4 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

5 Stop and restart the AAA Server by running `radiusd.sh restart`

Changing the Accounting Log Rollover Interval

The log rollover interval specifies how often a new log file is created to store accounting records. The interval is determined by the finest unit of time in the timestamp portion of the filename.

If `gzip` is in your UNIX path, log files can be automatically compressed when they rollover by adding a `on-endfile “gzip -9 !”` to the `log.config` file.

1 In a text editor, open the `log.config` file found in the server's configuration file directory. Locate the following line, which should be found near the beginning of the file:

```
filename session.%Y-%m-%d.log
```

2 Rename the file to reflect the rollover interval. For example, for hourly rollover:

```
%Y-%m-%d-%H
```

3 Save and close the file.

4 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

5 Stop and restart the AAA Server by running `radiusd.sh restart`

Modifying Accounting Logging Behavior

You can change the RAD-Series Server's logging behavior by modifying the server's `radius.fsm` file.

All RADIUS accounting message types marked LOG are logged by the server. To log any type of accounting messages, simply change the action to LOG for the event handler that corresponds to the message. These should always take the next action ReplyHold.

Interim Accounting Logging

To indicate that a session is still active, a client may send an Accounting-Alive (Accounting-Interim-Update) message at regular intervals during the session. To generate logs when the server receives this message, rather than wait for Accounting-Stop:

1 In a text editor, open the file `radius.fsm` (found in `/etc/opt/aaa` by default).

2 Find the lines:

```
AcctLog:
      *.*.ACCT_START          ReplyPrep         ReplyPolicyHold
      *.*.ACCT_STOP           LOG               ReplyHold
      *.*.ACCT_ALIVE          ReplyPrep         ReplyPolicyHold
```

3 Change:

```
     *.*.ACCT_ALIVE           ReplyPrep         ReplyPolicyHold
```

To:

```
     *.*.ACCT_ALIVE           LOG               ReplyHold
```

4 To turn **on** logging for any other message types:

- Change `ReplyPrep` to `LOG`
- Change `ReplyPolicyHold` to `ReplyHold`.

To turn **off** logging, reverse the settings.

5 Save and exit `radius.fsm`.

6 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

7 Stop and restart the AAA Server by running `radiusd.sh restart`

Proxy Accounting Messages

Normally, the server logs all accounting messages locally. To log messages on a central server, follow the steps in "Proxying Accounting Requests to a Central Server" on page 33 to change the .fsm settings.

Reporting Server Statistics

1 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

2 Run `radcheck.sh localhost`

In the output you can see the total number of:

- Accounting-Requests
- Authentication-Requests
- Authentication-Successes
- Authentication-Failures

See “Reporting Server Status” on page 25 for how to read the output.

Reporting Active Server Sessions

If the RAD-Series Server is tracking sessions for a given realm, a session table entry is set up when the user is authenticated and the server returns an Access-Accept. The session entry remains active until the session is terminated for some reason, such as the receipt of an Accounting-Stop request from the access device (the normal case), or if the server itself terminates the session due to, say, a session collision or an Interim-Accounting timeout. Accounting messages are logged to the accounting logfile when received. Which accounting messages are logged depends on the FSM you are using. The default FSM only logs Accounting-Request-Stop messages. The `logall.fsm` FSM will log all the accounting messages.

Note: Not all wireless access points support RADIUS session-based account logging. Verify support with the access point vendor.

Viewing Active Sessions

To view currently active sessions, use the `sesstab` utility. See “sesstab” utility on page 99 for a desction of its options. To use the utility:

1. Change directory to `/`*`<server utilities path>`* (`/opt/aaa/utl` by default).
2. Run `sesstab.sh` *`<options>`*

Stopping Active Sessions

To clear sessions that were terminated on the access device but are maintained as active by the RAD-Series, see “Stopping Sessions Using radpwtst” utility on page 94.

Overview of the Server’s Session Management

If the RAD-Series Server is managing a session for a particular user, the server will allocate a internal data structure, called a session entry, in which information about the user’s session is held and updated. The session entry structure information includes session identification parameters (`User-Name, NAS-Identifier, NAS-Port, session id, Acct-Session-Id`, etc) as well as the attributes from the last received accounting request.

Session States

A session will be in one of the following states: `PENDING, ACTIVE, FINISHED, UNCONFIRMED, MIA, DROPPED, COLLISION`, or `RELEASING`. These states are defined in the dictionary.Interlink’s Session-State attribute.

A session will normally transition from `PENDING` to `ACTIVE` to `FINISHED`, after which the session ends and the session entry is deallocated.

The normal session states, and the events which cause a transition from one state to another are these:

State	Description
PENDING	This is the first state for a session. When the server successfully processes an Access-Request and returns an Access-Accept, the session entry is created and the session is placed in `PENDING` state, pending the arrival of the expected Accounting-Start message.
ACTIVE	This is the steady state in which a session normally resides. When the expected Accounting-Start message is received for a session in `PENDING` state, the session transitions into `ACTIVE` state. The session will normally stay in `ACTIVE` state until an Accounting-Stop message is received, signaling the end of the access device's session. If Interim-Accounting messages are received for an `ACTIVE` session, the session remains in `ACTIVE` state.
FINISHED	When an Accounting-Stop is received, an accounting record is normally written, and the `ACTIVE` session will transition into `FINISHED` state. The session entry will remain in `FINISHED` state for a configurable time (`Session-Finished-Timeout`), and then the session entry will be deleted.

The arrival of unexpected messages or the failure of expected messages to arrive in a timely manner may cause a session to transition into `UNCONFIRMED, EXPIRED, MIA, DROPPED, COLLISION`, or `RELEASING` state:

State	Description
UNCONFIRMED	If a session in `PENDING` state does not receive the expected Accounting-Start message within a configurable time (Session-Pending-Timeout), the session enters `UNCONFIRMED` state. If an `UNCONFIRMED` session does not receive an Accounting message within a configurable time (`Session-Unconfirmed-Timeout`), the session is terminated and the session entry is deallocated. If an Accounting message is received before the `Session-Unconfirmed-Timeout` expires, the session will transition into `ACTIVE` (for an a received Acct-Start or Interim-Acct) or `FINISHED` (for a received Acct-Stop).
COLLISION	If Collision-Checking is enabled and the server receives a new Access-Request whose NAS-Identifier and NAS-Port match those of an existing session which is in `PENDING` or `ACTIVE` or `UNCONFIRMED` or `MIA` state, the existing session is placed in `COLLISION` state. The session will remain in `COLLISION` state for a configurable period of time (`Session-Collision-Timeout`), after which the session is terminated and the session entry is deallocated. The receipt of an Acct-Stop before the Session-Collision-Timeout expires will transition the session into `FINISHED` state.
DROPPED	If the server receives an Accounting-Request from an access device with an Acct-Status-Type of Accounting-On or Accounting-Off, this indicates that all sessions from this access device should be terminated. The server will transition any sessions for this device which are in `ACTIVE` or `PENDING` or `UNCONFIRMED` or `MIA` state, into `DROPPED` state. A session will remain in `DROPPED` state for a configurable period of time (`Session-Dropped-Timeout`), after which the session is terminated and the session entry is deallocated. The receipt of an Acct-Stop before the Session-Dropped-Timeout expires will transition the session into `FINISHED` state.
MIA	Once the server receives an Interim-Accounting message for session in `ACTIVE` state, the server will expect to receive further Interim-Accounting messages periodically. The Interim-Accounting messages act as a keepalive for the session. If a subsequent expected Interim-Accounting message does not arrive in a timely manner, the session is placed into `MIA` state. The session will remain in `MIA` state for a configurable period of time (`Session-MIA-Timeout`), after which the session is terminated and the session entry is deallocated. The receipt of an Interim-Accounting message before the `Session-MIA-Timeout` expires will transition the session back into `ACTIVE` state. The receipt of an Acct-Stop before `the Session-MIA-Timeout` expires will transition the session into `FINISHED` state.
RELEASING	If a `PENDING` session entry has been created for a newly authenticated session during the processing of an Access-Request, but some subsequent FSM or POLICY step causes the server to reject the Access-Request, the session will transition into `RELEASING` state. A session entry in `RELEASING` state will be quickly deallocated.

Concurrent Session Limits

A user session which is in `ACTIVE` state, or in a state which could become `ACTIVE` (`PENDING`, `UNCONFIRMED`, or `MIA`), will count towards the server's overall license limit, and will count towards that user's `Simultaneous-Use` limit.

Accounting records generated by the server

If a session terminates for which the server has not received an Accounting-Stop from the access device, the server will internally generate an Acct-Stop message for this session based on information the server has saved in the session entry. This happens when a session expires in the `UNCONFIRMED`, `COLLISION`, `MIA`, or `DROPPED` states. The server-generated Accounting-Stop will be logged in the accounting log file, available for network management, auditing, or billing. A server-generated accounting record can be recognized by its session state of `UNCONFIRMED`, `COLLISION`, `MIA`, or `DROPPED`.

The server's algorithm for measuring Interim-Accounting intervals

The `Session-MIA-Timeout` is the time in seconds the RAD-Series Server awaits the next Interim-Accounting message before moving an `ACTIVE` session into the `MIA` state, or before removing a session already in the `MIA` state. The default of `Session-MIA-Timeout` is 0, a special value that indicates the RAD-Series Server will measure the time interval between received Interim-Acct messages, and use that measured value to time out subsequent Interim-Acct messages. This section describes the server's algorithm by which this interval is determined.

1. The server measures the time T, which is the time between the receipt of the Accounting-Start and the receipt of the first Interim-Accounting message. The value of T, plus a small configurable *`delta`* value, will represent the maximum time the server will wait for the next expected Interim-Accounting message. Note—If the server receives an Interim-Accounting while in `PENDING` state, this means either the Accounting-Start was lost, or the Interim-Accounting has crossed paths with the Accounting-Start. In this case, the server will calculate T as the time between the sending of the Access-Accept and the receipt of the first Interim- Accounting message.
2. There is no configuration parameter to override the measured value of T. The server always does this calculation when an Interim-Accounting is received.
3. The configurable *`delta`* value is named `Acct-Interim-Grace-Period`, and its default value is 15 seconds. The *`delta`* value provides for small variations in the delivery interval of Interim-Accounting messages, to allow for network transit time, retransmissions, etc.
4. When the 2nd, 3rd, etc Interim-Accounting message is received, the value of T is recalculated. The recalculation of T is as follows: T is set to the minimum of [a] the previous value of T, and [b] the measured time period between the current time and the time when the previous Interim-Accounting was received, and [c] the configured value of `Minimum-Acct-Interim-Interval` if this value is greater than zero. The idea is that the server will

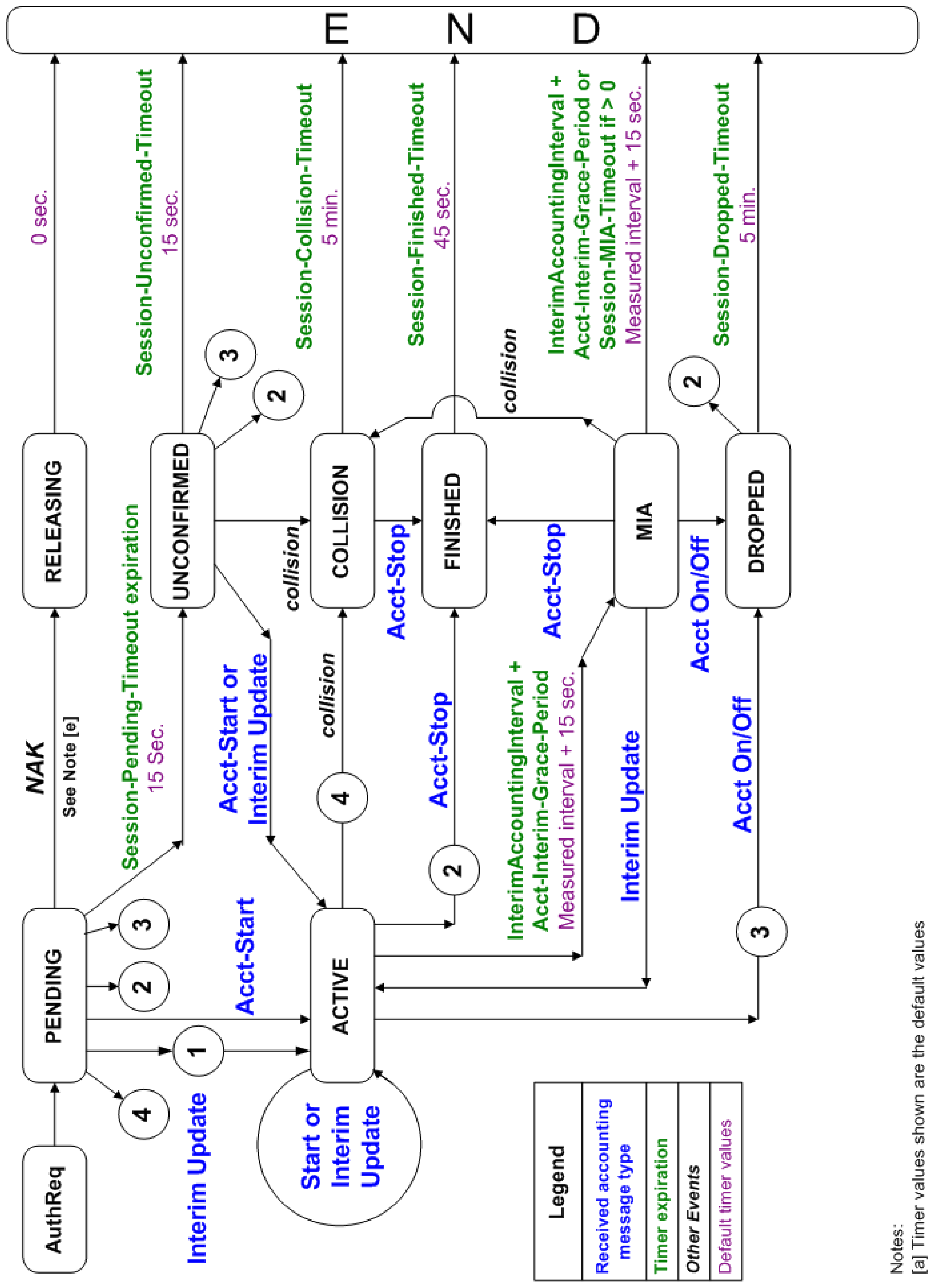
shorten the expected time T if the initial Interim-Accounting message was lost, i.e. the first measurement of T had inadvertently measured the time between the Accounting-Start and the 2nd Interim-Accounting message.

5 If the session is in the `ACTIVE` state and if time period T+*`delta`* expires without the receipt of an Interim-Accounting message, the session will move into `MIA` state and stay in the `MIA` state for a (default) maximum of (T+*`delta`*) seconds. This allows for two consecutive Interim- Accounting messages to be lost before the session is declared finished. The (T+*`delta`*) is the default value for wait within the `MIA` state. There is a configurable `Session-MIA-Timeout` value which overrides this measured (T+*`delta`*), for customers who may prefer a shorter or longer expiration of the `MIA` state. The `Session-MIA-Timeout` parameter contains a non-negative value representing the maximum number of seconds to stay in `MIA` state. The default value is zero, which is a special value which means "use (T+*`delta`*) seconds as the timeout value for the `MIA` state".

6 If an Interim-Accounting message is received while the session is in `MIA` state, the session transitions back into `ACTIVE` state.

7 If the state timer expires while the session is in `MIA` state, the session terminates: the session entry is removed from the session table, resources are freed (e.g. assigned IP addresses are returned to the DHCP server's named address pool), an accounting message is generated within the server and logged, and the session entry is deallocated.

Session Flow Diagram



Notes:
[a] Timer values shown are the default values
[b] only ACTIVE and PENDING states count towards a user's Simultaneous-Use
[c] only ACTIVE and UNCONFIRMED states count towards the licensed concurrent session limits
[d] Collision checking only happens in PENDING, ACTIVE, UNCONFIRMED, MIA, and EXPIRED states if Session-Collision-Checking parameter is enabled
[e] ***NAK*** event occurs if: 1) can't get IP address from DHCP server, 2) some policy step fails, or 3) can't get token from generic token pool

Server Programs

radiusd

`radiusd` is the main RAD-Series Server program, used to handle Access-Requests and Accounting-Requests from RADIUS clients. Authentication and accounting requests come to `radiusd` in the form of UDP packets conforming to the RADIUS protocol.

Message processing is based upon a finite state machine that `radiusd` loads into memory when it is first started. You can configure which finite state machine the RAD-Series Server loads upon startup with the `‘-f FSM-file’` option, but it is static after server startup.

`radiusd` runs as a daemon that you can start using the command line or through an inetd service.

To start `radiusd` from the command line:

1 Change directory to `/`*`<server utilities path>`* (`/opt/aaa/utl` by default).

2 Run `radiusd.sh start`

Synopsis

```
Radiusd  -c <Working-directory> -d <Config-directory>
-da <AATV-directory> -dd <Data-directory>
-ddebug <Debugfile-directory> -di <IPC-directory>
-dl <Logfile-directory> -dm <Accounting-directory>
-a <Livingston-directory> -dr <Run-directory>
-debughistory { on | off } -errorlog { enabled | disabled }
-f <FSM> -g { logfile|syslog|stderr|syslog+logfile|syslog+stderr }
-h -l <Log-format> -n -p <Authentication-port> -q <Accounting-port>
-pp <Authentication-relay-port> -qq <Accounting-relay-port> -s
-syslogfacility <facility> -syslogtimestamp { on | off }
-add-hostname-to-logfile-header { yes | no } -v -z -x
```

Options

You can start radiusd with any of the following options to override built-in defaults

Option	Description
-a <*Livingston-directory*>	Location of Livingston Call Detail Records (when used as an alternative to the default MERIT format). The default is `/var/opt/aaa/logs/radacct`.
-add-hostname-to-logfile-header <yes/no >	If enabled (Yes), the RAD-Series Server will insert the server's hostname in the logfile lines, following the timestamp. The default is `no`.
-c <*Working-directory*>	New current working directory. If you specify this directory, you can use paths relative to the working directory for subsequent `radiusd` directory options. It defaults to the CWD at the time radiusd is started.
-d <*Config-directory*>	Location of configuration files. The default is `/etc/opt/aaa`.
-da <*AATV-directory*>	Location of binary AATVs. The deafault is `/opt/aaa/aatv`.
-dd <*Data-directory*>	Location of active session information stored in the session.las file. The `sesstab` utility requires that this option be defined if the server has been installed in a nondefault location. The default is `/opt/aaa/data.`
-ddebug <*Debugfile-directory*>	Server debug file (`radius.debug`) directory. The default is `/var/opt/aaa/logs`.
-debughistory { on | off }	Enables/Disables debug history. Debug history generates `radius.debug%Y%m%d` formatted file names and if rolled over because of size, a "`partNNNN`" string will be added to the end of the file name. The default is `OFF`.
-di <*IPC-directory*>	Location of shared memory operation files. The default is `/var/opt/aaa/ipc`.
-dl <*Logfile-directory*>	Server log file directory. The default is `/var/opt/aaa/logs`.
-dm <*Accounting-directory*>	Location of accounting session records in MERIT format. The default is `/var/opt/aaa/logs/acct`.
-dr <*Run-directory*>	Location of the file where the server's process ID is stored. The default is `/var/opt/aaa/run.`
-errorlog { enabled | disabled }	Indicates if server will write the '`errorlog`' file. The default is `enabled`.
-f <*FSM*>	Alternate FSM file to load upon startup, instead of the default `radius.fsm` file.

Option	Description
-g {logfile | syslog | stderr | logfile+syslog | stderr+syslog}	Where server events are logged. Enter option followed by `logfile` (server logfile), `syslog` (system logfile), or `stderr`. You can also combine `logfile+syslog` and `stderr+syslog` to enable logging to two locations at the same time. The order of the two items is not important.
-h	Display the help syntax.
-l <*Logfile-name*>	Server log file name. The default is `logfile.%Y%m%d`.
-n	Start new session table for LAS upon server startup.
-p <*Authentication-port*>	Default port number to listen for authentication requests on. See "Managing RADIUS Listen Sockets" on page 265 for a complete description.
-pp <*Authentication-relay-port*>	Default port number to relay authentication requests to, if the remote server uses a port other than UDP standard 1812 for authentication requests.
-q <*Accounting-port*>	Default port number to listen for accounting requests on. See "Managing RADIUS Listen Sockets" on page 265 for a complete description.
-qq <*Accounting-relay-port*>	Default port number to relay accounting requests to.
-s	Run in single process (non-spawning) mode.
-syslogfacility <*facility*>	If logging to syslog has been enabled and if "`-syslogfacility`" is specified, all syslog messages are directed to the specified facility. if not specified, then some syslog messages are directed to the auth facility and others are directed to the daemon facility. The supported facility values are: `local0, local1, local2, local3, local4, local5, local6, local7, kern, user, mail, daemon, auth, syslog, lpr, news, uucp, cron`
-syslogtimestamp {on|off}	This parameter controls whether to add the logfile's timestamp+loglevel information to the syslog lines. The default is `ON`.
-v	Display `radiusd` version information and exit.
-x	Add to debug flag value, repeat the –x from one to four times to enable 1 to 4 levels of debugging. These switches should follow all other switches.
-z	Empty (zap) the log file upon startup.

radiusd.sh

`radiusd.sh` is a script that can be used to start the RAD-Series Server at boot time or to manually start, stop, restart or reload the server.

Synopsis

```
radiusd.sh start|stop|restart|reload|status
```

Arguments

The ‘`start`’ argument will start the RAD-Series Server if it is not already running.

The ‘`stop`’ argument will stop the RAD-Series Server if it is running.

The ‘`restart`’ argument will stop the RAD-Series Server if it is running and then start it back up.

The ‘`reload`’ argument will HUP the RAD-Series Server if it is running so that it can re-read its configuration files.

The ‘`status`’ argument will report the current status of the RAD-Series Server.

radcheck

`radcheck` determines whether a given RAD-Series Server is operational. radcheck may be invoked from any host, not just those registered in the clients file, although more information is returned to those so registered.

To start `radcheck` from the command line:

1 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

2 Run `radcheck.sh localhost`

See "Reporting Server Status" on page 25 for a description of radcheck output.

Synopsis

```
radcheck -all -d <Directory> -h -ipv6 <on/off> -p <UDP-port>
-r <Retries> -t <Timeout> -v -writesess -x <ServerName>
```

Arguments

`<ServerName>` can be an IPv4 address, IPv6 address or fully-qualified domain name which maps to an IP address on which the RAD-Series Server is listening.

Options

Option	Description
-all	Display all of the server's statistics. If not specified, the server will display the statistics as configured by the "radcheck" parameters (see "Server Tracking Properties" on page 178 for the description of the "radcheck" parameter)
-d <Directory>	The directory containing the RAD-Series Server dictionary and clients files, instead of the default `/etc/opt/aaa` directory. If no -d option is given, radcheck will look in the default location. An error is displayed if the configuration files cannot be found.
-h	Display the radcheck syntax.
-ipv6 <on/off>	Enable/Disable IPv6 communications. If value is '`on`' then the status request will be sent using IPv6. If the value is '`off`' then the request will be sent using IPv4. The default is `off`.
-p <UDP-port>	A UDP port on which the RAD-Series Server is listening for authentication requests. The default value is 1812.
-r <Retries>	Maximum number of retries instead of the default of 10.
-t <Timeout>	Alternate timeout value (in seconds) instead of the default of 3.
-v	Display `radcheck` version information.

Option	Description
-writesess	Force the RAD-Series Server to checkpoint the session table when the `radcheck` request is processed. This is of benefit to users who will be doing a subsequent `sesstab` inquiry and wants the most-current session information.
-x	Add to debug flag value.

radpwtst

`radpwtst` is a test client program. It generates RADIUS packets to perform a password authentication on a specified user using the current RAD-Series Server configuration.

If the authentication succeeds, `radpwtst` displays "`authentication OK`" on standard output. Otherwise, `radpwtst` displays "`User-Name authentication failed.`"

radpwtst can also send accounting requests. When a response is received, radpwtst displays "`Accounting Response received.`"

To run `radpwtst` from the command line:

1 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

2 Run **`radpwtst.sh`** `<options>` <***User-Name***>

Synopsis

```
radpwtst -a <ACKs> -c <Code> -d <Directory> -f <File> -g <Group> -h
-i <client-id> -ipv6 <on/off> -l <NAS-Port> -n -p <UDP-port>
-r <Retries> -s <Servername> -secret <secret> -t <Timeout> -u <Type>
-v -w <Password> -x -z <Challenge-password> -:<attribute=value>
<User-Name>
```

Arguments

<*User-Name*> User-Name to send with the test request. If no realm is specified, the server looks for the user in the NULL realm data store. By default, the NULL realm data store is the users file distributed with the server.

Options

Option	Description
-a <*ACKS*>	Number of responses to ignore. Used to test possible retransmission problems.
-c <*Code*>	RADIUS packet type codes from the following list: 1: Access-Request 4: Accounting-Request n: Packet code n, 0 <= n <=255
-d <*Directory*>	The directory containing the server's `clients`, `dictionary`, and `vendors` files, instead of the `default /etc/opt/aaa` directory. If no - d option is given, radpwtst will look for the default location. An error will be displayed if the configuration files cannot be found.

Option	Description
-f <*File*>	A prefix for a file in the users file format. The name of this file is assumed to be `<File>.users` and is found in the configuration directory. This file contains check-items and reply- items to send with `radpwtst`. They are grouped into pseudo- users having names that may be specified by the following `-g` option.
-g <*Group*>	An arbitrary “pseudo-user” named group in the file specified by the above `-f` option. This file contains arbitrary check-items and reply-items (see users for more information) grouped by these pseudo-user names. If no `-g` option is given, the DEFAULT entry (if one is present) will be used. In this way, attribute-value pairs may be communicated to remote RADIUS servers.
-h	Displays the `radpwtst` help syntax.
-i <*client-id*>	The value can be a DNS name, an IPv4 or IPv6 address. If it is a DNS name, it is mapped into an IPv4 or IPv6 address. The final IP address, as determined by `-i`, is sent as the NAS-IP-Address attribute or the NAS-IPv6-Address attribute. And “`-i -1`” (client-id of -1), the special value of -1 tells `radpwtst` to NOT SEND a NAS-IP-Address. This might be of use to someone that wants to instead send NAS-Identifier and suppress the NAS-IP-Address. The default is the originating machine's IP address.
-ipv6 <on/off>	If the value is ‘`on`’ then the request will be sent using IPv6. If the value is ‘`off`’ then the request will be sent using IPv4. The default is `Off`.
-l <*NAS-Port*>	The NAS port number to send instead of the default of 1. And if “`-l -1`” (NAS-Port of -1), the special value -1 tells `radpwtst` to NOT SEND a NAS-Port attribute. This might be of use to someone who doesn’t want to send NAS-Port at all, or wants to send NAS-Port-Type instead.
-n	Forces the Authentication-Only value to be used in the Service-Type A-V pair.
-p <*UDP-port*>	An alternate UDP port number. For an Access-Request the default UDP port number is 1812 or the number specified in the `/etc/services` file for `radiusd`. For an Accounting-Request the default port number is 1813 or the number specified in the `/etc/ services` file for radacct.
-r <*Retries*>	Maximum number of retries instead of the default of 10.
-s <*Servername*>	An alternate server instead of the built-in default of radius.localdomain.

Option	Description
-secret <*secret*>	Shared secret (no default). If not specified, `radpwtst` will look in the `clients` file for the shared secret.
-t <*Timeout*>	Timeout value (in seconds) instead of the default of 3.

-u <*Service-Type*>	This specifies the Service-Type to send instead of using the default auth value. Note, that the default auth value will fail if no password (or an empty password) is included in the Access-Request (default or -c 1) produced by `radpwtst`. RADIUS server requires a valid (nonempty) password be provided in Access- Request packets where the Service-Type is Authenticate-Only.
	Valid Service-Type values are:
	`admin`
	`auth`
	`dumb`
	`exec`
	`outbound`
	`ppp`
	`slip`
	`dbadmin`
	`dbdumb`
	`dbppp`
	`dbslip`
	Note: db stands for dial back in the last four types.

-v	Displays the `radpwtst` version.
-w <*Password*>	Password to use, instead of being prompted for one.
-x	Increases the debug level which displays additional debug output and account logging on the screen.
-z <*Challenge-password*>	Challenge password to use in response to an Access-Challenge, instead of being prompted for one.
-:<*attribute=value*>	One or more A-V pairs to add to the test request.

Stopping Sessions Using radpwtst

You can clear sessions with `radpwtst`. To clear a session, you must know its `Session-Id` (`Class` attribute value), which you can determine by running `sesstab`. The following command line will manually stop a session by sending an Accounting-Request stop message:

```
radpwtst -c 4 -s <Server FQDN>|<Server IP-Address> -i <NAS-IP-address>
-l <Port> -:Acct-Status-Type=Stop -p <Server auth listen port>
-:Class="<class-value>"
```

radsignal

The `radsignal` utility performs 3 functions:

- Turns debugging on and off or sets the level of debug output, while the RAD-Series Server is running. This function replaces `raddbginc`.
- Initiate server logfile rollover.
- Initiate accounting stream file rollover.

To run `radsignal` from the command line:

1. Change directory to /<*server utilities path*> (`/opt/aaa/utl` by default).
2. Run **`radsignal.sh`** ***<pid>*** ***<command>*** ***<arguments>***

Synopsis

```
radsignal -h -v
radsignal -di <IPC-directory> <pid> debug <delta>
radsignal -di <IPC-directory> <pid> roll logfile
radsignal -di <IPC-directory> <pid> roll stream <stream-name>
```

Arguments

<*pid*> The process ID of `radiusd` (the RAD-Series Server program). You can determine this by running:

```
ps -eaf | grep radiusd
```

<*delta*> The number of debug levels to **increment** from the current level (not the number of the level you want). The debug level will not increase past level 4. 0 turns debugging off. All debug output is sent to the `radius.debug` file in `/var/opt/aaa/logs` or in an alternate location specified by the `radiusd -dl` option.

<*stream-name*> The name of the accounting stream to roll. If one is not specified then the default stream (`*default*`) is used.

`debug, roll logfile`, and `roll stream` are keywords that identify the invocation desired and are described below after the options.

Options

Option	Description
-di <*IPC-directory*>	IPC directory path; required if `radiusd` installed in a non-default location. `<IPC-directory>` must match the path given by the `radiusd -di` option. The default is "`/var/opt/aaa/ipc`"
-h	Displays the `radsignal` syntax and can not be combined with other options.
-v	Displays the `radsignal` version and can not be combined with other options.

DESCRIPTION `debug`

This form of invocation specifies that you wish to change the level of debug output generated by `radiusd`. You can use the radsignal command to turn debugging on and off or set the level of output while the RAD-Series Server is running using `radsignal`. Debugging output by the server can also be turned on when starting the server at a specified level of output. All of the debug output is sent to the `radius.debug` file, in the default directory or in an alternate location specified by the `radiusd -dl` option.

Debug Levels

When starting `radiusd`, you can turn on debug output and set the level of output with the `-x` option. Each instance of the `-x` option at the command line will increase the debug level by one up to level 4. After the server is started, you continue to control the level of debugging output with the `radsignal` command. Each debug level provides the information from the previous levels, plus its own. The higher the number, the more detail.

Level	Description
0	No debugging (default)
1	Brief trace
2	High-level FSM output, some function tracing, A-V pairs, etc.
3	Full function tracing
4	Low-level FSM and configuration file output

DESCRIPTION `roll logfile`

This form of invocation specifies that you wish `radiusd` to immediately roll the logfile. "rolling" the logfile consists of closing the current logfile and opening a new logfile. If, for example, the current logfile is named `'logfile.20160209'` then it will be renamed to `'logfile_part01.20160209'` and the current logfile will be called `'logfile_part02.20160209'`. This is the same as what is done if the logfile reaches the maximum configured file size. Each subsequent request to roll the logfile will increment the 2-digit "part number" up to a maximum of 99 and will grow to 3-digit or larger "part numbers" as required.

The file name modification algorithm is to insert "`_part`**`NN`**" into the filename preceding the last dot of the file name. The intent is to preserve the filename extension, in case the filename extension is used by the server administrator.

DESCRIPTION `roll stream`

This form of invocation specifies that you wish `radiusd` to immediately roll the accounting stream specified by <*`stream-name`*>. "rolling" the accounting stream consists of closing the current file used for that stream and opening a new file for the stream. If, for example, the current accounting logfile for the stream is named '`session.2016-02-09.log`' then it will be renamed to '`session.2016-02-09_part01.log`' and the current accounting logfile will be called '`session.2016-02-09_part02.log`'. This is the same as what is done if the accounting logfile reaches the maximum configured file size. Each subsequent request to roll the accounting stream will increment the 2-digit “part number” up to a maximum of 99 and will grow to 3-digit or larger “part numbers” as required.

The file name modification algorithm is the same as roll logfile above.

saltencrypt

vernxxx

The `saltencrypt` utility allow you to generate an encrypted version of a clear text password for use as an encrypted password for the administrator in the LDAP directories:

To run `radsignal` from the command line:

1 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

2 Run **`saltencrypt.sh`** ***`<salt>`*** ***`<secret1>`*** ***`<secret2>`*** ***`<password>`***

Synopsis

`saltencrypt` **`<salt> <secret1> <secret2> <password>`**

Arguments

<*`salt`*> The 4 hex character salt to use during encryption.

<*`secret1`*> The first of two secrets which must match **`Secret1`** in the ProLDAP configuration. It can be any string from 1 to 255 characters long. See "ProLDAP Connection Properties" on page 185.

<*`secret2`*> The second of two secrets which must match **`Secret2`** in the ProLDAP configuration. It can be any string from 1 to 255 characters long. See "ProLDAP Connection Properties" on page 185.

<*`password`*> The password you want to encrypt. It can be from 1 to 128 characters long.

sesstab

The `sesstab` utility reads binary `session.las` files and prints out specific session information to standard output. Using various command line options, it is possible to create flexible and powerful search criteria to obtain information from the `session.las` file.
Note that options `-a, -i, -n, -p, -r,` and `-s` can be repeated to add additional criteria. For example, "`-a fred@co.com -a joe@bp.com`" will report sessions for both users.

The information for each selected session is, by default, printed one line per user:

<Access-Id> <State> <Session-Id> <NAS-Identifier> <NAS-Port> <Token-Name> <Date> <Start-Time> <Connect-Time>

<Access-Id> is the `Authenticated-User-Name` i.e usually the `User-Name` but can be the `Inner-Identity` for tunneled users.

To run `sesstab` from the command line:

1 Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).

2 Run **`sesstab.sh <options>`**

Note: The session.las file is a binary snapshot of the sessions currently tracked by the RAD-Series Server. Completed sessions remain in session.las for another 45 seconds by default and are therefore available for sesstab to report on. The session.las file is updated every 5 minutes by default. See the "`-writesess`" option of "`radcheck`" on page 90, "las.conf" on page 233 and the `Session-Table-Update-Interval` in "LAS Session Configuration" on page 233, for ways to change this interval.

Synopsis

```
sesstab -a <username> -d <config_dir> -dd <sess_dir> -f -f -h
-i <userid> -n <nasid> -p <port> -r <realm> -s <state>
-t <Token-pool> -v <file> <file> <...>
```

Arguments

`<file>` One or more files that may be used as an alternate to the `session.las` file.

Options

Option	Description
-a <*username*>	Select only sessions with specified user name(s) in "name@realm" format.

Option	Description
-d <config_dir>	The directory containing the RAD-Series Server dictionary and vendors files. If no -d option is given, `sesstab` will look first for a directory `../aaa` and if none is found, use the default `/etc/opt/aaa` location. An error will be displayed if neither directory can be used to locate the various server configuration files.
-dd <*sess_dir*>	Directory where the session file(s) are located. The default is '`/var/opt/aaa/data`'
-f	Display sessions in detailed (multi-line) format.
-f -f	Display sessions in detailed (multi-line) format, plus attributes from last saved accounting message.
-h	Displays the `sesstab` syntax.
-i <*userid*>	Select only sessions with specified user name(s) in “name” format.
-n <*nasid*>	Select only sessions with the specified NAS-Identifiers(s).
-p <*port*>	Select only sessions with the specified NAS-Port(s).
-r <realm>	Select only sessions connecting from the specified realm(s).
-s <*state*>	Select only sessions with the specified session state(s).
-t <*Token-pool*>	Select only sessions with tokens from the specified pool.
-v	Display version of sesstab.

Troubleshooting

Debugging

The RAD-Series Server can be set to print out debugging information that may be useful for troubleshooting. The debug level and directory is set when the server is started. See "Setting Server Start Options" on page 23 and "radsignal" on page 95 to change the debug level.

All debug output is sent to the `radius.debug` file in the server log file directory.

Error Messages

Below are error messages that may be helpful for troubleshooting.

Server Log File Messages

These messages appear in the server's log file.

Message	Probable Cause	Possible Solution
Discarding '<packet type>' from unknown client '<ip address>'	Using DNS names, rather than IP addresses, to specify clients. Upon startup, entries in your clients file are not yet in the server's buffer, so this message may appear at the beginning of the server log file. It only indicates a problem if the server continues to fail to resolve DNS names.	Enter correct DNS name in clients file entry. Add both backward and forward entries for client in DNS database.
Missing required client 'type=' parameter. Ignoring client('name') on line y	Entry on line *y* of the clients file is missing the required "`type=`" parameter.	Add the omitted parameter to the clients file entry.
Couldn't get our own IPv4 address. ourhostname('XXX')	No entry for the RAD-Series Server in your DNS database or `/etc/hosts.`	Add both backward and forward entries for RAD-Series Server in DNS database or add the host name to /etc/hosts.
doconfig: init_fsm() failed init_fsm: FSM defined with -*x* states from radius.fsm	Finite state machine failed to initialize.	Verify that radius.fsm exists in the server configuration file directory. If you modified the .fsm, be sure all entries follow the correct syntax.

Message	Probable Cause	Possible Solution
file_pass: password failure for 'xxx'	User provided an incorrect password	Verify password with user. Add correct password to .users file or user profile database.
readSessionTable: Session checkpoint file '/var/opt/aaa/data/session.las' is too old, ignored.	The server has been down for an extended period or his clock is wrong	Check to make sure his clock is correct.
Vendor-specific attribute contains unknown vendor code: *y*	Server received an attribute for a vendor that is not defined in the vendors file.	Add vendor to the vendors file.
Received RADIUS attribute with unknown attribute code: Attribute#x is not in dictionary. Received attribute with unknown attribute code: Vendor(#n='<name>') Attribute#x is not in dictionary.	Server received an attribute that is not defined in the dictionary.	Add the attribute to the dictionary.custom file.

radiusd Error Messages

These messages are returned by the RAD-Series Server if radiusd fails to start.

Message	Probable Cause	Solution
Invalid auth port number	Authentication port number is a non-numerical value or out of range	Specify a numerical value in Start Options.
Invalid acct port number	Accounting port number is a non-numerical value or out of range	Specify a numerical value in Start Options.
Invalid option *option*	The named option is invalid.	Review the options in your configuration file entries. See Chapter 5 for a list of valid options.
Couldn't open *path*/logfile.yyyymmdd for logging, error 2 No such file or directory	The directory specified for the logfile does not exist.	Create the directory or specify the correct server log file directory in Server Connections.
Couldn't open *path*/logfile.yyyymmdd for logging, error 13 Permission denied	radiusd does not have write permission for the directory.	Change permissions of directory or specify an alternate server log file directory in Server Connections.

Message	Probable Cause	Solution
bind: Address already in use	Another process is already using the port.	Kill the process or change the port number used by radiusd. The port is specified in Start Options.
Cannot resolve client name 'nas' [gethostbyname() returned 'Unknown host']	NAS’s DNS entry is configured incorrectly or not configured at all.	Check DNS configurations for both forward and reverse entries.

Server Reply Messages

These messages may be sent to users when access has been rejected.

Message	Probable Cause	Possible Solution
Access not allowed	The user’s request matched some configured Deny item	
Authentication failure	The user’s authentication has failed.	
xyz access is prohibited	Users are prohibited from using protocol *xyz* (PPP or SLIP) by their service provider.	
Can’t process request now, try again later	The server has too many pending requests at the moment.	
Error creating file	The server has a file system problem.	Check file permissions for the file and directory.
Improper 'userid@realm' specification	User access ID should be in the format of userid@realm. Most probably, a user has omitted @realm from their logon name and the server has not been configured to handle a NULL realm.	
Invalid authentication realm	The realm identified in the user access ID has not been configured for the server.	Verify that the authfile has an entry for the realm and that radiusd is loading configuration files from the correct location.

Using External Data Stores

The RAD-Series Server allows you to store user profiles in a local realm file or to retrieve them from external data stores. This section shows how to configure the server to access user profiles from:

- LDAP directory servers

Using an LDAP Server

If you only need to retrieve user IDs and passwords from the LDAP server, just follow the procedure "ProLDAP Authentication Type" on page 211.

However, if you wish to implement simple authorization policies with check or reply items, we recommend that you review this entire topic. It will familiarize you with:

- Interlink-specific attributes and object classes
- Extending the standard LDAP schema

About ProLDAP™

ProLDAP™ is Interlink Networks' proprietary LDAP schema, based on the OpenLDAP 2.x release. It interfaces with any LDAP server using protocol version 3. Previous OpenLDAP releases are protocol version 2 and are not supported by the currently distributed Interlink Networks schema files.

To determine the version of the LDAP standard used by a commercially released LDAP server, consult the product documentation or contact the vendor. If your LDAP vendor has extended or otherwise modified the version 3 LDAP standard, you may need to modify the ProLDAP™ schema.

The standard schema file is `iaaa-radius.ldif`. We also include the `55iaaa-radius.ldif` file to extend the schema for Sun Java System Directory Server (formerly called iPlanet).

Note: There may be several schema files installed with an LDAP server. Modifying schema files can result in problems. Never delete unused schema elements from the standard schema. They require no additional operational or administrative overhead and deleting them could cause problems.

Securing LDAP Communications

To use Secure Socket Layer (SSL) when communicating with LDAP directories:

Note: SSL secures the connection to a specific host and logical port, so repeat this procedure for each directory separately.

1. See "ProLDAP Authentication Type" on page 211 for how to modify the LDAP directory configuration.
2. Modify the URL to be `LDAPS`.
3. See "LDAP Connection Properties" on page 185 for how to setup the RAD-Series Server TLS-CACertDir or TLS-CACertFile.
4. Enter the TLS-CertFile if required by the LDAP server.
5. Enter the TLS-Key-File if required by the LDAP server.
6. If the RAD-Series Server is already running:
   - Change directory to `/<server utilities path>` (`/opt/aaa/utl` by default).
   - Stop and restart the AAA Server by running `radiusd.sh restart`

The RAD-Series Server uses OpenSSL, which requires a random number generator that has been seeded with at least 128 bits of randomness. If there is no default seeding file or if the file is too short, the "PRNG not seeded" error message may occur. To avoid this, use an operating system with the randomness devices:

- `/dev/urandom`
- `/dev/random`

On systems without `/dev/urandom` or `/dev/random`, it's a good idea to use the Entropy Gathering Demon (EGD). OpenSSL will automatically look for an EGD socket at `/var/run/egd-pool, /dev/egd-pool, /etc/egd-pool` and `/etc/entropy`.

Failing that, set the environmental variable RANDFILE to point to the default random seed file:

```
/etc/opt/aaa/security/random.rnd
```

Password Encryption

The RAD-Series Server can receive passwords from the LDAP server in clear text format or encrypted by SSHA, MD5, SHA1, x-NT hash, x-lm hash and UNIX-crypt. If passwords are encrypted by other methods, configure the RAD-Series Server to bind to the LDAP server for authentication.

Extending the ProLDAP™ Schema

Do the following if you are storing check and reply items with user profiles in the LDAP directory.

If you are only storing user profiles based on the core LDAP schema (to retrieve the uid and password), this procedure is not necessary.

1 Copy the `iaaa-radius.schema` file to the LDAP server.

2 Modify `slapd.conf` by adding the following line:

```
include Path-to-radiusschemafile
```

Where *`Path-to-radiusschemafile`* is the full path to location of the `iaaa-radius.schema` file.

3 Add the Interlink-specific attributes to your LDAP user profiles. (See aaasample.ldif file for examples.)

If your LDAP server vendor has modified or extended the standard version 3 schema, the `iaaa-radius.schema` file may not work. In this case you will have to modify `iaaa-radius.schema`. The RAD-Series Server includes the `55iaaa-radius.ldif` file as an example of a file that has been modified for the Sun Java System Directory Server (formerly called iPlanet) LDAP server.

Interlink Object Classes

The extended schema files contain the Interlink object class:

Object Class	Description
aaaPerson	Represents the set of attributes a user entry must or may contain.

Interlink-specific LDAP Attribute Types

The following Interlink LDAP attributes are used to store check/reply items

Attribute Type	Description
aaaCheck	An A-V pair that must be present in the user entry for the entry to evaluate to True
aaaDeny	An A-V pair that must **not** be present in the user entry for the entry to evaluate to True
aaaReply	An A-V pair sent back to the access device to authorize the session (e.g. to set a session time limit)

LDIF User Entry Syntax

User entries must contain any attribute-value pairs required by the aaaPerson object class or whichever alternate object class you specify for the ObjectClass parameter.

In an LDIF file, a user entry is represented as follows:

```
dn: attr = value, attr = value,...
ObjectClass: aaaPerson
uid: value
userPassword: value
User-ID: value
User-Password: value
aaaCheck: attr = value
aaaDeny: attr = value
aaaReply: attr = value
```

Parameter	Description
dn	Distinguished Name. Identifies a directory entry in the LDAP schema by using a series of comma-separated attributes and values, where the left-most value specifies the actual directory object and the right-most value specifies the directory root point. For example, ou=people, o=interlink.com points to the organizational unit, people, located on the directory branch below the organization, interlink.com.
uid	User identifier. Normally, uid or cn is part of dn. This is the default filter for a realm. If you specify another attribute as the filter, this parameter is optional.
userPassword	LDAP attribute that identifies the password for the user.
ObjectClass	Indicates what A-V pairs must or may be used in the user entry. This attribute is not required for user authentication, but it is required if you wish to add User-ID, User-Password, Check/Deny, or Reply Items to the user profile. In addition to or instead of aaaPerson, other object classes (e.g., the standard LDAP schema Person class) may be specified.
User-ID	RADIUS attribute that may be used to identify a user (instead of uid).
User-Password	RADIUS attribute, defined in the aaaPerson object class.
aaaCheck, aaaDeny, aaaReply	Specify any A-V pair(s) defined in the RAD-Series Server's dictionary files that you wish to use as a check, deny, or reply item for the user.

Check and Reply Items

The ProLDAP™ implementation lets you store check and reply items with user profiles.

If your RAD-Series Server implementation includes user check or reply items, configure these items to be returnable in a search of the LDAP directory. Binding as the user will not return check items.

To Override Simultaneous Session Limit

This A-V pair overrides the global simultaneous session limit set in Server Properties.

```
aaaCheck: Simultaneous-Use = Max-number-sessions
```

To Control Access Points

These A-V pairs specify the NAS port or machine through which the user is permitted to access the network.

```
aaaCheck: NAS-Port = Port-number
aaaCheck: NAS-ID = ID-string
aaaCheck: Calling-Station-ID = User-MAC-address
```

To Deny Access

These A-V pairs deny the user access from the port or machine specified.

```
aaaDeny: NAS-Port = Port-number
aaaDeny: NAS-ID = value
aaaDeny: Called-Station-ID = AP-MAC-Address
```

To Set Reauthentication Session Timeout

```
aaaReply: Session-Timeout = Number-seconds
aaaReply: Termination-Action = 1
```

To Set Idle Timeout

```
aaaReply: Idle-Timeout = Number-seconds
```

To Assign Static IP Address

This A-V pair cannot be used with an 802.1X framework.

```
aaaReply: Framed-IP-Address = value
```

To Apply Filters

```
aaaReply: Filter-ID = value
```

Notes on ProLDAP™ Configuration

This section contains a few notes about configuring the RAD-Series Server to use ProLDAP.

Case-sensitive Filter IDs

A feature of LDAP servers is that they match Distinguished Names in a case-insensitive manner. Using case-sensitive filter IDs could cause Distinguished Names to be mismatched or not found.

Clear-text passwords stored in LDAP

The regular expression `^{.+}.*$` indicates that the password is hashed using the scheme named between the { and } characters. There is no specification for handling clear-text (unhashed) passwords that contain a leading prefix of `{.+}`. This leads to confusions, such as:

`{md5}` looks like hasher '`md5`' produced a zero-length hash

`{nothing}interesting` looks like hasher '`nothing`' produced a hash of '`interesting`'

Any clear-text password that appears to be hashed will be rejected as invalid by the Interlink RAD-Series Server. Most likely, the apparent hasher will be rejected as invalid. Even if the apparent hasher is valid, the apparent hash will be invalid. Therefore, a user could supply the correct password, only to have the server reject it as invalid. It should be noted that the LDAP standard already recommends against storing clear-text passwords in an LDAP directory.

Using Advanced Policy

The RAD-Series Server Advanced Policy module lets you define advanced policies to control the processing of requests. Advanced policies are logical expressions that result in a true or false answer when evaluated against a request. Depending on the outcome, A-V pairs may be added to the request as reply items and event codes may be returned to the FSM to control progress to the next state.

Unlike the user check and deny items, advanced policies can include a wide range of criteria and comparisons using:

- Boolean operators and relative operators
- Complex expressions
- Nested statements

Advanced policies are specifically used to implement:

- Dialed Number Identification Service (DNIS) routing
- Dynamic Access Control (DAC) time-based provisioning
- Control access according to NAS addresses or ports
- Logical user groups (other than realm)

Decision Files

Advanced policies are defined in decision files. The older format contains policy group entries which are still supported but it is not documented here. If you are using the group policy format, then refer to your old documentation. The new format is more of a scripting language and should be easier to use. The Decision file must be all the same format, the formats can not be intermixed.

Decision files make policy decisions by matching requests to a sequence of conditions. The decision file is evaluated against the request removing, modifying and/or adding A-V pairs as directed until an exit command is encountered. Any remaining lines are not evaluated. The exit command specifies the state machine event to give the state machine. The event is used to control the flow through the state machine. For instance, the NAK event will cause the state machine to send a NAK to the request. See "Finite State Machine (FSM)" on page 256 for more information on FSMs.

Implementing Advanced Policies

The default FSM file distributed with the RAD-Series Server provides several places to implement advanced policies:

1 Request pre-processing policy

2 User/Realm policy

3 Reply post-processing policy

4 Proxy send request policy

5 Proxy receive response policy

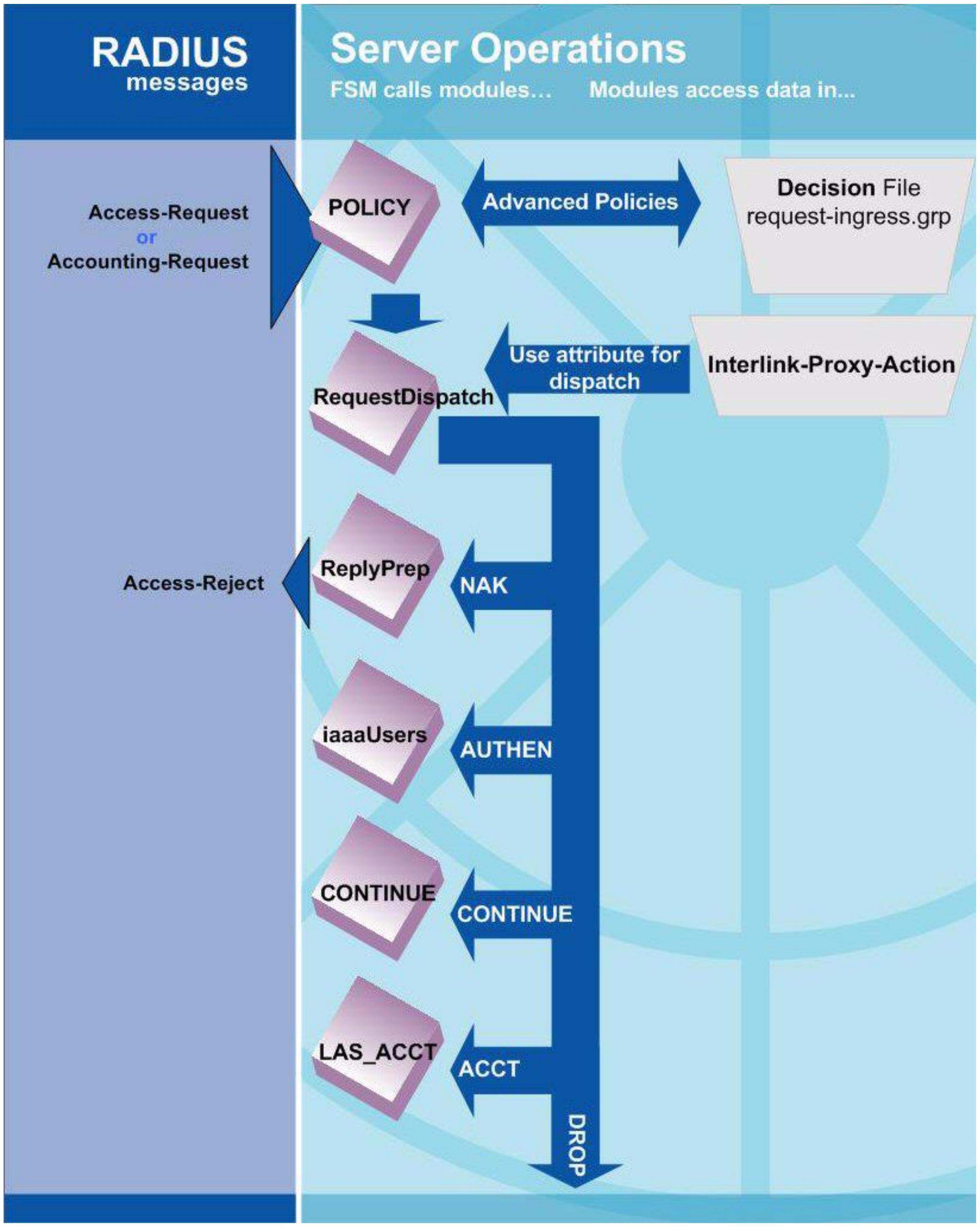
The `radius.fsm` file can also be modified to implement advanced policies in decision files at additional points in the FSM file. See “Calling Decision Files” on page 159 for further details on modifying the FSM file to implement advanced policies

Request pre-processing policy

All requests are subjected to the request pre-processing policy that is defined in the `request-ingress.grp` decision file. The request pre-processing policy is applied as the very first step in the FSM, before the request is dispatched for processing.

The request pre-processing policy may alter the request arbitrarily:

- A-V pairs may be added, changed, or removed
- The request classification may be altered
- The request may be rejected immediately
- The request may be dropped entirely, there will be no reply sent



Flow of Request Pre-Processing Policy

User/Realm policy

All requests are subjected to user/realm policy after authentication. User/realm policy is applied only after a successful authentication.

A user policy is specified in a `Policy-Pointer` attribute on the request as either a check item or a reply item, see “Policy-Pointer” on page 55 for information on configuring it. If the `Policy- Pointer` attribute is found in the check items then the Server will not look for one in the reply items. The value of the `Policy-Pointer` attribute is treated just like the `Xstring` value from the FSM file; it specifies the URL for the decision file to be evaluated.

A realm policy is specified in a `Policy-Pointer` parameter in a `ProLDAP` realm entry in an `authfile`. The value of the `Policy-Pointer` parameter is treated just like the `Xstring` value from the FSM file; it specifies the URL for the decision file to be evaluated.

If a request contains a `Policy-Pointer` attribute, as either a check item or a reply item, the named advanced policy will be applied. If a request contains no `Policy-Pointer` attribute and the realm entry has a `Policy-Pointer` parameter, the named advanced policy will be evaluated. If neither the request, nor the realm entry, contain a `Policy-Pointer`, then no user or realm policy is applied; in this case the `POLICY` action returns an `ACK` event to the FSM.

See the “*Authorization flow in default Finite State Machine*” on page 9 for the flow of user/realm policy.

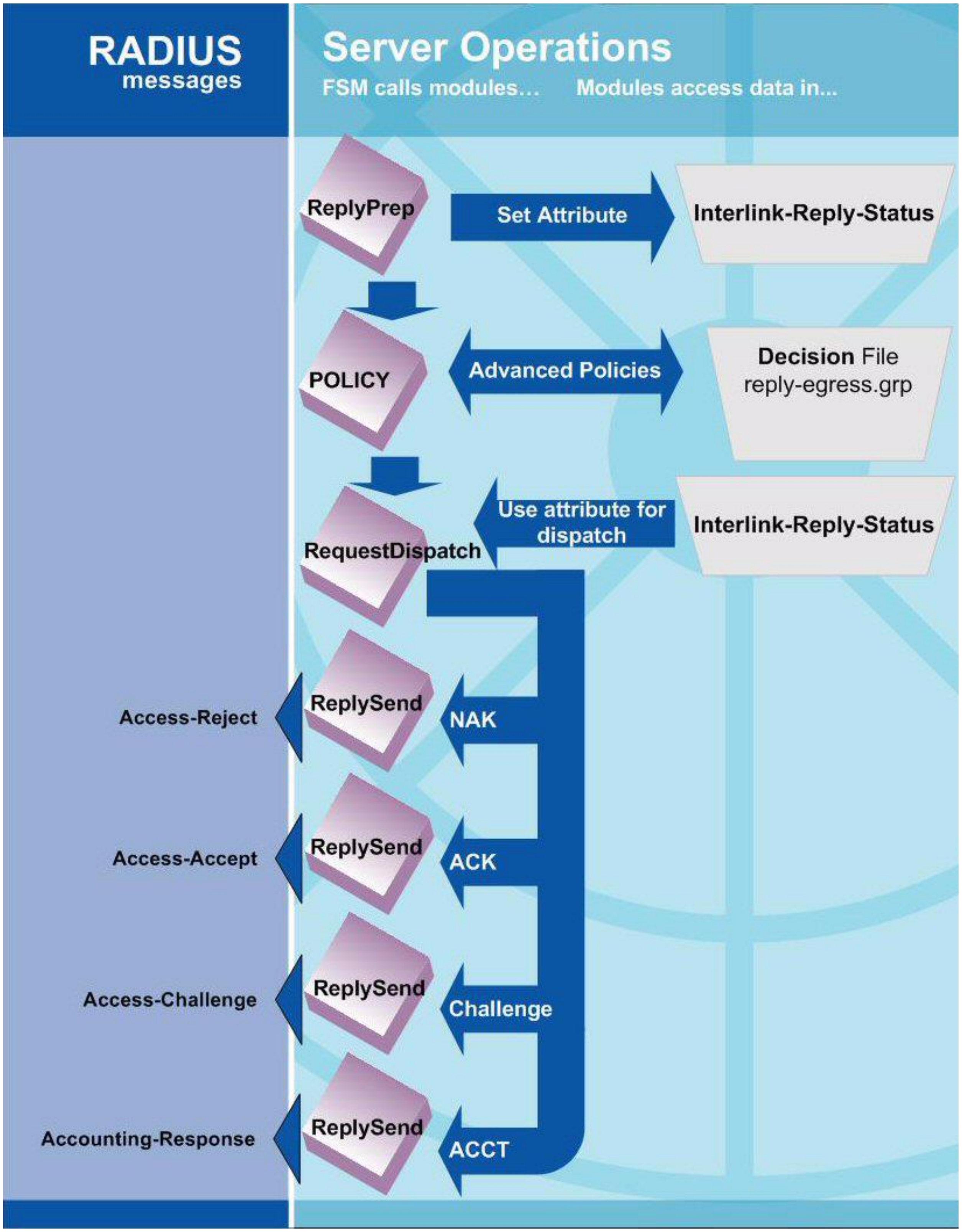
Reply post-processing policy

All replies are subjected to the reply post-processing policy that is defined in the `reply-egress.grp` decision file. The reply post-processing policy is applied as the very last step in the FSM, just before the RADIUS reply message is created and sent.

The reply post-processing policy may alter the request arbitrarily:

- A-V pairs may be added, modified, or removed
- The reply type may be changed
- The request may be dropped entirely, there will be no reply sent

If the client is defined as `type=NAS` or `type=PROXY+PRUNE` (possibly including vendors), the pruning rules specified in the dictionary files will have been applied according to the reply type that was in effect before the post-processing policy is evaluated.



Flow of Reply Post-Processing Policy

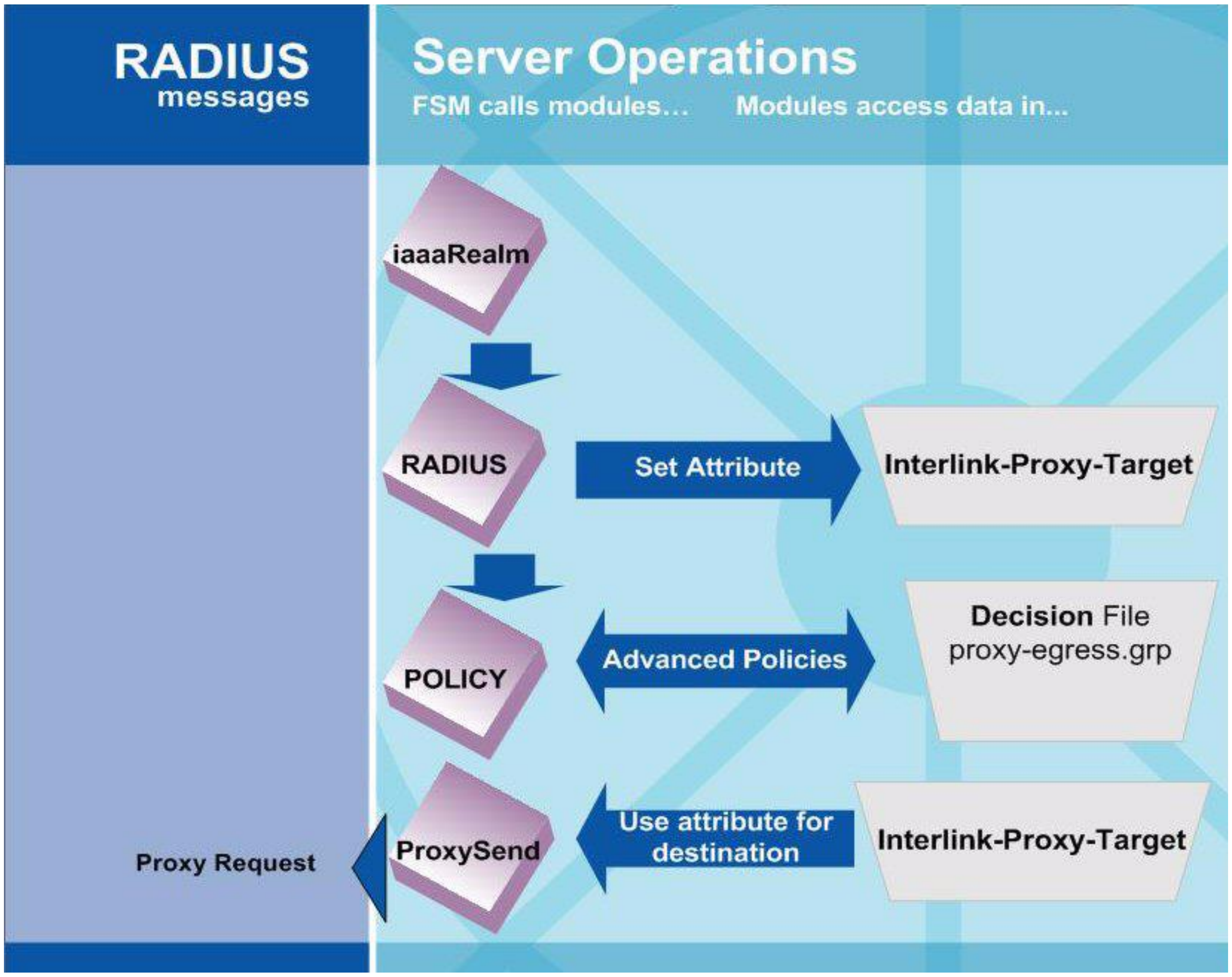
Proxy send request policy

All requests that need to be proxied are subjected to the proxy send request policy defined in the proxy-egress.grp decision file. The proxy send request policy is applied as the very last step in the FSM before the RADIUS proxy request message is created and sent.

The proxy send request policy may alter the request arbitrarily except as noted below:

- A-V pairs may be added, modified, or removed
- The request may be rejected immediately
- The request may be dropped entirely, there will be no reply sent
- The proxy target host may be changed

Note: Do NOT modify or remove, any Proxy-State or Proxy-Action A-V pairs. Doing so will interfere with proxy functionality.



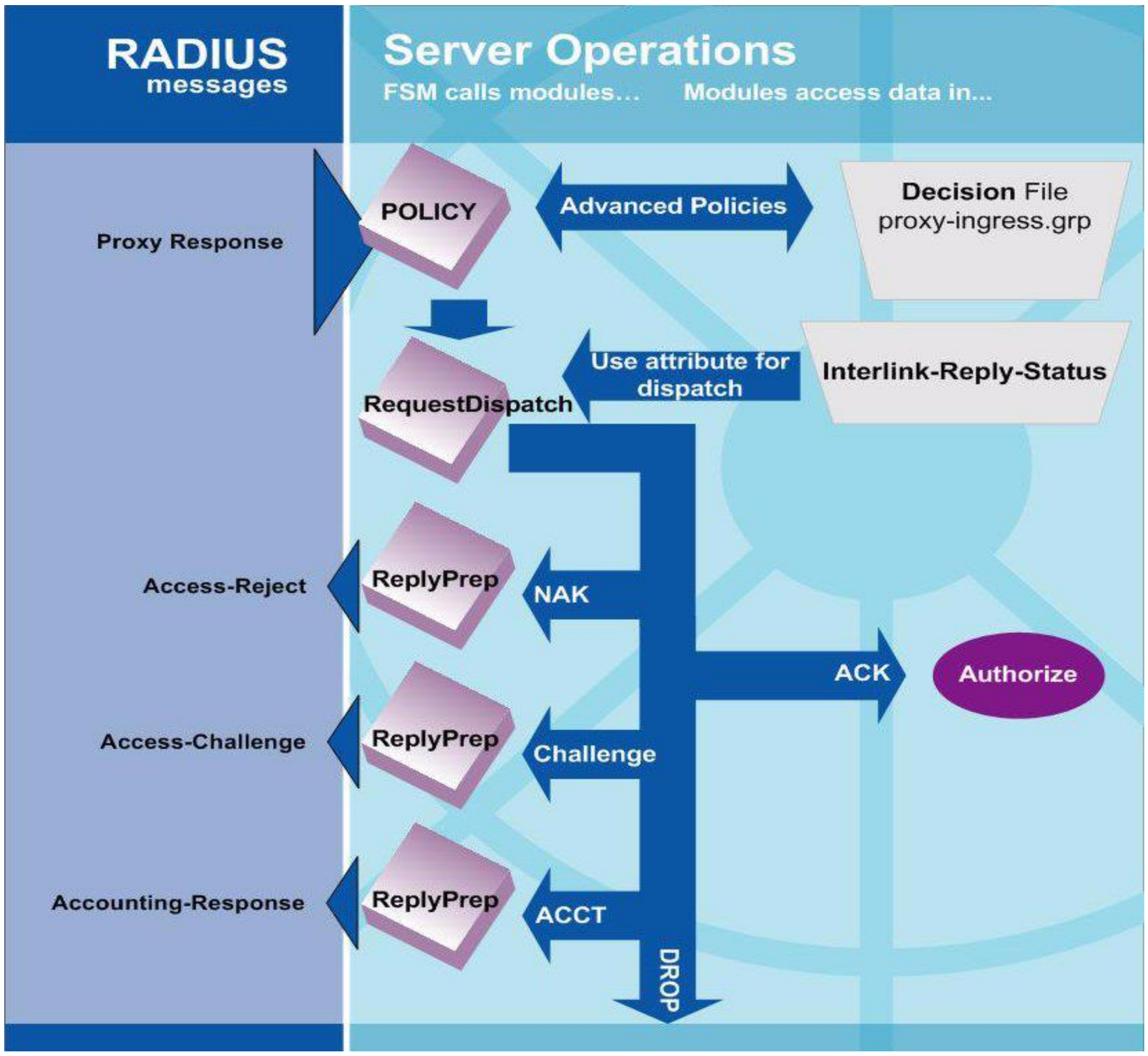
Flow of Proxy Send Request Policy

Proxy receive response policy

All proxy replies are subjected to the proxy receive response policy defined in the proxy-ingress.grp decision file. The proxy receive response policy is applied as the very first step in the FSM after the reply is received.

The proxy receive response policy may alter the request arbitrarily:

- A-V pairs may be added, modified, or removed
- The reply type may be altered
- The request may be rejected immediately
- The request may be dropped entirely, there will be no reply sent



Flow of Proxy Receive Response Policy

Policy Syntax

There are several action command keywords:

- **if (*<bool-expr>*) { *<action-list1>* }** else { *<action-list2>* }
- **delete *<attr-spec>***
- **insert *<attr-spec>* = *<value-expr>***
- **modify *<attr-spec>* = *<value-spec>***
- **exit "*<event-name>*"**
- **log "*<log-level>*" "*<log-message>*", *<attr-spec>*, ...** *<attr-spec>*

Keywords and function names are case-sensitive.

Command	Parameter Description
if	***<bool-expr>*** is a boolean expression. See "Boolean Expressions" on page 137 for a description. ***<action-list1>*** and ***<action-list2>*** are sequences of action commands that may include additional if commands, nested to an arbitrary depth. When the else clause is omitted, ***<action-list2>*** may be considered an empty sequence of action commands.
delete	***<attr-spec>*** is an attribute specification. The use of instance specification, including "last" and "*", is allowed. The use of attribute functions is not permitted
insert	***<attr-spec>*** is an attribute specification. The use of instance specification, including "begin" and "last" is allowed. The use of attribute functions is not allowed. The default instance for ***<attr-spec>*** is "last". ***<value-expr>*** is a value expression. ***<value-expr>*** may use an attribute specification. The use of instance specification, including "last", is allowed. The use of attribute functions is allowed. If ***<value-expr>*** refers to an attribute, the default instance is "last". If ***<value-expr>*** refers to an instance that is not present, a no-such-instance run- time error occurs. See "Error Handling" on page 156 for details. The types of ***<attr-spec>*** and ***<value-spec>*** must be compatible. See "Type Compatibility" on page 140 for details.
modify	***<attr-spec>*** is an attribute specification. The use of instance specification, including "last", is allowed. The use of attribute functions is not allowed. The default instance for ***<attr-spec>*** is "last". If ***<attr-spec>*** refers to an instance that is not present, a no-such-instance run-time error is generated. See "Error Handling" on page 156 for more details. ***<value-spec>*** is a value specification. ***<value-spec>*** may use attribute-specifications. The use of instance specification, including "last", is allowed. The use of attribute functions is allowed. If ***<value-spec>*** refers to an attribute, the default instance for <value-spec> is "last". If ***<value-spec>*** refers to an instance that is not present, a no-such-instance run- time error is generated. See "Error Handling" on page 156 for more details. The value types for ***<attr-spec>*** and ***<value-spec>*** must be compatible. See "Type Compatibility" on page 140 for details.

Command	Parameter Description
exit	**<*event-name*>** must be a quoted string. **<*event-name*>** must specify an event that is defined. There are a number of predefined events. Additional events may be defined in the FSM file using "`%event <name>`" syntax. See "Events" on page 258 for more detailed information on finite state machine events. Event names are case-insensitive (MyEvent is the same as MYEVENT)
log	**<*log-level*>** must be a quoted string. **<*log-level*>** must be a log-level type: `ALERT, CRITICAL, ERROR, WARNING, NOTICE, INFO` **<*log-level*>** is case-insensitive ("ERROR" is the same as "Error"). **<*log-message*>** must be a quoted string. Multiple instances for **<*attr-spec*>** are allowed and cause all named instances to be reported in the logfile. The use of instance specification, including "last" and '*', is allowed. The default instance specification for the attributes in the **log** command is "last". If **<*attr-spec*>** refers to an instance that is not present, this will be indicated in the logfile output. A no-such-instance run-time error will NOT be generated in this case.

There are several Attribute Functions:

- **`count( <attr-spec> )`**
- **`length( <attr-spec> )`**
- **`substr( <attr-spec> ... )`**
- **`tolower( <attr-spec> )`**
- **`toupper( <attr-spec> )`**

Keywords and function names are case-sensitive.

Function	Parameter Description
count	***`<attr-spec>`*** is an attribute specification. The use of instance specification, including “last” and “*”, is allowed. The use of an attribute function is not allowed. The default instance for ***`<attr-spec>`*** is “*”.
length	***`<attr-spec>`*** is an attribute specification. The use of instance specification, including “last”, is allowed. The use of an attribute function is not allowed. The default instance for ***`<attr-spec>`*** is “last”.
substr	***`<attr-spec>`*** is an attribute specification. The use of instance specification, including “last”, is allowed. The use of an attribute function is not allowed. The default instance for ***`<attr-spec>`*** is “last”. Keywords of “offset”, “before”, “before last”, “after” and “after last” define the rest of the command format. See “substr **function**” on page 152 for a detailed description.
tolower	***`<attr-spec>`*** is an attribute specification. The use of instance specification, including “last”, is allowed. The use of an attribute function is not allowed. The default instance for ***`<attr-spec>`*** is “last”.
toupper	***`<attr-spec>`*** is an attribute specification. The use of instance specification, including “last”, is allowed. The use of an attribute function is not allowed. The default instance for ***`<attr-spec>`*** is “last”.

Decision files are evaluated from beginning to end. The Decision file is evaluated against the request removing, modifying and/or adding A-V pairs as directed until an exit command is encountered. Any remaining lines are not evaluated. The exit command specifies the state machine event to give the state machine. The event is used to control the flow through the state machine. For instance, the NAK event will cause the state machine to send a NAK to the request. If the end of the file is reached without executing an exit command then the ACK event will be returned to the state machine. See “Finite State Machine (FSM)” on page 256 for more information on FSMs.

Use a pound sign (#) to comment out any lines you do not want applied.

Decision files must be stored in the same directory that contains the RAD-Series Server's other configuration files. There are no required naming conventions for a decision file, but the file name must match its reference in `radius.fsm`, `authfile`, realm files, or the default users file.

Attribute Specifications

Attribute specifications have several pieces:

- Attribute Name
- Tag Value
- Value
- Instance
- Functions

Attribute Names

Attribute names are specified by simply writing the name of the attribute.

For example:

```
Reply-Message
Cisco-AvPair
Xvalue
```

Attribute names are defined in the RAD-Series Server's dictionary files.
Attribute names are case-insensitive (`Reply-Message` is the same as `REPLY-MESSAGE`).

Named Attribute List

For all commands, the attributes are assumed to reside on the request's default attribute list, which is unnamed and unspecified. However, a request may hold multiple attribute lists, and a plugin may create a list, name this list, and populate this list with attributes which can be operated on by advanced policy commands.

For example, suppose a plugin has created and populated a list named "`Proxy-Response-Attributes`". Referencing the Attribute Name "`Reply-Message`" would attempt to find and operate on a `Reply-Message` attribute found in the request's default attribute list (e.g. "`delete Reply-Message`"). Referencing the Attribute Name "`Proxy-Response-Attributes/Reply-Message`" would attempt to find and operate on a `Reply-Message` attribute found the the request's "`Proxy-Response-Attributes`" attribute list (e.g. "`delete Proxy-Response-Attributes/Reply-Message`"). If a list named, say "`Wireless-Attributes`", is referenced in a policy but that named list does not exist, then it is created.

The syntax of an <attr-spec> specifying a specific named list is:

> `ListName/AttributeName` if no instance specifier is given, or
> `ListName/AttributeName[instance]` otherwise.

Note: List names are case-sensitive.

Examples:

```
delete Proxy-Response-Attributes/Reply-Message[*]
delete Proxy-Response-Attributes/Reply-Message
delete Proxy-Response-Attributes/NAS-IP-Address[last]
delete Proxy-Response-Attributes/NAS-IP-Address[0]
```

Tag Value

If the attribute is a tagged attribute then the reference to the attribute may optionally include a tag value using :<tag>: format. See “Value Specifications” on page 124 for more information.

Value

See “Value Specifications” on page 124 for a description of the value syntax.

Attribute Instance Specifications

There can be more than one instance of a given attribute on the request. It is sometimes necessary to specify which instance is of interest. It is sometimes necessary to specify the absolute location of an attribute instance (when inserting, for example).

Attribute instance specifications are provided using [] syntax following the attribute name. The instance of interest is indicated inside the []'s. The instance can be specified numerically using a positive integer constant, using a keyword or using the special symbol '*'.

For example:

```
Reply-Message
Reply-Message[2]
Reply-Message[last]
Reply-Message[*]
```

White space is allowed around and between the []'s.

For example:

```
Reply-Message [ 3 ]
Reply-Message[ last]
```

For attributes of type TLV where a subattribute is referenced, the instance specifier refers to the instance of the top-level parent TLV and is therefore placed after the top-level parent attribute name, for example:

```
WiMAX-Time-Of-Day-Time.Hour

WiMAX-Time-Of-Day-Time[2].Hour

WiMAX-Time-Of-Day-Time[last].Hour

WiMAX-Time-Of-Day-Time[*].Hour
```

The instance(s) of subattribute(s) cannot be specified and defaults to [last].

No instance specification

When the specific instance is of no consequence, it may be left unspecified. This is equivalent to specifying [last] (see “**`last`** keyword” on page 122) in all cases except the count() function for which the default instance is [*]. See the individual action-command, attribute function and comparison operator descriptions for details.

Numeric instance specification

When a specific instance is desired, it may be specified numerically. Instances are numbered from 0 (the first instance). Negative instance numbers are not allowed.

Suppose that:

```
Reply-Message = "abc"
Reply-Message = "def"
Reply-Message = "ghi"
```

Then:

```
Reply-Message[0] = "abc"
Reply-Message[1] = "def"
Reply-Message[2] = "ghi"
```

Keyword instance specification

When a specific instance is desired, it may be specified using one of the following keywords which are case sensitive or the special symbol '*****':

- **`begin`** keyword

  When the beginning of the list is desired (when inserting), use the “begin” keyword. This keyword is only supported by the **`insert`** command, on the left side of the '**=**'.

  For example:

  ```
  insert Reply-Message[begin] = "This is first"
  ```

  See “insert” on page 117 for details and examples.

  Using this keyword in other places results in an invalid-instance-specification load-time error. See “Error Handling” on page 156 for more details.

- **`last`** keyword

  When the last instance (the one located closest to the end of the list) of a particular attribute is desired, use the “last” keyword.

  For example:

  ```
  Reply-Message[last]
  ```

  This is the default in all situations. See the individual action command, attribute function and comparison operator descriptions for details.

Suppose that:

```
Reply-Message = "abc"
Reply-Message = "def"
```

*Th*en:

```
Reply-Message[last] = "def"
```

- * keyword

  When all instances are desired, use the '*****' symbol.

  For example:

  ```
  Reply-Message[*]
  ```

  This form is only supported by the delete and log commands and the count() attribute function. Using this form in unsupported contexts results in an invalid-instance-specification load-time error. See “Error Handling” on page 156 for more details.

Attribute Functions

Attribute functions are provided using keyword(<attr-spec>) syntax.

For example:

```
count( Reply-Message )
count( Reply-Message[0] )
count( Reply-Message[*] )
```

White space is allowed around and between the '()' characters.

For example:

```
count( Reply-Message )
count( Reply-Message)
count (Reply-Message)
count (Reply-Message )
```

Are all valid.

There are many possible attribute functions. See description in “Attribute Functions” on page 152 for details.

Value Specifications

Constant values may be specified in various ways depending on the value type. See "Type Compatibility" on page 140 for more details.

1. Integer values

   Integer values may be specified as decimal integers, including a leading '-' sign. Integer values can not have any leading zeros. A tag may also be specified by using :tag: syntax prefixed to the value. The tag value must be in the range of 0 to 31.

   For example these are legal:

   ```
   10
   0
   -37
   :3:10
   :0:37
   ```

   These are NOT legal:

   ```
   0123
   :32:92
   :0:-37
   ```

   Integer values may also be specified as hexadecimal integers.
   Hexadecimal values are always unsigned.

   For example:

   ```
   0x3
   0x6F32e5
   0x00AB34
   :1:0x3
   :9:0xf3AB
   ```

   Integer values can be used with integer, unsigned8, unsigned16, signed32, signed64, unsigned64 and tag-int type attributes.

   Integer values used with an integer attribute can have a value range of 0 to 4294967295.
   Integer values used with an signed32 attribute can have a value range of -2147483648 to 2147483647.
   Integer values used with an signed64 attribute can have a value range of -9223372036854775808 to 9223372036854775807.
   Integer values used with an unsigned64 attribute can have a value range of 0 to 18446744073709551615.
   Integer values used with a unsigned16 attribute can have a value range of 0 to 65535.
   Integer values used with an unsigned8 attribute can have a value range of 0 to 255.
   Integer values used with a tagged integer attribute can have a value range of 0 to 16777215.

   Violating the value range results in an invalid integer value load-time error. See "Error Handling" on page 156 for more details.

   An integer value can be entered as a negative number but the attributes and values will be treated as unsigned numbers when comparisons are performed.

2. Named integer values

   Named integer values are enclosed in double quotes "". A tag may be provided by using :tag: syntax prefixed to the named value.

   For example:

   ```
   Service-Type = "FRAMED"
   Tunnel-Type = :12:"PPTP"
   ```

   Named integer values are defined in the RAD-Series Server's dictionary files.
   Named integer values are case-insensitive (FRAMED is the same as Framed).
   Named integer values can only be used with integer and tag-int type attributes that have defined named values.
   Named integer values can only be used with the attribute for which they are defined.
   Using an undefined named value results in an unknown named value load-time error. See “Error Handling” on page 156 for more details.

3. String values

   String values are enclosed in double quotes "". There are escape sequences to allow for the inclusion of non-printable characters in a string value. Tags may be specified by using :tag: syntax prefixed to the value.

   The defined escape sequences and meanings are:

Sequence	Meaning
`\0`	Null
`\″`	Double Quote
`\\`	Backslash
`\b`	Bell
`\n`	Line Feed
`\r`	Carriage Return
`\t`	Tab
`\h`	Backspace
`\f`	Form Feed
`\xHH`	The octet whose value is the specified 2 digit hex number

   For example:

   ```
   "a string value"
   "and\x20some\x20\"escapes\"\x20\n"
   :21:"a string value"
   ```

   String values can be used with string, tag-str and octets type attributes.

   The length of a string attribute and/or string value is not limited except by memory. However, if a string attribute ultimately is inserted into a RADIUS packet then the RAD-Series Server will enforce the following maximums and if violated will result in logging the error in the logfile and dropping the request:

string and octets	253 characters
tag-str	252 characters
vendor specific string and octets	247 characters
vendor specific tag-str	246 characters

4. TLV values

TLV values are enclosed within curly-braces which are in turn enclosed by double quotes "".

For example, subattributes of the WiMAX-Time-Of-Day-Time TLV attribute would be specified as:

```
"{ Hour = 1, Minute = 20, UTC-Offset = -5 }"
```

5. IP Address values

IP Address values are enclosed in double quotes "".
IPv4 address values must be specified in standard dotted-quad notation.
IPv6 address values must be specified in IPv6 standard notation.

For example:

```
“1.2.3.4”
```

```
“2001:1:2::3”
```

The use of an invalid IP address results in a syntax-error load-time error. See “Error Handling” on page 156 for more details.

6 IPv6prefix values

The IPv6 prefixed values are enclosed in double quotes "".
The IPv6 prefixed values must be specified in IPv6 subnet standard notation.

For example:

“1:2:3:4::/64”

The use of an invalid IPv6 prefixed address results in a syntax-error load-time error. See “Error Handling” on page 156 for more details.

7 InterfaceId values

The InterfaceId values are enclosed in double quotes "".
The InterfaceId values must be specified in “hhhh:hhhh:hhhh:hhhh” standard notation.

8 Date values

Date values are enclosed in double quotes, in any of the same formats as described in the "Date Attribute Value Formats" on page 168

9 Tag-Str values

Attributes of type Tag-Str use string values. See String values (item 3 above) for details.

10 Tag-Int values

Attributes of type Tag-Int use integer and named integer values. See Integer values (item 1 above) and Named integer values (item 2 above) for details.

11 Octets values

Attributes of type octets use string values. See String values (item 3 above) for details.

12 Unsigned8 values

Attributes of type unsigned8 use integer values. See Integer values (item 1 above) for details.

13 Unsigned16 values

Attributes of type unsigned16 use integer values. See Integer values (item 1 above) for details.

14 Signed32 values

Attributes of type signed32 use integer values. See Integer values (item 1 above) for details.

15 Signed64 values

Attributes of type signed64 use integer values. See Integer values (item 1 above) for details.

16 Unsigned64 values

Attributes of type unsigned64 use integer values. See Integer values (item 1 above) for details.

Attribute Value Handling

Attributes in RADIUS Messages

Each attribute in a RADIUS message has several fields: attribute code, attribute length, tag and value. For tagged-string attributes, the tag field is optionally present. For vendor-specific attributes, the value field has several sub-fields: vendor code, attribute code, attribute length (details vary by vendor and/or attribute code). See the RADIUS RFCs for additional details.

Attributes in the RAD-Series Server

In the RAD-Series Server, attributes have several fields: vendor code, attribute code, tag and value. For all attributes, the tag field and the vendor code field are always present.

Tagged Attributes and the Advanced Policy Language

In the RAD-Series Server advanced policy language, the tag field of tagged attributes and the value field of tagged attributes are treated separately.

For conditional expressions in the If command, comparisons consider only the value field of attributes and constants. The tag field is ignored.

For the insert command, the tag field from the value on the right hand side of the '=' is preserved in the inserted attribute.

For the modify command, the tag field from the value on the right hand side of the '=' is ignored. The tag field of the modified (target) attribute is left unchanged.

For attribute functions that operate on string values (length, substr, tolower, toupper), only the value field is considered. The tag field is ignored. For attribute functions that return the same type of attribute as their parameter (substr, tolower, toupper), the tag field of the source attribute is preserved in the function return value.

Operators

The following operators may be used:

Operator	Description
=	Equal to, see "= Operator" on page 130
!=	Not equal to, see "!= Operator" on page 131
>	Greater than, see "> Operator" on page 134
<	Less than, see "< Operator" on page 132
>=	Greater than or equal to, see ">= Operator" on page 135
<=	Less than or equal to, see "<= Operator" on page 133
&&	Logical AND, see "&& Operator" on page 137
||	Logical OR, see "|| Operator" on page 137
!	Logical NOT, see "! Operator" on page 138
()	Parentheses, see "() Operator" on page 138

Comparison operators

General comparison operator

1. Syntax

   <value1> <operator> <value2>

2. Parameters

   <value1> and *<value2>* are value expressions.

   <value1> and *<value2>* may use attribute-specifications. The use of instance specification, including "last", is allowed. The use of attribute functions is allowed.

   Either *<value1>* or *<value2>* (or both) must refer to an attribute.

   This determines the type for the comparison.

   The types of *<value1>* and *<value2>* must be compatible. See "Type Compatibility" on page 140 for details.

   If *<value1>* or *<value2>* refers to an attribute, the default instance is "last".

3. Operation

   Returns a boolean value that indicates the result of the comparison.

   If *<value1>* or *<value2>* refers to an attribute that is a tagged type (tag-int or tag-str), only the value portion is used for comparison purposes. See "Tagged Attributes and the Advanced Policy Language" on page 128.

   For string and octets type values the comparison is performed using the native collation sequence of the system (ASCII), in a case-sensitive fashion. Extended characters (those outside the 7-bit ASCII range) are considered as positive values; in other words, characters are considered to be unsigned.

   For integer type values the comparison is performed in a signed fashion, using native system byte ordering.

   For IP Address type values the comparison is performed in an unsigned fashion, using big-endian (network/MSB-first) byte ordering.

   For date type values the comparison is performed in an unsigned fashion, using native system byte ordering.

4. Examples

   Some equivalent specifications (NAS-Port is an example):

   ```
   NAS-Port
   NAS-Port[last]
   ```

   Instance specifications that are allowed:

   ```
   NAS-Port
   ```

```
NAS-Port[0]
NAS-Port[302]
NAS-Port[last]
```

Instance specifications that are **NOT** allowed:

```
NAS-Port[begin]
NAS-Port[*]
```

Using any of these specifications results in an invalid-instance-specification load-time error. See "Error Handling" on page 156 for more details.

See the individual comparison operator descriptions below for more complete examples.

= Operator

1. Syntax

   <value1> = <value2>

2. Parameters

   See Parameters in "General comparison operator" on page 129 for details.

3. Operation

   Compares two values for equality:

   false: *<value1>* not equal to *<value2>*
   true: *<value1>* equal to *<value2>*

   See Operation in "General comparison operator" on page 129 for more details.

   If *<value1>* or *<value2>* refers to an attribute for which there are no instances on the request, the comparison yields false.

4. Examples

   Suppose that:

   ```
   NAS-Port = 2
   ```

   Then:

   ```
   NAS-Port = 2 is true
   NAS-Port = 3 is false
   2 = NAS-Port is true (same as 2 = NAS-Port[last])
   3 = NAS-Port is false (same as 3 = NAS-Port[last])
   ```

   Suppose that:

   ```
   NAS-Port = 2
   NAS-Port = 3
   ```

   Then:

   ```
   NAS-Port = 2 is false
   NAS-Port = 3 is true
   NAS-Port[0] = 2 is true
   ```

```
NAS-Port[1] = 2 is false
NAS-Port[0] = 3 is false
NAS-Port[1] = 3 is true
NAS-Port = NAS-Port is true
NAS-Port[last] = NAS-Port is true
2 = NAS-Port is false (same as 2 = NAS-Port[last])
3 = NAS-Port is true (same as 3 = NAS-Port[last])
NAS-Port[2] = 2 is false (NAS-Port[2] is not present)
```

Suppose that:

```
Reply-Message = "abc"
Tunnel-Password = :2:"abc"
Tunnel-Password = :1:"abc"
```

Then:

```
Tunnel-Password[0] = Tunnel-Password[1] is true
Tunnel-Password[0] = Reply-Message is true
Tunnel-Password[0] = "abc" is true
Reply-Message = Tunnel-Password is true
"abc" = Tunnel-Password[0] is true
:21:"abc" = Tunnel-Password is true
:21:"abc" = Reply-Message is true
```

!= Operator

1. Syntax

   `<value1> != <value2>`

2. Parameters

   See Parameters in “General comparison operator” on page 129 for details.

3. Operation

   Compares two values for inequality:

   false: `<value1>` equal to `<value2>`
   true: `<value1>` not equal to `<value2>`

   See Operation in “General comparison operator” on page 129 for more details.

   If `<value1>` or `<value2>` refers to an attribute for which there are no instances on the request, the comparison yields true.

4. Examples

   Suppose that:

   ```
   NAS-Port = 2
   ```

   Then:

   ```
   NAS-Port != 2 is false
   NAS-Port != 3 is true
   2 != NAS-Port is false
   ```

```
3 != NAS-Port is true
NAS-Port[2] != 2 is true (NAS-Port[2] is not present)
```

Suppose that:

```
NAS-Port = 2
NAS-Port = 3
```

Then:

```
NAS-Port != 2 is true
NAS-Port[0] != 2 is false
NAS-Port[1] != 2 is true
NAS-Port != 3 is false
NAS-Port != 4 is true
2 != NAS-Port is true (same as 2 != NAS-Port[last])
3 != NAS-Port is false (same as 3 != NAS-Port[last])
4 != NAS-Port is true (same as 4 != NAS-Port[last])
NAS-Port[2] != 7 is true (NAS-Port[2] is not present)
```

Suppose that:

```
Tunnel-Password = :2:"abc"
Tunnel-Password = :1:"abc"
```

Then:

`Tunnel-Password[0] != Tunnel-Password[1]` is false

< Operator

1. Syntax

   `<attr1> < <attr2>`

2. Parameters

   See Parameters in "General comparison operator" on page 129 for details.

3. Operation

   Compares two values for less than:

   false: *`<value1>`* not less than *`<value2>`*
   true: *`<value1>`* less than *`<value2>`*

   See Operation in "General comparison operator" on page 129 for more details.

   If *`<value1>`* or *`<value2>`* refers to an attribute for which there are no instances on the request, the comparison yields false.

4. Examples

   Suppose that:

   ```
   NAS-Port = 2
   ```

   Then:

```
NAS-Port < 2 is false
NAS-Port < 3 is true
2 < NAS-Port is false
37 < NAS-Port[2] is false (NAS-Port[2] is not present)
```

Suppose that:

```
NAS-Port = 2
NAS-Port = 3
```

Then:

```
NAS-Port < 2 is false
NAS-Port < 3 is false
NAS-Port < 4 is true
2 < NAS-Port is true (same as 2 < NAS-Port[last])
```

Suppose that:

```
Tunnel-Type = :2:"PPTP" (this has the value 1)
Tunnel-Type = :1:"VLAN" (this has the value 13)
```

Then:

```
Tunnel-Type[0] < Tunnel-Type[1] is true (same as 1 < 13)
Tunnel-Type < 7 is false (same as 13 < 7)
```

Suppose that:

```
Tunnel-Password = :2:"abc"
Tunnel-Password = :1:"def"
```

Then:

```
Tunnel-Password[0] < Tunnel-Password[1] is true
```

<= Operator

1. Syntax

   `<attr1> <= <attr2>`

2. Parameters

   See Parameters in "General comparison operator" on page 129 for details.

3. Operation

   Compares two values for less than or equal to:

   false: `<value1>` not less than or equal to *`<value2>`*
   true: *`<value1>`* less than or equal to *`<value2>`*

   See Operation in "General comparison operator" on page 129 for more details.

   If *`<value1>`* or *`<value2>`* refers to an attribute for which there are no instances on the request, the comparison yields false.

4. Examples

   Suppose that:

   ```
   NAS-Port = 2
   ```

   Then:

   ```
   NAS-Port <= 2 is true
   NAS-Port <= 3 is true
   2 <= NAS-Port is true
   3 <= NAS-Port is false
   NAS-Port[2] <= 55 is false (NAS-Port[2] is not present)
   ```

   Suppose that:

   ```
   NAS-Port = 2
   NAS-Port = 3
   ```

   Then:

   ```
   NAS-Port <= 2 is false
   NAS-Port <= 3 is true
   2 <= NAS-Port is true (same as 2 <= NAS-Port[last])
   3 <= NAS-Port is true (same as 3 <= NAS-Port[last])
   ```

   Suppose that:

   ```
   Tunnel-Password = :2:"abc"
   Tunnel-Password = :1:"abc"
   Tunnel-Password = :3:"def"
   ```

   Then:

   ```
   Tunnel-Password[0] <= Tunnel-Password[1] is true
   Tunnel-Password[0] <= Tunnel-Password[2] is true
   ```

> Operator

1. Syntax

   `<attr1> > <attr2>`

2. Parameters

   See Parameters in "General comparison operator" on page 129 for details.

3. Operation

   Compares two values for greater than:

   false: `<value1>` not greater than `<value2>`
   true: `<value1>` greater than `<value2>`

   See Operation in "General comparison operator" on page 129 for more details.

   If `<value1>` or `<value2>` refers to an attribute for which there are no instances on the request, the comparison yields false.

4. Examples

   Suppose that:

```
NAS-Port = 2
```

   Then:

```
NAS-Port > 1 is true
NAS-Port > 2 is false
NAS-Port > 3 is false
NAS-Port > 4 is false
1 > NAS-Port is false
2 > NAS-Port is false
3 > NAS-Port is true
4 > NAS-Port is true
NAS-Port[2] > 1 is false (NAS-Port[2] is not present)
```

   Suppose that:

```
NAS-Port = 2
NAS-Port = 3
```

   Then:

```
NAS-Port > 1 is true
NAS-Port > 2 is true
NAS-Port > 3 is false
NAS-Port > 4 is false
1 > NAS-Port is false
2 > NAS-Port is false
3 > NAS-Port is false
4 > NAS-Port is true
```

   Suppose that:

```
Tunnel-Password = :2:"def"
Tunnel-Password = :1:"abc"
```

   Then:

```
Tunnel-Password[0] > Tunnel-Password[1] is true
```

>= Operator

1. Syntax

   `<attr1> >= <attr2>`

2. Parameters

   See Parameters in "General comparison operator" on page 129 for details.

3. Operation

   Compares two values for greater than or equal to:

false: `<value1>` not greater than or equal to `<value2>`
true: `<value1>` greater than or equal to `<value2>`

See Operation in "General comparison operator" on page 129 for more details.

If `<value1>` or `<value2>` refers to an attribute for which there are no instances on the request, the comparison yields false.

4. Examples

Suppose that:

```
NAS-Port = 2
```

Then:

```
NAS-Port >= 1 is true
NAS-Port >= 2 is true
NAS-Port >= 3 is false
NAS-Port >= 4 is false
1 >= NAS-Port is false
2 >= NAS-Port is true
3 >= NAS-Port is true
4 >= NAS-Port is true
NAS-Port[2] >= 2 is false (NAS-Port[2] is not present)
```

Suppose that:

```
NAS-Port = 2
NAS-Port = 3
```

Then:

```
NAS-Port >= 1 is true
NAS-Port >= 2 is true
NAS-Port >= 3 is true
NAS-Port >= 4 is false
1 >= NAS-Port is false
2 >= NAS-Port is false
3 >= NAS-Port is true
4 >= NAS-Port is true
```

Suppose that:

```
Tunnel-Password = :2:"abc"
Tunnel-Password = :1:"abc"
Tunnel-Password = :3:"def"
```

Then:

```
Tunnel-Password[0] >= Tunnel-Password[1] is true
Tunnel-Password[2] >= Tunnel-Password[1] is true
```

Boolean Expressions

&& Operator

1. Syntax

   `<left-expr>` `&&` *`<right-expr>`*

2. Parameters

   `<left-expr>` and *`<right-expr>`* are boolean expressions.

3. Operation

   The && operator evaluates the *`<left-expr>`* always. The *`<right-expr>`* is evaluated only when the *`<left-expr>`* evaluates to a true value.

   This is the traditional short-circuit evaluation most people expect.

   The value is true if both the *`<left-expr>`* and *`<right-expr>`* evaluate to true values. Otherwise the value is false.

4. Examples

   Suppose that:

   ```
   NAS-Port = 2
   Reply-Message = "hello"
   ```

   Then:

   ```
   NAS-Port = 2 && Reply-Message = "hello" is true
   ```

|| Operator

1. Syntax

   `<left-expr>` `||` *`<right-expr>`*

2. Parameters

   `<left-expr>` and *`<right-expr>`* are boolean expressions.

3. Operation

   The || operator evaluates the *`<left-expr>`* always. The *`<right-expr>`* is evaluated only when the *`<left-expr>`* evaluates to a false value.

   This is the traditional short-circuit evaluation most people expect.

   The value is true if either the *`<left-expr>`* or the *`<right-expr>`* evaluates to a true value. Otherwise the value is false.

4. Examples

   Suppose that:

   ```
   NAS-Port = 2
   ```

```
Reply-Message = "hello"
```

Then:

```
NAS-Port = 2 || Reply-Message = "xyz" is true
NAS-Port = 7 || Reply-Message = "hello" is true
```

! Operator

1. Syntax

   `! <expr>`

2. Parameters

   <expr> is a Boolean expression.

3. Operation

   The ! operator evaluates the *<expr>*. The value is the boolean complement of the *<expr>* value.

4. Examples

   Suppose that:

   ```
   NAS-Port = 2
   ```

   Then:

   ```
   ! NAS-Port = 3 is true
   ! NAS-Port > 1 is true
   ! NAS-Port = 2 is false
   ```

() Operator

1. Syntax

   `( <expr> )`

2. Parameters

   <expr> is a boolean expression.

3. Operation

   The *<expr>* is evaluated. The value is the result of the evaluation.

4. Examples

   Suppose that:

```
NAS-Port = 2
```

Then:

```
(NAS-Port = 2) is true
((NAS-Port = 2)) is true
(NAS-Port = 3) is false
((NAS-Port = 3)) is false
```

Operator precedence and association

When multiple operators appear in a boolean expression, the following precedence and association rules are applied.

1. Precedence rules

   The precedence rules in decreasing order are:

   ()
   !
   && ||

2. Association rules

   The association rules are:

   && left-to-right
   || left-to-right
   ! right

3. Examples:

   The boolean expression:

   ```
   Reply-Message = "hello" && NAS-Port > 7 ||
   Reply-Message = "goodbye" || Reply-Message = "nothing"
   ```

   Is fully parenthesized as:

   ```
    ( ( (Reply-Message = "hello") && (NAS-Port > 7) ) ||
        (Reply-Message = "goodbye") ) ||
      (Reply-Message = "nothing") )
   ```

   And is evaluated:

   ```
   if ( Reply-Message = "hello" )
       if ( NAS-Port > 7 )
           return true
   if ( Reply-Message = "goodbye" )
       return true
   if ( Reply-Message = "nothing" )
       return true
   return false
   ```

   The boolean expression:

```
Reply-Message = "goodbye" ||
! Reply-Message = "hello" && NAS-Port > 7
```

Is fully parenthesized as:

```
( (Reply-Message = "goodbye") ||
  ( ! (Reply-Message = "hello") ) &&
  (NAS-Port > 7)
```

And is evaluated:

```
if ( Reply-Message = "goodbye" )
    if ( NAS-Port > 7 )
        return true
    else
        return false
else
    if ( Reply-Message = "hello" )
        return false
    else
        if ( NAS-Port > 7 )
            return true
        else
    return false
```

Type Compatibility

Attribute types are compatible if the value-types are the same.

Integer-value attribute types:
- integer
- tag-int
- unsigned8
- unsigned16
- signed32
- signed64
- unsigned64

String-value attribute types:
- string
- tag-str
- octets

Date-value attribute types:
- date

IPv4-address-value attribute types:
- ipaddr
- ip46addr

IPv6-address-value attribute types:
- ipv6addr
- ip46addr

IPv6-prefix-value attribute types:
 ipv6prefix

InterfaceId-value attribute types:
 interfaceid

It is not permissible to mix attributes from different value-type groups. Doing so results in a type- mismatch load-time error. See “Error Handling” on page 156 for more details.

Since an integer attribute can be copied into any of the other three integer type attributes, it is possible to end up with a value that does not fit into that attribute when placed into a RADIUS packet. When the RAD-Series Server encounters excess bits while putting an attribute into the packet it will log the issue in the logfile and ignore the excess high order bits.

Action Commands

if Command

1. Syntax

```
if ( <bool-expr> ) { <action-list1> } else { <action-list2> }
if ( <bool-expr> ) { <action-list1> }
```

2. Parameters

   `<bool-expr>` is a boolean expression. See “Boolean Expressions” on page 137.

   `<action-list1>` and `<action-list2>` are sequences of action commands that may include additional if commands, nested to an arbitrary depth.

   When the else clause is omitted, `<action-list2>` may be considered an empty sequence of action commands.

3. Operation

   Conditionally evaluates policy actions.

   Evaluates the `<bool-expr>`. See “Tagged Attributes and the Advanced Policy Language” on page 128 for how tags are handled.

   If the result of evaluating `<bool-expr>` is true, evaluates `<action-list1>`.

   If the result of evaluating `<bool-expr>` is false (and an else clause is present), evaluates `<action-list2>`.

4. Examples

- Suppose that:

```
Session-Limit[0] = 10
Session-Limit[1] = 300
```

  The commands:

```
   if ( Session-Limit[1] < 30 )
```

```
    {
        modify Session-Limit[1] = 30
    }
    else
    {
        if ( Session-Limit[1] > 240 )
        {
            modify Session-Limit[1] = 240
        }
    }
```

Results in:

```
    Session-Limit[0] = 10
    Session-Limit[1] = 240
```

- Suppose that:

```
NAS-IP-Address = “192.168.1.2”
NAS-Identifier = “fred”
Port-Limit = 23
```

The commands:

```
    if ( (NAS-IP-Address = "192.168.1.2") &&
         ((NAS-Identifier = “jack”) || (Port-Limit > 20))
       )
    {
        exit “NAK”
    }
```

Results in:

Request being rejected.

delete Command

1. Syntax

   `delete <attr-spec>`

2. Parameters

   <*attr-spec*> is an attribute specification, see“ Attribute Specifications” on page 120. The use of instance specification, including '*' is allowed. The use of attribute functions is **not** permitted.

   The default instance for <*attr-spec*> is “last”.

3. Operation

   Deletes the named instance from the request. All instances may be deleted by using the '*' instance specifier.

   If <*attr-spec*> refers to an instance that is not present, no instances will be deleted. A no-such-instance run-time error is NOT generated in this case.

4. Examples

- Suppose that:

```
NAS-Port = 2
Reply-Message = "Hello, world!"
Reply-Message = "So long"
NAS-IP-Address = "2.3.4.5"
```

The command:

```
   delete Reply-Message[*]
```

Results in:

```
   NAS-Port = 2
   NAS-IP-Address = "2.3.4.5"
```

- Suppose that:

```
NAS-Port = 2
Reply-Message = "Hello, world!"
Reply-Message = "So long"
NAS-IP-Address = "2.3.4.5"
```

The command:

```
   delete Reply-Message
```

Results in:

```
   NAS-Port = 2
   Reply-Message = "Hello, world!"
   NAS-IP-Address = "2.3.4.5"
```

- Suppose that:

```
NAS-Port = 2
Reply-Message = "Hello, world!"
Reply-Message = "So long"
NAS-IP-Address = "2.3.4.5"
```

The command:

```
   delete Reply-Message[0]
```

Results in:

```
   NAS-Port = 2
   Reply-Message = "So long"
   NAS-IP-Address = "2.3.4.5"
```

- Suppose that:

```
NAS-Port = 2
Reply-Message = "Hello, world!"
```

The command:

```
   delete NAS-IP-Address[*]
```

Results in:

```
NAS-Port = 2
Reply-Message = "Hello, world!"
```

- Suppose that:

  ```
  NAS-Port = 2
  Reply-Message = "Hello, world!"
  ```

  The command:

  ```
  delete NAS-IP-Address[0]
  ```

  Results in:

  ```
  NAS-Port = 2
  Reply-Message = "Hello, world!"
  ```

- Suppose that:

  ```
  NAS-Port = 2
  Reply-Message = "Hello, world!"
  ```

  The command:

  ```
  delete NAS-IP-Address[last]
  ```

  Results in:

  ```
  NAS-Port = 2
  Reply-Message = "Hello, world!"
  ```

insert Command

1. Syntax

   `insert <attr-spec> = <value-expr>`

2. Parameters

   <attr-spec> is an attribute specification, see“ Attribute Specifications” on page 120. The use of instance specification, including “begin” and “last”, is allowed. The use of attribute functions is **not** allowed.
   The default instance for *<attr-spec>* is “last”.

   <value-expr> is a value expression.
   <value-expr> may use attribute specification. The use of instance specification, including “last”, is allowed. The use of attribute functions is allowed.
   If *<value-expr>* refers to an attribute, the default instance is “last”.
   If *<value-expr>* refers to an instance that is not present, a no-such-instance run-time error occurs. See “Error Handling” on page 156 for details.

   The types of *<attr-spec>* and *<value-spec>* must be compatible. See “Type Compatibility” on page 140 for details.

3. Operation

   Insert <`attr-spec`> into the request, having the value of <`value-expr`>. See "Tagged Attributes and the Advanced Policy Language" on page 128 for how tags are handled.

   If <`attr-spec`> refers to an instance that is not present, the location will be the end the list.

   If <`attr-spec`> refers to a tagged attribute (tag-int or tag-str) and <*value-spec*> is not a tagged value, the tag for the inserted attribute will be set to 0.

   If <`attr-spec`> refers to an attribute that is not tagged and <`value-spec`> is a tagged value, the tag is ignored.

   The instance location specified by <`attr-spec`> indicates the desired target location for the inserted instance. The algorithm used is "final opportunity", as opposed to "earliest opportunity". This means inserting "last" is the same as inserting at the end and inserting instance n occurs just before the already-present instance n (or the end if instance n is not already present).

4. Examples

- Suppose that:

```
NAS-Port = 2
Reply-Message = "message#1"
Reply-Message = "message#2"
NAS-IP-Address = "2.3.4.5"
```

  The command:

```
   insert Reply-Message = Reply-Message
```

  Results in:

```
   NAS-Port = 2
   Reply-Message = "message#1"
   Reply-Message = "message#2"
   NAS-IP-Address = "2.3.4.5"
   Reply-Message = "message#2"
```

- Suppose that:

```
NAS-Port = 2
Reply-Message = "message#1"
Reply-Message = "message#2"
NAS-IP-Address = "2.3.4.5"
```

  The command:

```
   insert Reply-Message[0] = "a new message"
```

  Results in:

```
   NAS-Port = 2
   Reply-Message = "a new message"
   Reply-Message = "message#1"
   Reply-Message = "message#2"
```

```
NAS-IP-Address = "2.3.4.5"
```

- Suppose that:

  ```
  NAS-Port = 2
  Reply-Message = "message#1"
  Reply-Message = "message#2"
  NAS-IP-Address = "2.3.4.5"
  ```

  The command:

  ```
  insert Reply-Message[1] = "a new message"
  ```

  Results in:

  ```
  NAS-Port = 2
  Reply-Message = "message#1"
  Reply-Message = "a new message"
  Reply-Message = "message#2"
  NAS-IP-Address = "2.3.4.5"
  ```

- Suppose that:

  ```
  NAS-Port = 2
  Reply-Message = "message#1"
  Reply-Message = "message#2"
  NAS-IP-Address = "2.3.4.5"
  ```

  The command:

  ```
  insert Reply-Message[2] = "a new message"
  ```

  Results in:

  ```
  NAS-Port = 2
  Reply-Message = "message#1"
  Reply-Message = "message#2"
  Reply-Message = "a new message"
  NAS-IP-Address = "2.3.4.5"
  ```

- Suppose that:

  ```
  NAS-Port = 2
  Reply-Message = "message#1"
  Reply-Message = "message#2"
  NAS-IP-Address = "2.3.4.5"
  ```

  The command:

  ```
  insert Reply-Message[last] = "a new message"
  ```

  Results in:

  ```
  NAS-Port = 2
  Reply-Message = "message#1"
  Reply-Message = "message#2"
  NAS-IP-Address = "2.3.4.5"
  Reply-Message = "a new message"
  ```

- Suppose that:

  NAS-Port = 2
  NAS-IP-Address = "2.3.4.5"

  The command:

  ```
  insert Reply-Message[begin] = "Hello, world!"
  ```

  Results in:

  ```
  Reply-Message = "Hello, world!"
  NAS-Port = 2
  NAS-IP-Address = "2.3.4.5"
  ```

- Suppose that:

  `Reply-Message = "hello"`

  The command:

  ```
  insert Tunnel-Password = Reply-Message
  ```

  Results in:

  ```
     Reply-Message = "hello"
     Tunnel-Password = :0:"hello"
  ```

- Suppose that:

  `Tunnel-Password = :2:"abc"`

  The command:

  ```
  insert Reply-Message = Tunnel-Password
  ```

  Results in:

  ```
  Tunnel-Password = :2:"abc"
  Reply-Message = "abc"
  ```

- Suppose that:

  `Tunnel-Password = :2:"abc"`

  The command:

  ```
  insert Tunnel-Password = :3:"def"
  ```

  Results in:

  ```
  Tunnel-Password = :2:"abc"
  Tunnel-Password = :3:"def"
  ```

- Suppose that:

  `Reply-Message = "hello"`

  The command:

  ```
  insert Reply-Message = :3:"def"
  ```

  Results in:

```
Reply-Message = "hello"
Reply-Message = "def"
```

- Suppose that:

  ```
  Reply-Message = "abc"
  ```

  The command:

  ```
  insert NAS-Port = count( Reply-Message[*] )
  ```

  Results in:

  ```
  Reply-Message = "abc"
  NAS-Port = 1
  ```

modify Command

1. Syntax

   modify <*attr-spec*> = <*value-spec*>

2. Parameters

   <*attr-spec*> is an attribute specification, see" Attribute Specifications" on page 120. The use of instance specification, including "last", is allowed. The use of attribute functions is **not** allowed.
   The default instance for <*attr-spec*> is "last".
   If <*attr-spec*> refers to an instance that is not present, a no-such-instance run-time error is generated. See "Error Handling" on page 156 for more details.

   <*value-spec*> is a value specification.
   <*value-spec*> may use attribute-specifications. The use of instance specification, including "last", is allowed. The use of attribute functions is allowed.
   If <*value-spec*> refers to an attribute, the default instance for <*value-spec*> is "last".
   If <*value-spec*> refers to an instance that is not present, a no-such-instance run-time error is generated. See "Error Handling" on page 156 for more details.

   The value types for <*attr-spec*> and <*value-spec*> must be compatible. See "Type Compatibility" on page 140 for details.

3. Operation

   Modifies <*attr-spec*> to have the value of <*value-spec*>. See "Tagged Attributes and the Advanced Policy Language" on page 128 for how tags are handled.

   If <*attr-spec*> refers to a tagged attribute (tag-int or tag-str) and <*value-spec*> is a tagged value, the tag of <*attr-spec*> is not modified. Only the value of <*attr-spec*> is modified.

4. Examples

- Suppose that:

  ```
  Reply-Message = "123"
  Reply-Message = "456"
  ```

  The command:

```
modify Reply-Message = "abc"
```

Results in:

```
Reply-Message = "123"
Reply-Message = "abc"
```

- Suppose that:

```
Reply-Message = "123"
Reply-Message = "456"
```

The command:

```
modify Reply-Message = Reply-Message[0]
```

Results in:

```
Reply-Message = "123"
Reply-Message = "123"
```

- Suppose that:

```
NAS-Identifier = "abc.def.ghi"
```

The command:

```
modify NAS-Identifier = "wxyz"
```

Results in:

```
NAS-Identifier = "wxyz"
```

- Suppose that:

```
Tunnel-Password = :2:"abc"
```

The command:

```
modify Tunnel-Password = "def"
```

Results in:

```
Tunnel-Password = :2:"def"
```

- Suppose that:

```
Tunnel-Password = :2:"abc"
```

The command:

```
modify Tunnel-Password = :4:"ghi"
```

Results in:

```
Tunnel-Password = :2:"ghi"
```

- Suppose that:

```
Reply-Message = "hello"
Tunnel-Password = :17:"abc"
```

The command:

```
modify Reply-Message = Tunnel-Password
```

Results in:

```
Reply-Message = "abc"
Tunnel-Password = :17:"abc"
```

- Suppose that:

  ```
  Reply-Message = "hello"
  Tunnel-Password = :17:"abc"
  ```

  The command:

  ```
  modify Tunnel-Password = Reply-Message
  ```

  Results in:

  ```
  Reply-Message = "hello"
  Tunnel-Password = :17:"hello"
  ```

- Suppose that:

  ```
  NAS-Port = 7
  Reply-Message = "abc"
  Reply-Message = "def"
  ```

  The command:

  ```
  modify NAS-Port = count( Reply-Message[*] )
  ```

  Results in:

  ```
  NAS-Port = 2
  Reply-Message = "abc"
  Reply-Message = "def"
  ```

- Suppose that:

  ```
  Reply-Message = "abc"
  Reply-Message = "def"
  ```

  The command:

  ```
  modify Reply-Message[0] = Reply-Message[1]
  ```

  Results in:

  ```
  Reply-Message = "def"
  Reply-Message = "def"
  ```

exit Command

1. Syntax

   exit "<*event-name*>"

2. Parameters

   <*event-name*> must be a quoted string.

   <*event-name*> must specify an event that is defined. There are a number of predefined events. Additional events may be defined in the FSM file using "%event <*name*>" syntax.

See “Events” on page 258 for more detailed information on finite state machine events.

Event names are case-insensitive (MyEvent is the same as MYEVENT).

3. Operation

   Terminates evaluation of the policy and returns the named event to the finite state machine. See “FSM Interaction” on page 156 for more details.

   The use of an undefined event name results in an undefined-event load-time error. See “Error Handling” on page 156 for more details.

4. Examples

```
exit “ACK”
```

log Command

1. Syntax

   log "<*log-level*>" "<*log-message*>"
   log "<*log-level*>" "<*log-message*>", <*attr-spec*>
   log "<*log-level*>" "<*log-message*>", <*attr-spec*>, ... <*attr-spec*>

2. Parameters
   <*log-level*> must be a quoted string.
   <*log-level*> must be a log-level type:
   `ALERT, CRITICAL, ERROR, WARNING, NOTICE, INFO`
   <*log-level*> is case-insensitive ("`ERROR`" same as "`Error`").
   <*log-message*> must be a quoted string.

   Multiple instances for <*attr-spec*> are allowed and cause all named instances to be reported in the logfile, see“ Attribute Specifications” on page 120. The use of instance specification, including “last” and “*”, is allowed. The default instance specification for the attributes in the log command is “last”.

   If <*attr-spec*> refers to an instance that is not present, this will be indicated in the logfile output. A no-such-instance run-time error will NOT be generated in this case.

3. Operation

   Causes a message to be written to the logfile.

   When attributes are specified, they are reported one value per line in the logfile. Multiple instances are reported one value per line as well.

   All log output lines will include the file/line location of the log command that generated the message.

   All log output will be generated using the standard logging functions which puts a timestamp on the output line.

4. Example

```
log "Warning" "This user should not come in through this NAS",
     User-Name, NAS-IP-Address
```

Attribute Functions

- `count` **function**

  1. Syntax

     count (<*attr-spec*>)

  2. Parameters

     <*attr-spec*> specifies an attribute (possibly including vendor or instance), see“ Attribute Specifications” on page 120.
     The default instance for <*attr-spec*> is '*'.

  3. Operation

     Returns an integer value that indicates the number of instances.

     If <*attr-spec*> refers to instance '*', then count() yields the total number of instances of <*attr-spec*> that are present.

     If <*attr-spec*> refers to a specific instance that is present, then count() yields the value 1.

     If <*attr-spec*> refers to an instance that is not present, then count() yields the value 0. A no-such-instance run-time error is NOT generated in this case.

- `length` **function**

  1. Syntax

     length (<*attr-spec*>)

  2. Parameters

     <*attr-spec*> specifies an attribute (possibly including vendor or instance) which must be of type String, Tag-Str, or Octets, see“ Attribute Specifications” on page 120.
     The default instance for <*attr-spec*> is 'last'.

  3. Operation

     Returns an integer value that indicates the number characters in the string attribute. For a Tag-Str attribute it does not include the tag octet. See “Tagged Attributes and the Advanced Policy Language” on page 128 for how tags are handled. For a TLV attribute, the number of subattributes is returned.

     If <*attr-spec*> refers to an instance that is not present, then a no-such-instance run-time error is generated. See “Error Handling” on page 156 for more details.

- `substr` **function**

  Using 'offset' **keyword**

  1. Syntax

     substr (<attr-spec> offset <*start*>)
     substr (<*attr-spec*> offset <*start*> length <*number*>)

2. Parameters

   <*attr-spec*> specifies an attribute (possibly including vendor or instance) which must be of type String, Tag-Str, or Octets, see“ Attribute Specifications” on page 120.
   The default instance for <*attr-spec*> is 'last'.
   <*start*> is the offset from the beginning of the string to the first character of the desired substring. It must be a non-negative integer constant.
   <*number*> is the optional length of the desired substring. It must be a non-negative integer constant.
   If “length <*number*>” is not present then the length defaults to the remainder of the string.

3. Operation

   Returns the requested substring, with same type as the source. See “Tagged Attributes and the Advanced Policy Language” on page 128 for how tags are handled.

   When the length is not specified the remainder of the string value is used.

   If the offset is off the end of the string then substr returns an empty string.

   For example:

   ```
   insert Reply-Message = "a string of characters"
   substr( Reply-Message offset 0 length 8 )
   ```

   Returns the string: “`a string`”.

   ```
   insert Reply-Message = "a string of characters"
   substr( Reply-Message offset 16 length 82 )
   ```

   Returns the string: “`acters`”.

   ```
   insert Reply-Message = "a string of characters"
   substr( Reply-Message offset 12 )
   ```

   Returns the string: “`characters`”.

   ```
   insert Reply-Message = "a string of characters"
   substr( Reply-Message offset 32 )
   ```

   Returns the empty string: “”.

   If <*attr-spec*> refers to an instance that is not present, then a no-such-instance run-time error is generated. See “Error Handling” on page 156 for more details.

Using 'before' **keyword**

1. Syntax

   substr (<*attr-spec*> before "<*before-string*>")
   substr (<*attr-spec*> before last "<*before-string*>")

2. Parameters

   <*attr-spec*> specifies an attribute (possibly including vendor or instance) which must be of type String, Tag-Str, or Octets, see“ Attribute Specifications” on page 120.
   The default instance for <*attr-spec*> is 'last'.

<*before-string*> must be a quoted string constant.

3. Operation

   Returns the requested substring with same type as the source. See “Tagged Attributes and the Advanced Policy Language” on page 128 for how tags are handled.

   The substring will start from the beginning of the string up to but not including the first occurrence of <*before-string*> for '`before`'.

   The substring will start from the beginning of the string up to but not including the last occurrence of <*before-string*> for '`before last`'.

   When <*before-string*> is not found the entire string is returned.

   For example:

   ```
   insert Reply-Message = "a string of characters"
   substr( Reply-Message before " of" )
   ```

   Returns the string: “`a string`”.

   ```
   insert Reply-Message = "a string of characters"
   substr( Reply-Message before last " " )
   ```

   Returns the string: “`a string of`”.

   ```
   insert Reply-Message = "a string of characters"
   substr( Reply-Message before "not-there" )
   ```

   Returns the entire string: “`a string of characters`”.

   If <*attr-spec*> refers to an instance that is not present, then a no-such-instance run-time error is generated. See “Error Handling” on page 156 for more details.

Using 'after' **keyword**

1. Syntax

   substr (<*attr-spec*> after "<*after-string*>")
   substr (<*attr-spec*> after last "<*after-string*>")

2. Parameters

   <*attr-spec*> specifies an attribute (possibly including vendor or instance) which must be of type String, Tag-Str, or Octets, see“ Attribute Specifications” on page 120.
   The default instance for <*attr-spec*> is 'last'.
   <*after-string*> must be a quoted string constant.

3. Operation

   Returns the requested substring with same type as the source. See “Tagged Attributes and the Advanced Policy Language” on page 128 for how tags are handled.

   The substring will start following the first occurrence of <*after-string*> for '`after`'.

   The substring will start following the last occurrence of <*after-string*> for '`after last`'.

When <*after-string*> is not found the empty string is returned.

For example:

```
insert Reply-Message = "a string of characters"
substr( Reply-Message after " of" )
```

Returns the string: "` characters`".

```
insert Reply-Message = "a string of characters"
substr( Reply-Message after last " " )
```

Returns the string: "`characters`".

```
insert Reply-Message = "a string of characters"
substr( Reply-Message after "not-there" )
```

Returns the empty string: "".

If <*attr-spec*> refers to an instance that is not present, then a no-such-instance run-time error is generated. See "Error Handling" on page 156 for more details.

- `tolower` **function**

  1. Syntax

     tolower (<*attr-spec*>)

  2. Parameters

     <*attr-spec*> specifies an attribute (possibly including vendor or instance) which must be of type String, Tag-Str, or Octets, see" Attribute Specifications" on page 120.
     The default instance for <*attr-spec*> is 'last'.

  3. Operation

     Returns the string value converted to lowercase with same type as the source. See "Tagged Attributes and the Advanced Policy Language" on page 128 for how tags are handled.

     If <*attr-spec*> refers to an instance that is not present, then a no-such-instance run-time error is generated. See "Error Handling" on page 156 for more details.

- `toupper` **function**

  1. Syntax

     toupper (<*attr-spec*>)

  2. Parameters

     <*attr-spec*> specifies an attribute (possibly including vendor or instance) which must be of type String, Tag-Str, or Octets, see" Attribute Specifications" on page 120.
     The default instance for <*attr-spec*> is 'last'.

  3. Operation

     Returns the string value converted to uppercase with same type as the source. See "Tagged Attributes and the Advanced Policy Language" on page 128 for how tags are handled.

If <*attr-spec*> refers to an instance that is not present, then a no-such-instance run-time error is generated. See "Error Handling" on page 156 for more details.

Error Handling

This section describes how error conditions are handled by the advanced policy engine.

There are two types of errors: load-time errors and run-time errors.

1. Load-time Errors

   The following load-time errors are defined:

   syntax error
   type mismatch
   invalid instance specification
   invalid integer value
   invalid IP address constant
   invalid tag value
   unknown named value
   undefined event
   undefined value

   When a load-time error occurs, the decision file is considered invalid; it is not loaded. A message in the logfile will indicate the location of the problem by file and line number. An ERROR event is returned to the state machine. See "FSM Interaction" on page 156 for more details.

2. Run-time Errors

   The following run-time errors are defined:

   no such instance

   When a run-time error occurs, the policy evaluation is terminated. A message in the logfile will indicate the location of the problem by file and line number. An ERROR event is returned to the state machine. See "FSM Interaction" on page 156 for more details.

FSM Interaction

This section describes how the advanced policy engine interacts with the finite state machine.

1. Invoking Policies

   Policies are invoked using the POLICY action in the FSM. The `Xstring` parameter uses a URL-like syntax to specify the policy to be invoked.

   Example:

```
decisionfile://MyPolicy.policy
```

See "Calling Decision Files" on page 159 for more details on invoking policies.

When a policy is invoked, it is evaluated.

When a policy is evaluated, it may return an event to the state machine to direct the subsequent processing of a request.

2. Returning Events

   There are several ways to return an event to the state machine:

   `exit` Command
   Default Event
   Error Condition

   - `exit` Command

     Using the exit command causes the evaluation of the policy to be terminated. The specified event is returned to the state machine.

   - Default Event

     If evaluation of a decision file reaches the end without encountering an exit command, the default event is returned to the state machine.

     The default event is ACK.

   - Error Conditions

     When an error occurs, an ERROR event is returned to the state machine.

Internal Attributes

The following Interlink and Merit specific attributes are useful in policy conditions or replies.

Attribute	Description
`Interlink-Packet-Code`	An integer that indicates what type of RADIUS message has been received: either 1 (Access-Request) or 4 (Accounting-Request).
`Interlink-Proxy-Action`	A string (normally determined by information in the Access-Request or Accounting-Request) that indicates the name of the starting event in the FSM when the RAD-Series Server receives a RADIUS message. The default value can be preempted by modifying the `request-ingress.grp` decision file to determine the start event.
`Interlink-Proxy-Target`	A string that identifies the proxy target host system. The proxy target host system must be listed in the RAD-Series Server's clients file.
`Interlink-Reply-Status`	An integer that contains an FSM code representing the reply status: • `ACK` -- Access-Accept or Accounting-Response • `NAK` -- Access-Reject • `ACC_CHAL` -- Access-Challenge This attribute is used to preserve the reply status of a request while applying reply post-processing policy. This attribute is used to preserve the reply status of a proxy response while applying proxy receive response policy.
`Interlink-Request-Type`	A string that identifies the request type: • "REQUEST" -- a normal request • "CONTINUATION" -- a continuation of an in-progress EAP conversation
`User-Id`	After the RAD-Series Server parses the NAI (userid@realm), it assigns the userid value to this attribute.
`User-Realm`	After the RAD-Series Server parses the NAI (userid@realm), it assigns the realm value to this attribute.
`Time-of-Day`	A string containing the time of day the request was received. It uses a 24 hour clock in *`hh:mm`* format.
`Day-Of-Week`	An integer representing the day of the week the request was received, where 0 represents Sunday and 6 represents Saturday.
`Date-Time`	A string containing the date and time the request was received. It uses a 24 hour clock in *`yyyy:mm:dd:hh:mm`* format.

Calling Decision Files

Once policies are stored in decision files, call the decision files at the appropriate points in the `radius.fsm` file, `authfile`, or user profiles.

From radius.fsm

The `POLICY` action should be called in the FSM whenever you wish to use Advanced Policy functions, such as Dynamic Access Control. Use the `Xstring` parameter to pass a decision file name to the `POLICY` action. The `Xstring` value must be no more than 63 characters long.

Event POLICY *Next-state* Xstring=decisionfile://*Filespec*

Where *Filespec* is the optional path relative to the configuration directory and the file name.

The `POLICY` action will process the decision file based on the request and reply accordingly. See “User/Realm policy” on page 113 for more details.

You must specify the decision file to use each time the `POLICY` action is called.

Note: If the `POLICY` action occurs before the user’s profile is retrieved, reply items from the decision file are superseded by any duplicate attributes in a user profile. Conversely, if `POLICY` occurs after the user’s profile is retrieved, the user's reply items will be superseded by the policy group reply items.

The `DAC.fsm` and `DNIS.fsm` files distributed with the RAD-Series Server already call the corresponding decision file at the appropriate point in the RAD-Series Server’s process. You need only make implementation-specific changes. See “Modifying the FSM for DNIS” on page 162 and “Modifying the FSM for DAC” on page 161 for details.

From authfile

Place a `Policy-Pointer` in a LDAP realm data store naming the full path to the decision file containing the group authorization policies. Enclose the pointer in double or single quotes.

```
realm ProLDAP “comment”
{
      Policy-Pointer = “decisionfile://filespec”
      Directory . . .
}
```

Where *filespec* is the optional path relative to the configuration directory and the required file name.

The `POLICY` action will process the decision file based on the request and reply accordingly. See “User/Realm policy” on page 113 for more details.

From user profiles

In the default users file or realm files, place a `Policy-Pointer` as a check or reply item naming the optional path relative to the configuration directory and filename of the decision file prefixed with the string "decisionfile://". Enclose the pointer in double or single quotes.

```
carl Password=carl, Policy-Pointer = "decisionfile://filespec"
```

or

```
fred Password = fred
     Policy-Pointer = "decisionfile://filespec"
```

Where `filespec` is the optional path relative to the configuration directory and the file name.

The `POLICY` action will process the decision file based on the request and reply accordingly. See "User/Realm policy" on page 113 for more details.

Dynamic Access Control

Dynamic Access Control (DAC) enables you to provide different levels of network access to the same users depending on:

- Access periods
- Account and password expiration date and time

Dynamic Access Control utilizes three Interlink-specific attributes to check values in user requests:

Attribute	Description
Time-of-Day	A string containing the time of day the request was received. It uses a 24 hour clock in *hh:mm* format.
Day-Of-Week	An integer representing the day of the week the request was received, where 0 represents Sunday and 6 represents Saturday.
Date-Time	A string containing the date and time the request was received. It uses a 24 hour clock in *yyyy:mm:dd:hh:mm* format.

NOTE: This three attributes are in local time not UTC.

Modifying the FSM for DAC

Do the following to modify `radius.fsm` to support Dynamic Access Control.

1 In a text editor, copy the contents of `DAC.fsm`, by default found in `/etc/opt/aaa`.

2 Open `radius.fsm` (in the same directory) and paste the contents of `DAC.fsm` over `radius.fsm`. Replace the complete text.

3 If you’re using a different decision file than the supplied `DAC.grp` decision file, change the `CheckDAC` state so that the POLICY action calls your DAC decision file. For example:

```
CheckDAC:
   *.*.ACK    POLICY   AuthWait    Xstring=decisionfile://DAC.grp
```

4 Save and close `radius.fsm`.

DAC Decision File

Edit the `DAC.grp` decision file to define your DAC policies. There are sample entries for most groups you will need. Modify each group to reflect your time-based policies. For example:

```
# Daytime Access Check

if ( (Access-Group[0] = "daytime") &&
     ((Time-Of-Day[0] >= "06:00") && (Time-Of-Day[0] <= "20:00")) )
{
    insert Reply-Message = "Daytime access allowed"
    exit "ACK"
}
```

Note: The Reply-Message reply item attribute may not be returned if the user is authenticated with a tunneled EAP method.

Comment out any condition you don’t need by placing a pound sign (#) before each line.

The last line should remain so any user that does not match one of the conditions will get rejected.

If you rename this file, edit `radius.fsm` so that the `CheckDAC` state `Xstring` parameter points to the correct filename.

Keep this file in the RAD-Series Server’s configuration directory, by default `/etc/opt/aaa`.

DNIS Routing

In a typical DNIS routing scheme, requests are handled according to the `Calling Station-Id` and `Called-Station-Id` attributes. The `POLICY` action matches the `Calling-Station-Id` and `Called-Station-Id` attribute values in the Access-Request to the conditions defined in the DNIS decision file and returns the matching policy group reply items and the FSM events `Forward` and `Abandon`.

The required events and states are defined in the `DNIS.fsm` file delivered with the RAD-Series Server.

Modifying the FSM for DNIS

Do the following to modify `radius.fsm` to support DNIS routing.

1. In a text editor, copy the contents of `DNIS.fsm`, by default found in /etc/opt/aaa.
2. Open `radius.fsm` (in the same directory) and paste the contents of `DNIS.fsm` over `radius.fsm`. Replace the complete text.
3. If you're using a different decision file than the supplied `DNIS.grp` decision file, change the Start3 state so that the POLICY action calls your DNIS decision file. For example:

```
Start3:

    *.*.AUTHEN          POLICY   Start4   Xstring=decisionfile://DNIS.grp
    *.*.AUTH_ONLY       POLICY   Start4   Xstring=decisionfile://DNIS.grp
    *.*.AUTHENTICATE    POLICY   Start4   Xstring=decisionfile://DNIS.grp
    *.*.ACCT            POLICY   Start4   Xstring=decisionfile://DNIS.grp
    *.*.LAS_ACCT        POLICY   Start4   Xstring=decisionfile://DNIS.grp
```

4. Modify the `Start4` state so that `Xstring` points to the fully qualified domain name, IPv4 address or IPv6 address of the server to which you are forwarding requests and must be listed in the RAD-Series Server's clients file. The clients file entry is needed to get the shared secret to use.

```
Start4:

    *.*.Forward     RAD2RAD     Start4a    Xstring=192.168.0.0
```

5. Save and close `radius.fsm`.

DNIS Decision File

Edit the `DNIS.policy` decision file to reflect your station-based access policies. For example, change the Calling-Station and Called-Station numbers in the Controlled Access condition:

```
# Controlled Access

if ( ( count(Calling-Station-Id[0]) > 0 ) &&
     ( Calling-Station-Id[0] = "5551234567" ) )
    {
        exit "Forward"
   }
}
if ( ( count(Called-Station-Id[0]) > 0 ) &&
     ( Called-Station-Id[0] = "5551236543" ) )
    {
        exit "Forward"
   }
}
```

Additional conditions and attributes can be added to this policy if your policies require that other conditions be met.

Comment out any condition you don't need by placing a pound sign (#) before each line.

The last line should remain unchanged so it can authenticate any user that does not match one of the other conditions.

If you rename this file, edit `radius.fsm` so that the `Start3` state `Xstring` parameter points to the correct filename.

Keep this file in the RAD-Series Server's configuration directory, by default `/etc/opt/aaa`.

Configuration Files

About Configuration Files

This section provides reference information for all the files that you may need to edit to maintain the RAD-Series Server configuration. Some of this information is also in the MAN pages distributed with the server. By default the man pages are located in `/opt/share/aaa/man`.

Throughout this section, required parameters are marked in bold face type and optional ones in a normal face. In addition, when a variable should be replaced with a specific value by the administrator, the variable will appear in *italics*. For example:

```
realm ProLDAP "comment"
{
      Directory "directory comment"
      {
            URL "ldap://192.168.3.8:389"
            SearchBase "ou=devision,dc=com"
      }
}
```

File Format

Entries

All configuration files consist of one or more entries. These entries include parameters that define a configuration item for the RAD-Series Server: a client, a user profile, a realm, and others.

A parameter (field) may contain one or more A-V pairs. See page 167 for a description of A-V pair syntax.

Entries that define grouped configuration objects use a block format:

```
block-name
{
      Parameter value
      Parameter "value string"
      . . .
}
```

Enclose string values within single or double quotes if the value contains spaces or special characters other than underscore (_) and dash (-). There is no difference between using single or double quotes.

Line Lengths

The `radius.fsm` file has a maximum line length of 256 characters. The users files have a configurable maximum line length of 16,384 characters. All other configuration files (`authfile, clients`, etc) have a maximum line length of 4095 characters.

Delimiters

Fields within entries are delimited by whitespace (one or more spaces or tabs).

You may also use comma-space to delimit A-V pair lists in users files and .fsm files.

Comment Lines

Comment lines are indicated by a pound sign ('#') character as the first character following any leading whitespace. End-of-line (trailing) comments may be present on a configuration data line following a '#' after all configuration data. The RAD-Series Server ignores all comment lines and trailing comments, as well as empty lines and entirely-blank lines. All lines are counted for the purpose of reporting errors, warnings, or changes.

File Location

By default, configuration files are located in:

```
/etc/opt/aaa
```

MAN pages are located in:

```
/opt/share/aaa/man
```

Commonly Used Files

The configuration files you'll most commonly work with are:

- `aaa.config` defines all RAD-Series Server properties.
- `authfile` defines realm datastores.
- `clients` defines client attributes and shared secret.
- decision files contain advanced policy information for user authorization and session control based on a scripting language. See "Using Advanced Policy" on page 110 for information about creating decision files.
- `dictionary` and `dictionary.*` files define all attributes and values recognized by the RAD-Series Server. These A-V pairs convey information in requests and responses. This file also contains definitions for all the authentication types that the server recognizes.There is now a `dictionary.custom` file for customer additions.

  Please do not edit the other dictionary files as they will be no longer be updated if modified when an RAD-Series Server upgrade is done.
- `EAP.authfile` defines realm authentication actions.
- `las.conf` defines internal Session Manager behavior. It enables session tracking and specifies some session timing values.
- `log.config` defines accounting message logging behavior.
- `radius.fsm` is the Finite State Machine table. It can be edited to reorder processing steps or call custom plug-ins.
- realm (`.users`) files contain user profile entries, including check/deny and reply items. Realm files are sometimes referred to as `.users` files, because they contain profiles for users in a single realm and may be given any descriptive name, such as the realm name. All realm files **must** have the extension `.users`.
- `users` defines user profiles which can be used for exceptions to the normal realm based configuration.The default `users` file contains only the test_user entry after an initial installation.
- `vendors` contains optional entries for vendor names and numbers and it maps external attributes to and from vendor-specific attributes.

Note: The `dictionary*, vendors, radius.fsm` and `log.config` files must be manually-edited. Furthermore, these files are read once at RAD-Series Server startup time and only `log.config` is re-read when the server receives a HUP signal.

Note: A decision file is loaded and cached when first referenced during run-time, and once cached is not re-read following a HUP.

Attribute-Value Pairs

The RAD-Series Server sends information in terms of attributes. When a message is exchanged among access devices and servers, one or more attributes and values are sent pairwise as an attribute-value pair (A-V pair).

```
attribute = value
```

Attributes are defined to be one of the following value types: IPv4 address, IPv6 address, ip46addr, string, vendor, tag string, tag integer, date, integer, string, unsigned8, unsigned16, signed32, signed64, unsigned64, IPv6 prefix and interface-id values. The attribute may take any of the supported, legal values defined for it in the `dictionary` files.

The RAD-Series Server supports most standard RADIUS attributes for session control and provisioning. Attribute names and their enumerated values are defined in the `dictionary` files. See the RADIUS RFC documents (2865, 2866, 2867, 2868, 2869, 3162, 4679, 4818, 5176, 5904, 6519, 6911, 6929, 6930) for a description of the attributes.

Note: The Reply-Message attribute is not supported as a reply item for users in EAP PEAP and EAP TTLS realms.

This section describes Interlink-specific configuration, RADIUS and session attributes with the files where they are most likely to be used.

Attribute-Value Pair Syntax (realm, users and .fsm)

When specifying A-V pairs in a realm file, `users` file or `.fsm` file entry, you should put a space before and after the equals (=) or not equal (!=) operator. Some other formats will work but are not guaranteed to be supported in future versions. An example is:

```
Simultaneous-Use = 3
```

Within a list of A-V pairs, A-V pairs are delimited by commas.

```
User-Name = msmith, Password = nopass, NAS-IP-Address = 192.168.6.59
```

Reply item A-V pairs in a .users file are listed on separate, indented lines.

```
User-Name = msmith, Password =pass, NAS-IP-Address = 192.168.6.59

    Session-Timeout = 60
    Idle-Timeout = 15
```

String values **must** be enclosed by single quote or double quote characters if they contain spaces or special characters other than underscore (_) and dash (-). There is no difference between using single or double quotes. Otherwise, the quotation marks are optional:

```
Deny-Message = “Access Denied”

Deny-Message = Access_Denied
```

IPv4 address values use the common dotted-quad notation.

```
NAS-IP-Address = 10.11.0.9
```

IPv6 address values use the customary representation of groups of 16-bit hexadecimal values separated by colons (:):

```
NAS-IPv6-Address = 2001:abc::123
```

Date Attribute Value Formats

Date type attributes can be entered in many formats. The rules are:

- When configuring a date attribute value, the values represent local time unless a “UTC” is appended to the time value string, in which case the values are interpreted as UTC.
- The “UTC” is case-insensitive.
- The “UTC” can immediately follow the *`hh:mm:ss`*, or can be placed after a delimiter.
- If the optional hh:mm:ss are not specified, the “UTC” can be placed immediately after the `yyyy-mm-dd` or `mmm-dd-yyyy`, or can be placed after a delimiter.
- The delimiters are space “ “, dash “-”, slash “/”, and colon “:”.
- The value must be enclosed in quotes if it contains a space, otherwise quotes are optional.
- Spaces are the only allowed delimiters which may precede or follow the value.

Format: *`mmm-dd-yyyy HH:MM:SS`* `UTC` or *`yyyy-mm-dd HH:MM:SS`* `UTC`

- ‘mmm’ is a three character month abbreviation (Jan, Feb, Mar, etc.).
- ‘mm’ is the two digit month (1-12).
- ‘dd’ is the day (1 to 31).
- ‘yyyy’ is the year expressed as four digits (2016).
- ‘HH’ is the two digit hour.
- ‘MM’ is the two digit minutes.
- ‘SS’ is the two digit seconds.
- “UTC” is an optional field that will override the default, which is the local time zone.
- The ‘HH:MM:SS’ is optional, if not present HH:MM:SS defaults to 00:00:00.

The following are valid date values:

```
Expiration = "Apr 17 2016"
Expiration = "Apr 18 2016 UTC"
Expiration = Apr-19-2016UTC
Expiration = "Apr-19-2016 10:32:59"
Expiration = "Apr-09-2016 10:32:59 utc"
Expiration = "2016-04-05 0:0:0"
Expiration = 2016-03-16
Expiration = 2016-03-17UTC
Expiration = 2016-03-18/UTC
Expiration = 2016-03-15/01:02:03
Expiration = 2016-03-15/04:05:06utc
Expiration = 2016-03-15/04:05:07/utc
Expiration = "2016-04-05 0:0:1 UTC"
Expiration = "2016-04-05 0:0:2utc"
Expiration = "2016-04-05 0:0:3 UTC  "
Expiration = " 2016-04-05 0:0:3 UTC "
```

The following example is a syntactically valid A-V pair list:

```
Password = “rock”, Service-Type = “Framed”, Nas-Port = 42
```

The following examples are **not** syntactically valid A-V pair lists since the attributes are not separated by both a comma and a space:

```
Password="rock"Service-Type="Framed"Nas-Port=12
```

```
Password = rock Service-Type = Framed Nas-Port = 123
```

Tagged Attributes

A RAD-Series Server message may include multiple attributes that are tagged to organize them into defined groups. Depending on its capabilities, a client or server will selectively use one set of tagged attributes. Tagged attributes may be used as check or reply items.

Tagged attributes follow the syntax:

`Attribute = :Tag:Value`

`Attribute` is the attribute name.

`Tag` is a unique integer (in the range 0-31) that provides a means of grouping attributes which refer to the same tunnel.

`Value` is the attribute value (in that set). A string value containing white space must be enclosed by double or single quotes:

```
Tunnel-Assignment-Id = :2:”Hello World”
```

For example, an Access-Accept may contain several different tunnel definitions as tagged attribute sets. `Tunnel-Type = :1:PPTP` indicates that PPTP is the Tunnel-Type in attribute set 1, which defines one type of tunnel that might be established for a user. A client which supports tagged attributes can selectively define the tunnel by using only the values belonging to one attribute set.

TLV Attribute Value Format

Consider a TLV attribute named “TLVx” with three subattributes: “Sub1” of type integer, “Sub2” of type TLV, and “Sub3” of type ipaddr. Subattribute Sub2, itself a TLV, has two subattributes: “Sub2a” of type string and Sub2b of type unsigned64.

The TLV value is specified as a comma-separated list of subattributes A-V pairs enclosed within curly braces, e.g.:

```
TLVx = { Sub1 = 1234, Sub3 = 192.168.1.3 }
```

TLVs can be nested, again with curly braces, e.g.:

```
TLVx = { Sub1 = 1234, Sub2 = { Sub2a='hello world', Sub2b =
   111222333444555666 }, Sub3 = 192.168.1.3 }
```

Note: As with other A-V pair configurations, the entire TLV A-V pair must be configured on a single line.

aaa.config

The `aaa.config` file contains user-defined entries for RAD-Series Server properties. These entries override the server's built-in defaults.

All entries are delimited by whitespace (tabs or spaces).

If a value contains spaces, use single or double-quote characters to enclose the value.

Including Files

Configuration data can be store in multiple text files and use the include to load them at RAD-Series Server startup. For each text file, add a one line entry to the `aaa.config` file that follows the format:

```
INCLUDE filename
```

If *`filename`* specifies a relative path, the RAD-Series Server will look for the file in the configuration directory.

Server Variable Syntax

RAD-Series Server variables override the server's built-in defaults.

To add a server variable, enter a line in `aaa.config` as:

```
variable = value
```

or

```
variable    value
```

Server variables are loaded into memory when the RAD-Series Server starts. Reload the server configuration by HUPing the server, if you change server variables in the `aaa.config` file.

Any space or tab characters before the variable or surrounding the equal sign character are ignored. The equal sign is optional

General Server Properties

Variable	Description
add-hostname-to-logfile-header (boolean)	If enabled (Yes), the RAD-Series Server will insert the server’s hostname in the logfile lines, following the timestamp. The parameter is also available as a command line switch, in which case it takes effect at the very beginning of the logfile. See “radiusd” on page 86. The default value is Disabled (no).
authenticate_as_computer (boolean)	If enabled (Yes), the RAD-Series Server will strip “host/” from the EAP-Identity sent by Microsoft clients configured to authenticate as computer on a wireless connection. When using TLS authentication, the remaining string is compared to the selected “`Client User Name Attribute`” field in the client certificate. When using PEAP/MSCHAPv2 authentication, the remaining string is used as the inner-realm `userid`. The default value is Enabled (Yes).
auth_failure_loglevel	The value is a single character which specifies the log level for reporting authentication failures in the server’s logfile. The acceptable values are 'I', 'A', 'C', 'E', 'N', and 'W'; representing `LOG_INFO, LOG_ALERT, LOG_CRIT, LOG_ERR, LOG_NOTICE,` and `LOG_WARNING,` respectively. The single character value may optionally be enclosed in single or double quotes, or be presented as a standalone single character. For example: `auth_failure_loglevel N` The parameter is optional and defaults to ‘I’ (INFO).
avpair_checking (boolean)	Whether or not the RAD-Series Server should perform sanity checks of A-V pairs in messages by checking to see if the flags indicate a valid A-V pair, and if it's a string type A-V pair, whether the string's use count is valid and print the diagnostic information. Values are on or off. This is for diagnostic purposes only and should not be used in production. The default value is off.
compress-logfile (integer)	This parameter indicates if the old logfile should be compressed when rolled over (by date or by size) to a new logfile. Enabled (Yes) is the default.

Variable	**Description**
CUI-Encryption-Secret	Configures the secret used for encryption of the real user identity to generate a Chargeable-User-Identity (CUI) when needed. The CUI- Encryption-Secret can be from 0 to 127 characters long. It can contain any printable character except for quotes and must be enclosed in quotes if there are spaces. If not configured or if configured as the empty string, then a default internal secret will be used to generate the CUI. The default value is an empty string. If a CUI is configured in a user’s profile, then this configured CUI will be used, and the user’s CUI will then change only as often as his configuration changes. If a CUI is not configured for a user, then the RAD-Series Server will generate a CUI when needed, and the user’s generated CUI will change weekly if no CUI-Encryption-Secret is configured and if one is configured then it changes weekly or as often as the CUI-Encryption-Secret is changed, whichever is sooner.
cwd (string)	Changes the current working directory of the RAD-Series Server to the specified path. This will be the location of any core files if the server aborts. This allows for overriding the radiusd –c option. The default value is the -c option value. If no -c option was used, then the default is the current working directory of the shell that started radiusd.
default_reply_holdtime (integer)	Number of seconds the RAD-Series Server holds a request after replying to it in case a retransmission of the response is necessary. The value should be twice the default retransmission period of the access devices involved. This does not apply to packets that are forwarded to another server. A value of zero invokes special behavior in which the REPLY AATV does not change the hold time for a request. This would cause all received authentication or accounting requests to be held for the full TTL (time-to-live), regardless of how short or how long the request took before being replied to. **Note:** Using the special value of zero or using a hold time greatly in excess of the retransmission policy of a NAS may cause the authentication and accounting queues to grow excessively large impacting performance. Refer to the global_acct_q.limit and global_auth_q.limit variables.Tailoring of this value should be influenced by the total and holding values reported on a per-request basis. The default value is 6 seconds.

Variable	Description
default_retry_limit (integer)	Maximum number of retransmissions allowed before a RETRY event occurs (a RETRY event is similar to a TIMEOUT event and is handled by the built-in default FSM table). The purpose of this is to catch an authentication request and perform some action when a certain number of retransmissions from an access device occur. The default is 0 and no limits are imposed. In particular, it may be useful to have a primary authentication RAD-Series Server deny access (using the FAIL AATV) before a backup server starts to authenticate, allowing the backup server to backup just the primary and not the whole AAA system. The default value is 0.
default-source-ipv4-address (ipv4 address)	Sets the default IPv4 address to use as the source IPv4 address when proxying an Access-Request or Accounting-Request. This parameter can be overridden by the “srcrip” parameter in the clients file. The default value is ‘0.0.0.0’ which means the operating system will assign the source IP address.
default-source-ipv6-address (ipv6 address)	Sets the default IPv6 address to use as the source IPv6 address when proxying an Access-Request or Accounting-Request. This parameter can be overridden by the “srcrip” parameter in the clients file. The default value is ‘[::]’ which means the operating system will assign the source IP address.
dns_address_aging (integer)	The base value (in seconds) used to periodically refresh DNS entries. To ensure that all the client entries don't expire at once, a designed-in randomness adds zero, twenty, forty, or sixty minutes to the base value to determine when a DNS entry should be refreshed. The default value is 3600 (1 hour).
dns_address_window (integer)	When a DNS entry for a configured client expires (needs refreshing), all other clients that will be refreshed within the specified number of seconds are refreshed immediately. The default value is 60 seconds.
dns_max_aliases (integer)	Maximum number of aliases per client. The default value is 3.

Variable	Description
Enable-MS-Username-Delimiter (boolean)	If `ON`, then the server will, when parsing a `User-Name` value, look first for a Microsoft-style NAI in the form realm\user or realm/user before looking for a userid@realm or plain userid (no realm) NAI. When the parameter is `ON`, the server will interpret the forward slash '/' and back slash ‘\’ as a Microsoft realm-user delimiter. When `OFF`, the server will interpret the forward slash ‘/’ as an allowed character in the userid portion of a userid@realm NAI and the bsck slash ‘\’ as invalid character in the userid portion of a userid@realm NAI. The parameter is optional and defaults to `ON`.
errorlog-level (string)	The levels of logfile messages to include in the errorlog file. 'A' represents LOG_ALERT, 'C' represents LOG_CRIT, 'E' represents LOG_ERR, 'W' represents LOG_WARNING, and 'N' represents LOG_NOTICE. The default is 'ACEWN' i.e. include all non LOG_INFO messages. This parameter is only relevant if writing to the errorlog file is enabled.
global_acct_q.limit (integer)	The maximum number of simultaneous accounting requests to be held on the accounting queue at any one time. This includes pending requests in progress plus requests which have been replied to but whose *`reply-holdtime`* has not yet expired. When this limit is reached, new requests are discarded (not replied to) with a message in the logfile. The same equation used for sizing the global_auth_q.limit may also be used for global_acct_q.limit. The default value is 40,000.

Variable	Description
global_auth_q.limit (integer)	The maximum number of active authentication requests to be held on the auth queue at any one time. This includes pending requests in progress plus requests which have been replied to but whose `reply-holdtime` has not yet expired. When this limit is reached, new requests are discarded (not replied to) with a message in the logfile. The auth queue must be sized correctly in order to buffer the arrival rate of access requests. If the arrival rate of requests is relatively steady then size the queue and hold time as follows: global_auth.q.limit/default_reply_holdtime > transaction rate For example, if global_auth_q.limit=1000, default_reply_holdtime=5, and transaction rate=100 authentications/second, then 1000/5 = 200 ==> 200 > 100 For the first five seconds the queue is filling up to 500 authentication requests. From that point on the requests are still arriving at 100/second, but the oldest requests are being released at 100/second. Now, suppose global_auth_q.limit=1000, default_reply_holdtime=5, and transaction rate=250 authentications/second, then 1000/5 = 200 ==> 200 < 250 For the first four seconds the queue is filling up to 1000 authentication requests. For the fifth second another 250 requests arrive, but there is no room in the queue. No requests will be released until the end of the fifth second, since that is the hold time. Either the global.auth_q.limit must be increased, or the default_reply_holdtime must be decreased. Note: When the authentication queue limit is exceeded, the RAD-Series Server stops responding to the radcheck command. The default value is 40000.
ipv6_enabled (boolean)	Enables the IPv6 support in the RAD-Series Server. When IPv6 is disabled the server will ignore any IPv6 configurations except for LDAP. See “IPv6 OperationIPv6 Operation” on page 269 for more information. The default value is off.
list_copy_limit (integer)	For customized RAD-Series Server configurations that accumulate A-V pairs or generate large responses. The default value is 1024.
logfile_line_len (integer)	The maximum length, in bytes, of the server's logfile lines. The minimum value for this parameter is 1024; the maximum is 16384, and the default is 4096. Lines longer than this are truncated

Variable	Description
maximum-errorlog-file-size (integer)	The maximum size in bytes of the server's errorlog file. The minimum value for this parameter is 65536; the maximum (default) is 2147483647. When errorlog reaches this size, it is rolled over.
maximum-output-file-size (integer)	The maximum size in bytes of the RAD-Series Server log file and the accounting log file. The minimum value for this parameter is 65536; the maximum is 2147483647. When a logfile reaches this size, it is "rolled over". For example, if the current logfile is logfile.20161201, then it is renamed to logfile_part01.20161201 and the new logfile is named logfile_part02.20161201. See "DESCRIPTION `roll logfile`" on page 96 and "DESCRIPTION `roll stream`" on page 96 for more details of the file naming. The default value is 2147483647.
ourhostname (string)	This variable is no longer used. It is accepted but ignored. The functionality of this parameter has been superseded by the 'ipaddr' variable in the radius_socket block, the default-ipv4-source-address, the default-ipv6-source-address, and the 'srcip' in the clients file.
proxy_udp_recv_buffer_size (integer)	The requested UDP buffer size for response packets to proxied requests. The minimum value is 8192 bytes; the maximum is 8388608 bytes. The default value is 0 which lets the operating system set the UDP receive buffer size. Note: The operating system may not honor the requested buffer size. A logfile message will display the actual buffer size allocated by the operating system.
radius_log_fmt (string)	This variable overrides the radiusd -l option to specify the logfile name format string to be used. The default value is "logfile.%Y%m%d"
recv_buffer_size (integer)	The maximum size in bytes for an individual inbound RADIUS message. The minimum value is 4096; the maximum is 65535. The default value is 16536. This property is primarily intended for supporting an access client that might transmit very large packets.
send_buffer_size (integer)	The maximum size in bytes for an outbound RADIUS packet. The minimum value is 4096. The default value is 16536. This property is primarily intended for supporting a customized RAD-Series Server configuration that might transmit very large packets. Limiting this variable to be the UDP MTU for the network will prevent excessively large packets from being forwarded (or replied to) in certain circumstances.

Variable	Description
subprocess-termination-SIGSEGV-timer (integer)	Set the timeout value, in seconds, for a SEGV signal sent to a subprocess. A value of 0 diables this timeout. If the timer expires before the subprocess teminates, the server will send a KILL signal (-9) to the subprocess. This will cause an immediate termination of the subprocess. The valid values are from 0 to 600 seconds. The default value is 20.
subprocess-termination-SIGTERM-timer (integer)	Set the timeout value, in seconds, for a TERM signal sent to a subprocess. A value of 0 diables this timeout. If the timer expires before the subprocess teminates, the server will send a SEGV signal to the subprocess. This should also produce a subprocess core which should help in finding the cause of the subprocess hang. The valid values are from 0 to 600 seconds. The default value is 5.
users_file_maxlen (integer)	Set the maximum line length for reading the users file and/or realm files. The values may range from 1024 to 16534 characters per line. The default value is 1024

Server Tracking Properties

Variable	Description
log_forwarding	Turns on or off logging of packets forwarded through the RAD-*Series* Server to another RADIUS server. This allows finer logging detail when tracking problems, at the expense of increased log file size. The variable can occur multiple times to select the set of options desired. Valid options are: • on - Turns on the logging of forwarded packets • off - Turns off all logging of forwarded packets • +digest - log sent forwarding digest turned on • –digest - log sent forwarding digest turned off • +dump - printing of packet contents in hexadecimal is turned on • –dump - printing of packet contents in hexadecimal is turned off • +vector - log initial forwarding digest turned on • –vector - log initial forwarding digest turned off • clear - is equivalent to –digest plus –dump The default value is off.
log_generated_request	Turns on or off the logging of internally generated packets when they are created and whlog_foren they reach their end-state. It is useful for a customized RAD-Series Server configuration that produces accounting requests based on internal state transitions rather than on externally delivered requests. Valid values are on or off. The default value is on.
packet_log	This variable matches a current request with an original request, which may occur when logging certain attributes in a request log. It is useful for tracking situations where a remote RAD-Series Server is responding with incorrect values and to investigate if an AATV is corrupting the current request. The variable can occur multiple times to select the set of options desired. Valid options are: • default - sets +current and +original • clear (or none) - removes all options • +abort - Turn on abort and core-dump if there is a mismatch • –abort - Turn off abort and core-dump if there is a mismatch • +both (or +comp) - Turn on comparison of A-V pairs if +current and +original are set • –both (or –comp) - Turn off comparison of A-V pairs • +current (or +cur) - Turn on report only from the modified request • –current (or +cur) - Turn off report only from the modified request • +original (or +orig) - Turn on report only from original request • –original (or –orig) - Turn off report only from original request The default value is +current and +original

Variable	Description
radcheck	This variable enables (or disables) certain reports produced by the radcheck command. The variable can occur multiple times to select the set of options desired. Valid options are: • default - Turns on +queues. • clear - Turns off all the radcheck options and clears the authentication queue counters. • none - Turns off all the radcheck options. • reset - Clears the authentication queue counters. • +mf - Show malloc and free statistics. • -mf - Do not show malloc and free statistics. • +packets - Show statistics about the number of octets/packets received, replied, forwarded, replies received, and redone. • –packets - Do not show statistics about the number of octets/packets received, replied, forwarded, replies received, and redone. • +queues - Show authentication and accounting queue information such as: number of unique requests, number of queue overflows, number of duplicate requests. If the number of accounting requests greatly exceeds the number of authentication requests and the NAS is not sending interim accounting, then a NAS/network configuration error is possible. • –queues - Do not show authentication and accounting queue information. • +timeouts - Show various timeout counters. • –timeouts - Do not show various timeout counters. The default value is +queues.

Variable	**Description**
reply_check	This variable specifies which attributes to check in a reply to ensure they are the same as in the forwarded request. Options allow you to specify the action to take when a mismatch occurs. The variable can occur multiple times to select the set of options desired. Valid options are: • first - Turn on check only the first match, turns off all • all - Turn on check all attributes for matches, turns off first • default - Clears all options, sets first and clears the list of attributes to check • none - Clears the list of attributes to check • clear - Clears all options and sets first • +abort - Turn on abort and core-dump if a check fails • –abort - Turn off abort and core-dump if a check fails • +dump - Turn on logging of the offending packet (in hex) • –dump - Turn off logging of the offending packet • +ignore - Turn on ignoring of responses which have a mismatch • –ignore - Turn off ignoring of responses which have a mismatch • +verbose - Turn on logging of good and bad values of each attribute • –verbose - Turn off logging of good and bad values of each attribute • <Attribute-name> - Add this specific attribute to list of checks The default value is none.

SNMP Server Properties

The SNMP support uses the block format to define variables. This block is read at RAD-Series Server startup time, and, unlike most of the other aaa.config blocks, is not re-read when the server processes a HUP signal. The block format looks like this:

```
iaaa.SNMP
{
     variable value
}
```

Variable	Description
Enabled (boolean)	When Enabled is set to 'on', the RAD-Series Server will automatically check the server for an SNMP master agent to communicate with, and the server can be monitored by an SNMP workstation. When set to 'off', the server will not communicate with an SNMP master agent and cannot be monitored by an SNMP workstation. Valid values are on or off. The default value is off.

EAP-TLS Server Properties

For the EAP-TLS, EAP-TTLS, and EAP-PEAP protocols you need to provide several files used for encryption and identification. The locations of these files is specified in the securities.path block. This block is read at RAD-Series Server startup time, and, unlike most of the other aaa.config blocks, is not re-read when the server processes a HUP signal. The block format looks like this:

```
EAP-TLS
{
     RSA-Certificate-Path            "<file_path>"
     RSA-Key-Path                    "<file_path>"
     DSA-Certificate-Path            "<file_path>"
     DSA-Key-Path                    "<file_path>"
     CA-Path                         "<file_path>"
     Random-Path                     "<file_path>"
     CRL-Path                        "<file_path>"
     ECDHE-Parameter-Curve           "<curve_name>"
     Cert-Name-Attr                  "<attr_desc>"
     Tunneled-EAP-MTU-Reduction      <value>
     Max-Framed-MTU                  <value>
     SSL-Debug-Level                 <value>
     Cipher-List                     <string>
     TLS-Version                     <string>
     Use-Server-Cipher-Preferences   <yes/no>
}
```

Variable	Description
RSA-Certificate-Path (string)	Full path to the RAD-Series Server RSA certificate in `.pem` or `.cer` format. The CA certificate for this server certificate MUST be present in the CA list file. To disable the RSA cipher suites, configure this variable as empty (""). The default value is set based on the installation location and is by default set to: */etc/opt/aaa/security/rsa_cert.pem*
RSA-Key-Path (string)	Full path to the file in `.pem` or `.cer` format that contains the private key used to generate the RAD-Series Server RSA certificate. This file cannot have a pass phrase or be encrypted. To disable the RSA cipher suites, configure this variable as empty (""). The default value is set based on the installation location and is by default set to: */etc/opt/aaa/security/rsa_key.pem*
DSA-Certificate-Path (string)	Full path to the RAD-Series Server DSA certificate in `.pem` or `.cer` format. The CA certificate for this server certificate MUST be present in the CA list file. To disable the ECC cipher suites, configure this variable as empty ("") or do not configure it at all. The default value is set based on the installation location and is by default set to: */etc/opt/aaa/security/ecdsa_cert.pem*
DSA-Key-Path (string)	Full path to the file in `.pem` or `.cer` format that contains the private key used to generate the RAD-Series Server DSA certificate. This file cannot have a pass phrase or be encrypted. To disable the ECC cipher suites, configure this variable as empty ("") or do not configure it at all. The default value is set based on the installation location and is by default set to: */etc/opt/aaa/security/ecdsa_key.pem*
CA-Path (string)	Full path to the Certificate Authority certificate for the client and server certificates. Used by the RAD-Series Server to authenticate client certificates. The CA certificate must be in .pem format. The default value is set based on the installation location and is by default set to: */etc/opt/aaa/security/ca_list.pem*
Random-Path (string)	Full path to the random seed used to generate encryption keys. The default value is set based on the installation location and is by default set to: */etc/opt/aaa/security/random.rnd*
CRL-Path (string)	Full path to the Certificate Revocation List. This is used by EAP-TLS to identify certificates that are no longer valid and therefore the access will be denied. The default value is set based on the installation location and is by default set to: */etc/opt/aaa/security/random.rnd*

Variable	Description
ECDHE-Parameter-Curve (string)	The name of the elliptic curve to use for generating the ECDH ephemeral key. To enable the ECDHE cipher suites, the ECHDE parameter curve must be configured. To disable the ECC cipher suites, configure this variable as empty ("") or do not configure it at all. A few of the many possible names are: • secp256r1 • secp384r1 • secp521r1 The default value is "secp256r1".
Cert-Name-Attr (string)	For TLS this identifies the attribute in the client certificate that the RAD-Series Server should verify is the same as the `User-Name` in the authentication request. If it does not match, then the access will be denied. The default value is "Subject:CommonName". The list of valid certificate attributes is: • "Subject:CommonName" • "SubjectAltName:RFC822Name" • "Subject:EmailAddress" • "CheckAllNameFields"
Tunneled-EAP-MTU- Reduction (integer)	The number of bytes by which to reduce the `Framed-MTU` AVP value when a EAP-PEAP/EAP-TTLS inner (tunnel) request is created. EAP- PEAP and EAP-TTLS must have enough room in the outer packet to contain the inner (tunneled) EAP conversation plus any attributes (such as Reply-Message) that must be sent outside the tunnel during the exchange. This parameter specifies how much of the outer `Framed-MTU` value is reserved for these non-tunneled attributes when constructing an inner reply. A value of at least 100 appears to be required for EAP-PEAP/EAP-TLS. The minimum value is 0 and the maximum value is 512. The default value is 0
Max-Framed-MTU (integer)	This parameter represents the customer's desired MTU size, and affects two environments: • The maximum size of the RADIUS response packet in the IP network, and • The maximum size of the EAP-Request fragment sent over the Supplicant/Access Point interface. The minimum value is 512 and the maximum value is 4094. The default value is 1480
SSL-Debug-Level (integer)	The minimum RAD-Series Server debug level required so that SSL debug output is produced in the radius.debug file. A value of 0 disables SSL debug output. The minimum value is 0 and the maximum value is 4. The default value is 0

Variable	Description
TLS-Version (string)	This parameter specifies the TLS versions you whish to allow. There are five known strings, representing the five supportable TLS versions, "SSLv2", "SSLv3", "TLSv1" (=SSL3.1), "TLSv1.1" (=SSL3.2), "TLSv1.2" (=SSL3.3). SSLv2 and SSLv3 are deprecated and we recommend they not be used. Configure the supported versions by splicing together the strings of the supported versions, with spaces or '+' as delimiters e.g. "TLSv1.1+TLSv1.2". The server will negotiate up to the highest version offered by the supplicant in the ClientHello The version strings are order-independent and case-insensitive, so "TLSv1+TLSv1.1+TLSv1.2" is the same as "tlsV1.2+TLSV1+tlsv1.1". The default is "TLSv1+TLSv1.1+TLSv1.2".
Cipher-List (string)	This is an optional list of cipher suites, overriding the internal default list. This is either a named list known to OpenSSL, or an explicit list of ciphers, in order of preference. The default value for this parameter is currently the OpenSSL 1.0.2r named list "DEFAULT".
Use-Server-Cipher-Preferences (boolean)	This parameter can set to 'yes' or 'no' and controls whether to use the server's preferences instead of the client's preferences when choosing a cipher. When set to no, the server will always follow the clients preferences. The default value is yes.

LDAP Connection Properties

LDAP connections are supported in two ways, which are differently implemented but functionally equivalent.

This version of the server, and past versions, support LDAP connections via the ProLDAP plugin, which runs as a plugin in the main process.

LDAP connections, as of version 8.4, are also supported via a new ProLDAPX plugin. ProLDAPX is an extended, high performance adaptation of ProLDAP with the same basic functionality in both. If there are no performance issues then ProLDAP is recommended. If there is a large scale application which requires higher performance in a multi-CPU environment then please contact technical support for assistance in adapting your ProLDAP configuration to ProLDAPX.

The ProLDAPX plugin runs in the main process, in conjunction with a ProLDAPX subprocess. The work of the ProLDAP plugin is split into two components: the ProLDAPX plugin which reads configurations and interfaces to the server's finite state machine, and the ProLDAPX subprocess which manages the LDAP connections and retrieval from the LDAP server. The ProLDAPX plugin and ProLDAPX subprocess communicate over an internal shared memory interface.

ProLDAP Connection Properties

The ProLDAP support uses the block format to define variables.

The block format looks like this:

```
aatv.ProLDAP
{
      LDAP-Version                 <integer>
      Enable-Default-Conf          <boolean>
      Debug                        <integer>
      Retry-Interval               <integer>
      Retry-Wait                   <integer>
      TCP-Timeout                  <integer>
      Timeout                      <integer>
      TCP-Keepalive                <boolean>
      TCP-Keepalive-Idle           <integer>  # Supported on Linux only
      TCP-Keepalive-Interval       <integer>  # Supported on Linux only
      TCP-Keepalive-MaxCount       <integer>  # Supported on Linux only
      TLS-CACertDir                <string>
      TLS-CACertFile               <string>
      TLS-CertFile                 <string>
      TLS-KeyFile                  <string>
      Directory-Select-Mode        <string>
      Secret1                      <string>
      Secret2                      <string>
}
```

Variable	Description
LDAP-Version (integer)	The version of the LDAP protocol to employ. The valid values are 2 (for LDAPv2) and 3 (for LDAPv3). LDAPv2 is in historical status (see RFC3494) and is not recommended. The default value is 3.
Enable-Default-Conf (boolean)	Enables the OpenLDAP software to read its default configuration file, /etc/openldap/openldap.conf. This allows you to configure more of the OpenLDAP options. The default value is off. **Note**: Enable-Default-Conf configuration must precede any TLS-xxx configuration parameters.
Retry-Interval (integer)	Sets the number of seconds for the RAD-Series Server to wait before trying to reconnect to a LDAP directory server, when a realm has failover directory servers configured. The default value is 60 seconds.
Retry-Wait (integer)	Sets the number of seconds that the RAD-Series Server will wait before attempting to connect to the same failover LDAP server. When all failover directory servers configured for a realm are down, the RAD-Series Server will try to reconnect to one of the servers every time an access request is received. In that situation, this parameter guarantees that the software does not spend too much time in trying to reconnect those directory servers. The default value is 1 second.
Timeout (integer)	Sets the number of seconds that an LDAP connection will remain open when the RAD-Series Server has not been able to successfully perform any successful LDAP operation. This parameter allows better handling of the situation where the LDAP directory times out client connections. The default value is 60 seconds.
TCP-Keepalive (boolean)	When this option is set to Yes, the RAD-Series Server will send TCP Keepalive messages when the LDAP connection is idle for the period of time as specified by the TCP Keepalive Idle parameter. When this option is set to No, the RAD-Series Server will not send TCP Keepalive messages. The default is No.
TCP-Keepalive-Idle (integer)	The period of idleness (in seconds) before the first TCP Keepalive message is sent. The default value is zero, which means the system default is used. The minimum positive value is 60, requiring at least one minute of idle before sending the 1st keepalive probe. This parameter is ignored on non-Linux systems.
TCP-Keepalive-Interval (integer)	The interval (in seconds) between successive Keepalive probes until a response is received. The default value is zero, which means the system default is used. The minimum positive value is 5, requiring at least 5 seconds of idle between successive keepalive probes. This parameter is ignored on non-Linux systems.

Variable	**Description**
TCP-Keepalive-MaxCount (integer)	The number of consecutive unanswered Keepalive probes which are sent before the connection is dropped. The default value is zero, which means the system default is used. This parameter is ignored on non-Linux systems.
TCP-Timeout (integer)	Sets the number of 1/10 seconds that the RAD-Series Server will wait for an LDAP server when trying to establish the TCP connection. The default value is 30 (3 seconds).
Debug (integer)	Enables or disables OpenLDAP debugging. Output is written to the radius.debug file. The default is 0, disabled. A value of -1 maximizes LDAP debugging. See the OpenLDAP documentation for additional debug levels. **Note**: The OpenLDAP debugging only occurs if the RAD-Series Server debug has also been enabled.
TLS-CACertDir (string)	Used if SSL is enabled for the connection to the LDAP servers. This is the path to a directory containing individual Certificate Authority files. This is used if there is no single TLS CA certificate file specified. There is no default value. **Note**: Use of SSL requires either TLS-CACertDir or TLS-CACertFile to be defined.
TLS-CACertFile (string)	Used if SSL is enabled for the connection to the LDAP servers. The full path of a single file containing certificates for all the Certificate Authorities that clients will recognize. If defined, this variable will be used instead of the TLS CA certificate directory. There is no default value. **Note**: Use of SSL requires either TLS-CACertDir or TLS-CACertFile to be defined.
TLS-CertFile (string)	The TLS RAD-Series Server certificate file specifies the file that contains the certificate that the server will use. This parameter is necessary if any of the LDAP directories are configured to use SSL and require validation of the LDAP client certificate. There is no default value.
TLS-KeyFile (string)	The TLS RAD-Series Server private key file specifies the file that contains the private key that the server will use. This parameter is necessary if any of the LDAP directories are configured to use SSL and require validation of the LDAP client certificate. There is no default value.

Variable	Description
Directory-Select-Mode (string)	The RAD-Series Server can be configured to execute its LDAP searches in either a round-robin or a first-avilable manner. These Directory-Select-Mode values allow you to control that. • Round-Robin - sets the searches to be done in round robin or load balanced manner. • First-Available - sets the searches to be done in a first available manner. This can also be described as a fail-over mode since all requests are sent to the same server until it fails and then the next one in the list is tried. The default is Round-Robin. This option can be overridden on a realm by realm basis in the authfiles.
Secret1 (string)	The RAD-Series Server can be configured to encrypt the administrator’s password used in LDAP searches. For this to happen the two secrets used to encrypt the password must be configured. The strings can be from 1 to 255 alphnumeric characters. The default is no secret1 value.
Secret2 (string)	The RAD-Series Server can be configured to encrypt the administrator’s password used in LDAP searches. For this to happen the two secrets used to encrypt the password must be configured. The strings can be from 1 to 255 alphnumeric characters. The default is no secret2 value.

ProLDAPX Connection Properties

ProLDAPX is an extended, high performance adaptation of ProLDAP with the same basic functionality in both. If there are no performance issues then ProLDAP is recommended. If there is a large scale application which requires higher performance in a multi-CPU environment then please contact technical support for assistance in adapting your ProLDAP configuration to ProLDAPX.

The ProLDAPX plugin support uses the block format to define variables.The ProLDAPX block identically supports all of the parameters found in the aatv.ProLDAP{} block, plus some additional parameters.

The block format looks like this:

```
ProLDAPX
{

    # The following parameters are identical in function
    # to those in aatv.ProLDAP{} block,
    # but have a "ProLDAPX-" prefix in the ProLDAPX{} block
    ProLDAPX-LDAP-Version            <integer>
    ProLDAPX-Enable-Default-Conf     <boolean>
    ProLDAPX-Debug                   <integer>
    ProLDAPX-Retry-Interval          <integer>
    ProLDAPX-Retry-Wait              <integer>
    ProLDAPX-TCP-Timeout             <integer>
    ProLDAPX-Timeout                 <integer>
    ProLDAPX-TCP-Keepalive           <boolean>
    ProLDAPX-TCP-Keepalive-Idle      <integer> # Supported Linux only
    ProLDAPX-TCP-Keepalive-Interval  <integer> # Supported Linux only
    ProLDAPX-TCP-Keepalive-MaxCount  <integer> # Supported Linux only
    ProLDAPX-TLS-CACertDir           <string>
    ProLDAPX-TLS-CACertFile          <string>
    ProLDAPX-TLS-CertFile            <string>
    ProLDAPX-TLS-KeyFile             <string>
    ProLDAPX-Directory-Select-Mode   <string>
    ProLDAPX-Secret1                 <string>
    ProLDAPX-Secret2                 <string>

    # The remaining parameters are specific to ProLDAPX

    ProLDAPX-Subprocess-Logfile      <boolean>
    ProLDAPX-Subprocess-Debug-Level  <integer>
    ProLDAPX-Statistics-Interval     <integer>
    ProLDAPX-Maximum-Active-Requests <integer>

}
```

Variable	Description
ProLDAPX-LDAP-Version (integer)	The version of the LDAP protocol to employ. The valid values are 2 (for LDAPv2) and 3 (for LDAPv3). LDAPv2 is in historical status (see RFC3494) and is not recommended. The default value is 3.
ProLDAPX-Enable-Default-Conf (boolean)	Enables the OpenLDAP software to read its default configuration file, /etc/openldap/openldap.conf. This allows you to configure more of the OpenLDAP options. The default value is off. **Note**: Enable-Default-Conf configuration must precede any TLS-xxx configuration parameters.
ProLDAPX-Debug (integer)	Enables or disables OpenLDAP debugging. Output is written to the radius.debug file. The default is 0, disabled. A value of -1 maximizes LDAP debugging. See the OpenLDAP documentation for additional debug levels. **Note**: The OpenLDAP debugging only occurs if the RAD-Series Server debug has also been enabled.
ProLDAPX-Retry-Interval (integer)	Sets the number of seconds for the RAD-Series Server to wait before trying to reconnect to a LDAP directory server, when a realm has failover directory servers configured. The default value is 60 seconds.
ProLDAPX-Retry-Wait (integer)	Sets the number of seconds that the RAD-Series Server will wait before attempting to connect to the same failover LDAP server. When all failover directory servers configured for a realm are down, the RAD-Series Server will try to reconnect to one of the servers every time an access request is received. In that situation, this parameter guarantees that the software does not spend too much time in trying to reconnect those directory servers. The default value is 1 second.
ProLDAPX-Timeout (integer)	Sets the number of seconds that an LDAP connection will remain open when the RAD-Series Server has not been able to successfully perform any successful LDAP operation. This parameter allows better handling of the situation where the LDAP directory times out client connections. The default value is 60 seconds.

Variable	Description
ProLDAPX-TCP-Keepalive (boolean)	When this option is set to Yes, the RAD-Series Server will send TCP Keepalive messages when the LDAP connection is idle for the period of time as specified by the TCP Keepalive Idle parameter. When this option is set to No, the RAD-Series Server will not send TCP Keepalive messages. The default is `NO`.
ProLDAPX-TCP-Keepalive-Idle (integer)	The period of idleness (in seconds) before the first TCP Keepalive message is sent. The default value is zero, which means the system default is used. The minimum positive value is 60, requiring at least one minute of idle before sending the 1st keepalive probe. This parameter is ignored on non-Linux systems.
ProLDAPX-TCP-Keepalive-Interval (integer)	The interval (in seconds) between successive Keepalive probes until a response is received. The default value is zero, which means the system default is used. The minimum positive value is 5, requiring at least 5 seconds of idle between successive keepalive probes. This parameter is ignored on non-Linux systems.
ProLDAPX-TCP-Keepalive-MaxCount (integer)	The number of consecutive unanswered Keepalive probes which are sent before the connection is dropped. The default value is zero, which means the system default is used. This parameter is ignored on non- Linux systems.
ProLDAPX-TCP-Timeout (integer)	Sets the number of 1/10 seconds that the RAD-Series Server will wait for an LDAP server when trying to establish the TCP connection. The default value is 30 (3 seconds).
ProLDAPX-TLS-CACertDir (string)	Used if SSL is enabled for the connection to the LDAP servers. This is the path to a directory containing individual Certificate Authority files. This is used if there is no single TLS CA certificate file specified. There is no default value. **Note**: Use of SSL requires either TLS-CACertDir or TLS-CACertFile to be defined.
ProLDAPX-TLS-CACertFile (string)	Used if SSL is enabled for the connection to the LDAP servers. The full path of a single file containing certificates for all the Certificate Authorities that clients will recognize. If defined, this variable will be used instead of the TLS CA certificate directory. There is no default value. **Note**: Use of SSL requires either TLS-CACertDir or TLS-CACertFile to be defined.

Variable	Description
ProLDAPX-TLS-CertFile (string)	The TLS RAD-Series Server certificate file specifies the file that contains the certificate that the server will use. This parameter is necessary if any of the LDAP directories are configured to use SSL and require validation of the LDAP client certificate. There is no default value
ProLDAPX-TLS-KeyFile (string)	The TLS RAD-Series Server private key file specifies the file that contains the private key that the server will use. This parameter is necessary if any of the LDAP directories are configured to use SSL and require validation of the LDAP client certificate. There is no default value
ProLDAPX-Directory-Select-Mode (string)	The RAD-Series Server can be configured to execute its LDAP searches in either a round-robin or a first-avilable manner. These ProLDAPX-Directory-Select-Mode values allow you to control that. • Round-Robin - sets the searches to be done in round robin or load balanced manner. • First-Available - sets the searches to be done in a first available manner. This can also be described as a fail-over mode since all requests are sent to the same server until it fails and then the next one in the list is tried. The default is Round-Robin. This option can be overridden on a realm by realm basis in the authfiles.
ProLDAPX-Secret1 (string)	The RAD-Series Server can be configured to encrypt the administrator’s password used in LDAP searches. For this to happen the two secrets used to encrypt the password must be configured. The strings can be from 1 to 255 alphnumeric characters. The default is no secret1 value.
ProLDAPX-Secret2 (string)	The RAD-Series Server can be configured to encrypt the administrator’s password used in LDAP searches. For this to happen the two secrets used to encrypt the password must be configured. The strings can be from 1 to 255 alphnumeric characters. The default is no secret2 value.
ProLDAPX-Subprocess-Logfile (boolean)	If `ON`, the `ProLDAPX` subprocess will create and maintain its own logfile, named `proldapxsubprocess.logfile`, akin to the server’s `logfile`. If `OFF`, no subprocess logfile is created. The default value is `OFF`

Variable	Description
ProLDAPX-Subprocess-Debug-Level (non-negative integer)	If in the range 1-4, the `ProLDAPX` subprocess will create and maintain its own debug file, named `proldapxsubprocess.debug`, akin to the main process's `radius.debug` file. Each increase in value produces more detailed debug information, as "-x" in the radiusd process does. If zero, the `ProLDAPX` subprocess will not create a debug file. The default value is 0 (zero).
ProLDAPX-Statistics-Interval (non-negative integer)	If non-zero, this value represents a periodic interval, in seconds, when the `ProLDAPX` plugin and subprocess will generate usage statistics in the logfile. The default is 0 (zero), which means interval statistics will not be generated. HUPing the subprocess will produce statistics on demand.
ProLDAPX-Maximum-Active-Requests (positive integer)	This number represents the maximum number of LDAP lookups that can be in progress simultaneously by the subprocess. The default value is 10000, and the valid range is 100 through 40000.

Hunt Group RADIUS proxies Properties

The Hunt Group RADIUS proxies server wide defaults are configured as a HGRADIUS{} configuration block and can be overriden by realm entries in the authfile. Along with the HGRADIUS{}, you configure a group of home servers to proxy a request to via a new “`HGRADIUS`” authentication type (in the authfile), rather than configuring a single home server to proxy to via the existing “`RADIUS`” authentication type. Here are the parameters for HGRADIUS{}:

```
HGRADIUS
{
  Acct-Timeout              <integer>
  Auth-Timeout              <integer>
  Destination-Select-Mode   <string>
  Failure-Limit             <integer>
  Group-Downtime            <integer>
  Request-Tries             <integer>
  Server-Downtime           <integer>
  Statistics-Interval       <integer>
}
```

Note: The parameter names (“`Acct-Timeout`”, “`Auth-Timeout`”, etc) are case insensitive. The values for `Destination-Select-Mode` are also case insensitive.
The name of the block “HGRADIUS” is, as are all curly brace block names, case sensitive.

Parameter	**Description**
acct-timeout (integer)	Time (in seconds) that HGRADIUS waits for an accounting reponse before trying next server. The default value is 5.
auth-timeout (integer)	Time (in seconds) that HGRADIUS waits for an authentication reponse before tring next server. The default value is 5.
destination-select-mode (string)	Sets the Hunt Group type to Round-Robin or First-Available. The default value is Round-Robin.
failure-limit (integer)	Number of consecutive RADIUS request failures to declare a server down. The default value is 3.

Parameter	Description
group-downtime (integer)	Time (in seconds) that a HGRADIUS Group is down before resetting its status to up and thus start sending requests again to that group. The default value is 300.
request-tries (integer)	Maximum number of servers in a HGRADIUS Group to try before giving up on sending a particular request. The default value is 2.
server-downtime (integer)	Time (in seconds) that a HGRADIUS Server is down before resetting its status to up and thus start sending requests again to that server. The default value is 900.
statistics-interval (integer)	Time (in seconds) to collect HGRADIUS Statistics. The default value is 0, i.e. no statistics are collected.

Accounting Distribution Properties

The Accounting Distribution Properties define server wide default values for forwarding accounting requests. The forwarding is triggered by the presence of one or more “`AcctDist-Recipient`” AVPs in the accounting authreq. `AcctDist-Recipient` is an Interlink internal attribute, whose value holds the name or IP address of a RADIUS peer (must also be in the clients file) to whom we want to forward a copy of the accounting request.

When an accounting message is copied and forwarded, the server handles retransmitting the message if a timely Accounting-Response is not received. There is a timer for controlling when the request needs to be retransmitted, and a count for the maximum number of times a request is retransmitted before the server gives up on it. A NAS retransmission has no effect on the retransmission of these distributed accounting messages.

Note the `AcctDist-Recipient, AcctDist-Timeout` and `AcctDist-Retry-Limit` attributes are INTERNAL. They cannot be sent in the NAS’s Accounting-Request. They are not forwarded in the proxied accounting request. Something in the server has to create these attributes and add them to the accounting authreq. One way to do this is to modify the request-ingress policy to selectively added these AcctDist-xxx attributes to certain received accounting requests based on realm name.

The Accounting Distribution Properties are configured as a ACCTDIST{} configuration block with these parameters:

```
ACCTDIST
{
  Request-Timeout            <integer>
  Retry-Limit                <integer>
  Extra-Logging              <string>
  Statistics-Interval        <integer>
}
```

Note: The parameter names (“`Request-Timeout`”, “`Retry-Limit`”, etc) are case insensitive. The values for `Extra-Logging` are also case insensitive.
The name of the block “ACCTDIST” is, as are all curly brace block names, case sensitive.

Parameter	Description
request-timeout (integer)	Time (in seconds) that ACCTDIST waits for an accounting reponse before retransmitting the accounting request. The valid values are 1 to 20. The default value is 3.
retry-limit (integer)	Maximum number of retransmissions to try before giving up. The valid values are 0 to 10. The default value is 1.
extra-logging (string)	Enables additional messages to logfile without needing debug on. The default value is off.
statistics-interval (integer)	Time (in seconds) to collect ACCTDIST interval statistics before outptting them to the logfile. The default value is 0.

RADIUS Listen Socket Properties

The RADIUS listen sockets are configured as a set of one or more radius_socket{} configuration blocks, with these parameters:

```
radius_socket
{
  ipaddr           <ipaddress>            # Required,IPv4 or IPv6 address
  acctport         <port#>                # One or both of the ports
  authport         <port#>                #  must be specified
  acct_udp_recv_buffer_size <bufsize> # Optional, else use OS default
  auth_udp_recv_buffer_size <bufsize> # Optional, else use OS default
}
```

Note: The parameter names ("`ipaddr`", "`authport`", etc) are case insensitive. The values for `ipaddr` are also case insensitive.
The name of the block "radius_socket" is, as are all curly brace block names, case sensitive.

Parameter	**Description**
ipaddr (IP address)	The IP address can be an IPv4 specific address, an IPv6 specific address, the IPv4 ANY address (0.0.0.0), or the IPv6 ANY address (::). The IP address may optionally be enclosed in square brackets. This parameter is required. If the IPv4 ANY address is specified, the RAD-Series Server will listen on all IPv4 interfaces. If the IPv6 ANY address is specified, the server will listen for IPv4 and IPv6 messages on all interfaces. There is no default value.
acctport (integer)	Sets the UDP port number to receive accounting requests on. The minimum value is 0 and the maximum value is 65535. In order to use the radius_socket block, you must specify at least one of the two ports. There is no default value. If acctport is configured with the special value of zero, the RAD-Series Server will execute a hierarchy of steps to determine the accounting listen port: • If "-q acctport" is specified on the startup command, use that value, else • If the environment variable `RAD_ACCT_PORT` is defined, use that, else • If a RADIUS accounting port is configured in the `/etc/services` file, use that, else • Use 1813, as defined by the RADIUS RFC. If acctport is not configured, the server will not open an accounting listen socket for the given <ipaddress>.

Parameter	Description
authport (integer)	Sets the UDP port number to receive authentication requests on. The minimum value is 0 and the maximum value is 65535. In order to use the radius_socket block, you must specify at least one of the two ports. There is no default value. If authport is configured with the special value of zero, the RAD-Series Server will execute a hierarchy of steps to determine the accounting listen port: • If "-p authport" is specified on the startup command, use that value, else • If the environment variable `RAD_AUTH_PORT` is defined, use that, else • If a RADIUS auth port is configured in the `/etc/services` file, use that, else • Use 1812, as defined by the RADIUS RFC. If authport is not configured, the server will not open an authentication listen socket for the given <ipaddress>.
acct_udp_recv_buffer_size (integer)	The requested size in bytes that the operating system will use to buffer inbound RADIUS UDP packets received on the accounting port. This parameter allows the increasing of the amount of buffering that may be needed to handle a peak load. Increasing the buffering too much could have undesired affects. **Note**: The operating system may not honor the requested buffer size. A logfile message will display the actual buffer size allowed by the operating system. The minimum value is 8192 and the maximum value is 8388608. There is no default value. If this parameter is not specified or is specified with a value of zero, the value used is the operating system's UDP default buffer size.
auth_udp_recv_buffer_size (integer)	The requested size in bytes that the operating system will use to buffer inbound RADIUS UDP packets received on the authentication port. This parameter allows the increasing of the amount of buffering that may be needed to handle a peak load. Increasing the buffering too much could have undesired affects. **Note**: The operating system may not honor the requested buffer size. A logfile message will display the actual buffer size allowed by the operating system. The minimum value is 8192 and the maximum value is 8388608. There is no default value. If this parameter is not specified or is specified with a value of zero, the value used is the operating system's UDP default buffer size.

DHCP Server Properties

The DHCPv4 support uses the block format to define variables. The block format looks like this:

```
aatv.DHCP
{
      dhcp-enabled                                  value
      dhcp-server-name                              value
      dhcp-server-ip-address                        value
      dhcp-server-port                              value
      dhcp-relay-port                               value
      dhcp-client-hardware-type                     value
      initial-retransmission-interval               value
      max-retransmission-interval                   value
      max-number-of-dhcpdiscover-retransmissions    value
      max-number-of-dhcprequest-retransmissions     value
      max-dhcp-msg-len                              value
      send-maximum-dhcp-message-size-option         value
      send-user-class-option                        value
}
```

Variable	Description
dhcp-enabled (string)	If Yes, it enables the DHCP Relay function. The default is No.
dhcp-server-name (string)	Specifies the fully qualified domain name of the DHCPv4 server to use. Configure this variable or the dhcp-server-ip-address, at least one is required. There is no default value.
dhcp-server-ip-address (IP address)	Specifies the IPv4 address of the DHCPv4 server to use. Use this variable or the dhcp-server-name but one is required. This variable takes precedence over `dhcp-server-name`. There is no default value.
dhcp-server-port (integer)	The UDP port on the DHCP server to which DHCP requests are sent. The default value is 67.
dhcp-relay-port (integer)	The UDP port on the RAD-Series Server at which DHCP responses are received. The default value is 67.
dhcp-client-hardware-type (integer)	Value passed to the DHCP server to indicate hardware type. Options are: 0 (NONE) or 1 (ETHERNET). The default value is 1.
initial-retransmission-interval (integer)	Interval, in seconds, before the initial retransmission of a request to the DHCP server. The RAD-Series Server will double the retransmission interval for each subsequent retransmission. The default value is 4 seconds.

Variable	Description
max-retransmission-interval (integer)	The maximum interval in seconds at which the RAD-Series Server retransmits DHCP requests. The default value is 60 seconds.
max-number-of-dhcpdiscover-retransmissions (integer)	The maximum number of retransmissions RAD-Series Server makes when acknowledging IP address assignments from the DHCP Server. The default value is 2.
max-number-of-dhcprequest-retransmissions (integer)	The maximum number of retransmissions RAD-Series Server makes when requesting IP address assignments from the DHCP Server. The default value is 2.
max-dhcp-msg-len (integer)	The maximum size in bytes of messages that can be received from the DHCP server. The default value is 1500.
send-maximum-dhcp-message-size-option (boolean)	If Yes, send the Maximum-DHCP-Message-Size option to the DHCP server (required by some DHCP servers) in the Discovery message. If No, do not send this option. The default value in No.
send-user-class-option (boolean)	Specifies which attribute in the DHCP message will carry the IPv4 address pool name. If Yes, the pool name will be sent in the User-Class option. If No, the pool name will be sent in the Vendor-Class-Identifier option. The default value is No.

RSA SecurID Server Properties

The RSA SecurID authentication supports connections to RSA SecurID Authentication Manager versions 6.1.2 and later, 7.1 SP2, 7.1 SP3 and 8.1 SP2 and later. The RSA SecurID support uses the block format to define parameters. The block format looks like this:

```
SecurID
{
      Debug-Level                                  value
      Log-Statistics-Interval                      value
      Number-of-Authentication-Control-Blocks      value
      RSA-Trace-Level                              value
}
```

Variable	Description
Debug-Level (integer)	This parameter configures the debug level for RSA SecurID authentications. If the Debug-Level is > 0, then the SecurID subprocess will create and and maintain its own logfile, named `securidsubprocess.logfile`, akin to the server's `logfile`, and the SecurID subprocess will create and maintain its own debug file, `named securidsubprocess.debug`, akin to the server's `radius.debug` file. The subprocess statistics will be logged in the `securidsubprocess.logfile.` If the Debug-Level is 0, then neither a `securidsubprocess.logfile` nor a `securidsubprocess.debug` will be created. The subprocess statistics will be logged in the server's `logfile`. The valid range is 0 to 4. The default is 0, which means no debugging.
Log-Statistics-Interval (integer)	This parameter configures the interval, in seconds, at which RSA SecurID statistics are logged. If the Log-Statistics-Interval is N and N > 0 then the RSA SecurID subprocess will output statistics every N seconds. The valid range is 0 to 2147483647. The default is 0, which means no interval statistics are generated. HUPing the subprocess will produce statistics on demand independent of the Log- Statistics-Interval setting.

Variable	Description
Number-of-Authentication-Control-Blocks (integer)	This specifies the number of Authentication-Control-Blocks to allocate for RSA SecurID authentications. An Authentication-Control-Block tracks a RSA SecurID authentication from beginning to end. This controls the maximum number of concurrent pending authentications with the RSA SecurID Authentication Manager. The valid range is 1 to 8192. The default value is 1024.
RSA-Trace-Level (integer)	Specifies the level of tracing done by the RSA SecurID client library code. The RSA SecurID subprocess sets the `RSATRACELEVEL` environment variable to this value, for use by the RSA client library code. If non-zero then the RSA trace output is written to the file rsatrace.log in the server logfile directory. The valid range is 0 to 15. The default is 0, which means no rsatrace.log is generated.

Tunneling Properties

The server resolves tunneling hints in an Access-Request message in this way:

- If all hints match configured attributes, the configured values are returned to the client.
- If some of the hints match configured attributes, both configured values and hints are returned to the client.
- If none of the hints match configured attributes, then all hints and all configured values are returned to the client.

The tunneling support uses the block format to define variables. The block format looks like this:

```
aatv.Tunneling
{
     variable value
     . . .
}
```

Variable	Description
NO_HINT (String)	This parameter specifies what the RAD-Series Server should do with tunneling attributes configured as reply items for a user entry. There can only be one NO_HINT variable specified. The following entries are valid: NO_HINT Return-Configured-Tunnel-Attributes NO_HINT Return-No-Tunnel-Attributes NO_HINT Reject-Access-Request The default will return configured tunnel attributes. You cannot control how the server will resolve configured attributes and hints. When tunneling hints exist in an Access-Request, attribute values will be returned to the client as follows: • Configured attributes will be returned if all hints match configured values. • If some or no hints match configured attributes, the hints and attributes will be consolidated. If there are conflicting attribute values, the configured value will be used. • If no configured attributes exist, no attributes will be returned.
HINTS	This parameter specifies the behavior when RAD-Series Server receives an Access-Request that contains Tunnel Hint attributes. The values are: "HINTS Accept": Accept and process the received tunnel attributes. "HINTS Discard": Discard the received tunnel attributes. Processing of the Access-Request proceeds as if the tunneling attributes were never present. If no value is configured, the default is Accept.

Variable	Description
Tagged-VSA-Hints	This parameter specifies the behavior when the RAD-Series Server receives an Access-Request that contains VSA Tunnel Hint attributes. The options are: Accept: Accept and process the received VSA tunnel attributes. Discard: Discard the received tunnel attributes. Processing of the Access-Request proceeds as if the tunneling VSAs were never present. Reject: Fail the authentication by sending an Access-Reject If no value is configured, the default is Accept.
Tunnel-Password-Requires-Message-Authenticator	This parameter indicates if a Message-Authenticator attribute is required in a RADIUS message containing a Tunnel-Password attribute. The default is No. **Note**: The default is to not enforce the RFC for backwards-compatibility with RAD-Series Server versions prior to 8.1 which didn’t enforce the RFC requirement.

authfile and EAP.authfile

The `authfile` can be used to configure per-realm actions, but the actions in the `authfile` should be user data retrieval instead of authentication. For example, if a realm retrieves its user data from an LDAP directory with an LDAP Search operation to do EAP authentication, that realm must be configured in the `authfile` with the ProLDAP or ProLDAPX AATV and then again in the `EAP.authfile` with the EAP AATV.

The `EAP.authfile` is used to configure per-realm authentication actions. The distributed copy of this file contains a DEFAULT entry such as:

```
DEFAULT iaaaAuthenticate "default"
```

Note: Do not remove this line!

That provides the conventional authentication logic for any realm not explicitly configured to authenticate their users in a particular way, for example, PAP and Kerberos. Realms that do need some particular method for user authentication must be configured here explicitly.

Note: RFC7542 (NAI) places many restrictions on the <realm>:

- The only allowable characters are the letters A-Z and a-z, numbers 0-9, dots '.', and dashes '-'. The only non-alpha numerics allowed are dots '.', and dashes '-'.
- Also the realm name must start and end with an alphanumeric (no starting or ending dot or dash allowed).
- Also at least one dot is required, but one cannot have two dots in a row.

However, RAD-Series Server places no such restrictions on the realm name. Anything after the "@" delimiter is taken to be an acceptable realm, no validation is performed.

Note: It is recommend that the (unenforced) RFC realm character set be used, for better inter operability and better RFC compliance.

Authfile Entry Syntax

realm-name -flags auth-type auth-parameter

or

realm-name -flags auth-type auth-parameter
{
    extended-auth-parameters
}

realm-name - portion of the NAI format login following the @ sign (e.g.: yourcompany.com). It is recommended that the ***realm-name*** contain only the letters A-Z and a-z, the digits 0-9, dots '.', and dashes '-'; that the ***realm-name*** contain at least one dot, and that the realm-name begin and end with an alphanumeric.

flags - the optional flags are used to specify additional information about this realm entry. See Protocol and Case-sensitivity Flags below.

auth-type - name of the `Authentication-Type`, as defined in the dictionary file.

auth-parameter - meaning varies by `Authentication-Type`.

extended-auth-parameters - a sequence of parameters and values which vary by `Authentication-Type`.

Some `Authentication-Types` have an extended authentication parameter block as shown in the second example above. The content varies by `Authentication-Type`.

Examples

```
bigearth.net                iaaaFile        flatearth

satellite.com               iaaaFile        flatearth

flatland.com    -DEFAULT    iaaaFile        flatland

flatland.com    -EAP        iaaaFile        flatland

flatland.com    -AKA        localFile       flatland-aka
{
     Request-Attribute-For-Search           real-username
}

flatland.com    -SIM        localFile       flatland-sim
{
     Request-Attribute-For-Search           real-username
}

ldap.com ProLDAP "This is a comment like Corporate LDAP"
{
     Request-Attribute-For-Search           Real-Username
     Filter-Type                            CIS

     Directory-Select-Mode                  First-Available
     Directory "LDAP directory comment"
     {
          URL              "ldap://ldap1.ispx.com:389"
          Administrator    "cn=admin,ou=devision,dc=com"
          Password         "password"
          SearchBase       "ou=devision,dc=com"
          Authenticate     Search
     }
}

# Proxy example:

othernet.com    RADIUS     rad1.othernet.com
```

Protocol Flags

If users in the same realm require different authentication methods, you may be able to specify which realm entry to use by putting a flag in the authfile mapping. The flags match one of five mutually exclusive protocol identifiers that may be present in the Access-Request, plus a sixth “default” flag for users who do not explicitly match one of the other attributes. See “Configuring “Configuring EAP-AKA” on page 272 for information on configuring EAP-AKA. See “Configuring EAP-SIM” on page 293 for information on configuring EAP-SIM.

Flag	Matches Access-Request Attribute
-PW	User-Password
-CHAP	CHAP-Password
-EAP	EAP-Message
-AKA	EAP-Message Data Store lookup for EAP-AKA
-SIM	EAP-Message Data Store lookup for EAP-SIM
-DEFAULT	None. Authentication type to use for all users who do not match one of the other flags.

Case-sensitivity Flags

The RAD-Series Server can treat user names in a case-sensitive or a case-insensitive manner. These mutually-exclusive flags allow you to control that. The default is `-BIN` (case-sensitive). These flags are ignored for an `Authentication-Type` that uses an extended authentication parameter block and has filter-type capabilities.

Flag	Behavior
-BIN	Username is not modified before authentication.
-CIS	Username is converted to uppercase before authentication. Therefore the data store needs to store the username in uppercase unless the datastore is case-insensitive.

Authentication Types

Local Storage in Default users File

Local storage of user's credentials can be stored in a flat file on server wide basis. `users` is the file that is used to store these credentials. There is no entry required in the `authfile` to invoke this lookup. All authentications attempt to find the user's credentials in this file before using any other authentication type.

Note: The user's entry in the "`users`" file must include the full users' identity, including any realm name.

Local Storage – iaaaFile

Local storage of user's credentials can be stored in a flat file on a realm by realm basis. In the `authfile`, the `iaaaFile` authenication type is used to specify the realm and the flat file name in the *`auth-parameter`* field. For example:

```
fred.com    iaaaFile    fred.com
```

Specifies that realm fred.com users are to be looked up in the file, `fred.com.users`.

Note: The flat file name in the `authfile` does not include the required extension of "`.users`".

Unix Password

User's credentials can be stored in the Unix password file. In the `authfile`, the `PASSWD` authenication type is used to specify a realm whose users should be looked up in the Unix passwd file. The *`auth-parameter`* field is empty. For example:

```
fred.com    PASSWD     ""
```

Specifies that realm fred.com users are to be looked up in the Unix password file.

SecurID

User's credentials can be stored in RSA SecuID authentication server which does two factor authentication. In the `authfile`, the `SecurID` authenication type is used to specify a realm whose users should be authenticated using a SecurID ACE server. The *`auth-parameter`* field is empty. For example:

```
fred.com    SecurID    ""
```

Specifies that realm fred.com users are to be authenticated via the RSA SecurID ACE server. You will also need to export two files, `securid` and `sdconf.rec`, from the ACE server to allow secure communication between the RAD-Series AAA server and the SecurID ACE server. They need to be put in the configuration directory. There are other configuration parameters in the `aaa.config` file which affect the communications with the RSA SecurID ACE server.

ProLDAP Authentication Type

In the `authfile`, the `ProLDAP` authentication type requires extended parameters to configure the connection to the LDAP server. The extended parameters look like this (required subset is in bold):

```
<realm_name>  ProLDAP  "<realm_comment>"
{
     Policy-Pointer                    "decisionfile://<filespec>"
     Request-Attribute-For-Search      <attribute_name>
     Filter-Type                       <filter_value>
     Directory-Select-Mode             <mode_value>
     Directory "<LDAP_directory_name>"
     {
          URL                          "<type>://<server>[:<port>]"
          Administrator                "<admin>"
          Password                     "<password>"
          Encrypted-Password           "<encrypted-password>"
          SearchBase                   "<search_base>"
          Authenticate                 { Search | Bind | Auto }
          Filter                       <filter>
     }
     Directory "<LDAP_directory_name2>"
     {
          . . .
     }
}
```

Parameter	Description
Policy-Pointer	A policy to be applied to all of the realm's users. This field is optional.
Request-Attribute-For-Search	This indicates the search attribute to use for a user lookup. The attribute must be a string-type (string, tag-str, octets). When ProLDAP is used for EAP-SIM or EAP-AKA, the `Request-Attribute-For-Search` attribute must be configured with a value of `Real-Username`. The default value, if not present, is `User-Id`.

Parameter	Description
Filter-Type	The RAD-Series Server can treat user names in a case-sensitive or a case-insensitive manner. These `Filter-Type` values allow you to control that. • `BIN` - sets it to case-sensitive, username is used as is. • `CIS` - sets it to case-insensitive, username is converted to uppercase before the LDAP lookup is done. Therefore, LDAP needs to store the username in uppercase unless LDAP is case-insensitive. The default is `BIN` (case-sensitive). Username is not modified before authentication.
Directory-Select-Mode	When multiple directories are specified, the RAD-Series Server can be configured to execute its LDAP searches in either a round-robin or a first-avilable manner. These `Directory-Select-Mode` values allow you to control that. • `Round-Robin` - sets the searches to be done in round robin or load balanced manner. • `First-Available` - sets the searches to be done in.a first available manner. This can also be described as a fail-over mode since all requests are sent to the same server until it fails and then the next one in the list is tried. If this is not specified, then the value in the `aatv.ProLDAP{}` block in the `aaa.config` file is used. If neither is present, then the default is used. The default is `Round-Robin`
Directory	*`LDAP_directory_name`* is the name of the directory. This does not have to be the actual directory name, just a unique identifier. The `Directory` definition can be repeated up to four times to define more LDAP servers that have the same content and configuration. There is no default value, if not present.

Parameter	Description
URL	The URL (Universal Record Locator) defines the server, port and type of connection that should be used. The connection `type` can be ldap (un-encrypted) or ldaps (SSL encrypted). If you configure it as `ldaps` then the certificates need to match the IP address or fully qualified domain name for it to work. If using ldaps (SSL), also specify the RAD-Series Server's TLS CA certificate path and file in the ProLDAP Connection Properties" on page 185. The *`server`* can be a fully qualified domain name that maps to an IPv4 or IPv6 address. It also can be an IPv4 or IPv6 address. If it is an IP address then it should be enclosed in "[ ]". If it is an IPv6 address and there is a *`port`*, then it **must** be enclosed in "[ ]". The aaa.config ipv6_enabled parameter controls whether IPv6 is enabled or disabled for RADIUS messages but it has no effect on LDAP communications. If the *`server`* is configured with an IPv6 address, or a domain name that maps to an IPv6 address, the LDAP communications will be IPv6. The TCP *`port`* is optional and defaults to 389 for a *`type`* of `ldap` and to 636 for a *`type`* of `ldaps`. There is no default value, if the URL is not present. Here are several sample URLs (the double-quotes are optional): URL"ldap://srvr.ldap.com" URL"ldap://ipv6.ldap.com:389" URL"ldap://[192.168.3.1]:389" URL"ldap://[2003:34::42]:389" URL"ldaps://ipv4.ldap.com" URL"ldaps://ipv4.ldap.com:636" URL"ldaps://ipv6.ldap.com:636" URL"ldaps://[192.168.3.1]:636" URL"ldaps://[2003:34::42]:636"
Administrator	*Distinguished Name (dn)* of the administrative user permitted to search the LDAP directory. This ID should match the ID you set up on your directory for the RAD-Series Server. This user must have read access to all the users to be authenticated by the RAD-Series Server and their passwords. If this field is omitted, the server will perform a bind to the directory using the user credentials from the Access-Request if it is necessary to validate the user's password.There is no default value, if not present.
Password	The clear text password is used for the `Administrator` bind to the LDAP directory server. `Password` is not required if there is no `Administrator`. If an `Encrypted-Password` is configured it takes precidence as the password to use.There is no default value, if not present.

Parameter	Description
Encrypted-Password	The encrypted password is used for `Administrator` bind to the LDAP directory server. `Encrypted-Password` is not required if there is no `Administrator`. The encrypted password can only be used if the `secrets1` and `secrets2` are defined in `aatv.ProLDAP{}` block in the `aaa.config` file. All encrypted passwords must be generated using `saltencrypt` utility located in the binary directory using the secrets mentioned above. The encrypted password take precedence over the clear text password. There is no default value, if not present.
SearchBase	Required. Pointer into the directory where the RAD-Series Server will begin to search for users in this realm. Enter a comma-delimited list of attribute-value pairs that represent the directory levels, no spaces. There is no default value, if not present.
Authenticate	Determines the mode to be used to access this LDAP server for this LDAP directory. There are three mode: Bind, Search and Auto. The default access mode is Auto. • Bind When binding as the user for authentication is desired. No Policy-Pointers, check items, or reply items will be returned to the RAD-Series Server when Bind is specified. • Search When a LDAP search as the configured administrator is desired. The RAD-Series Server expects the user's password in the search result. RAD-Series Server must perform an administrator search on the LDAP server when Policy-Pointers, check items, or reply items must be returned. • Auto When a LDAP search as the configured administrator (search anonymously if no configured administrator) is desired. After the search the RAD-Series Server expects the password to be returned in the search results. The RAD-Series Server binds as the user if the password is not available. Policy-Pointers, check items, and deny items will not be returned by the LDAP server if the RAD-Series Server reverts to bind. This mode can affect performance, since two LDAP operations may occur for one authentication. **IMPORTANT**: You must use Auto with a Microsoft Active Directory.
Filter	`Filter` specifies which LDAP attribute to use in the lookup. The three values are `User-Id, UID` and `sAMAccountName`. User-ID is case sensitive in LDAP and `UID` is case insensitive in LDAP. `sAMAccountName` MUST be used when accessing Active Directory via its LDAP interface. The default value is `UID`.

Example ProLDAP authfile configuration

```
# This realm uses an LDAP database
realm3.com                    ProLDAP        “LDAP lookup”
{
     Request-Attribute-For-Search  User-Id
     Filter-Type                   CIS
     Directory-Select-Mode         First-Available
     Directory           “Directory 1”
     {
          URL                 “ldap://ldap1.ispx.com:389”
          Administrator       “cn=...,ou=...,ou=...,o=radius”
          Password            “password”
          Authenticate        Search
          Filter              uid
     }
}
```

ProLDAPX Authentication Type

ProLDAPX is an extended, high performance adaptation of ProLDAP with the same basic functionality in both. If there are no performance issues then ProLDAP is recommended. If there is a large scale application which requires higher performance in a multi-CPU environment then please contact technical support for assistance in adapting your ProLDAP configuration to ProLDAPX.

In the `authfile`, the `ProLDAPX` authentication type requires extended parameters to configure the connection to the LDAP server. The extended parameters look like this (required subset is in bold):

```
<realm_name> ProLDAPX "<realm_comment>"
{
     Policy-Pointer                 "decisionfile://<filespec>"
     Request-Attribute-For-Search   <attribute_name>
     Filter-Type                    <filter_value>
     Directory-Select-Mode          <mode_value>
     Directory "<LDAP_directory_name>"
     {
          URL                    "<type>://<server>[:<port>]"
          Administrator          "<admin>"
          Password               "<password>"
          Encrypted-Password     "<encrypted-password>"
          SearchBase             "<search_base>"
          Authenticate           { Search | Bind | Auto }
          Filter                 <filter>
     }
     Directory "<LDAP_directory_name2>"
     {
          . . .
     }
}
```

Parameter	Description
Policy-Pointer	A policy to be applied to all of the realm's users. This field is optional.
Request-Attribute-For-Search	This indicates the search attribute to use for a user lookup. The attribute must be a string-type (string, tag-str, octets). When ProLDAPX is used for EAP-SIM or EAP-AKA, the `Request-Attribute-For-Search` attribute must be configured with a value of `Real-Username`. The default value, if not present, is `User-Id`.

Parameter	Description
Filter-Type	The RAD-Series Server can treat user names in a case-sensitive or a case-insensitive manner. These `Filter-Type` values allow you to control that. • `BIN` - sets it to case-sensitive, username is used as is. • `CIS` - sets it to case-insensitive, username is converted to uppercase before the LDAP lookup is done. Therefore, LDAP needs to store the username in uppercase unless LDAP is case-insensitive. The default is `BIN` (case-sensitive). Username is not modified before authentication.
Directory-Select-Mode	The RAD-Series Server can be configured to execute its LDAP searches in either a round-robin or a first-avilable manner. These `Directory-Select-Mode` values allow you to control that. • `Round-Robin` - sets the searches to be done in round robin or load balanced manner. • `First-Available` - sets the searches to be done in.a first available manner. This can also be described as a fail-over mode since all requests are sent to the same server until it fails and then the next one in the list is tried. If this is not specified, then the value in the `ProLDAPX{}` block in the `aaa.config` file is used. If neither is present, then the default is used. The default is `Round-Robin`
Directory	*`LDAP_directory_name`* is the name of the directory. This does not have to be the actual directory name, just a unique identifier. The `Directory` definition can be repeated to define more LDAP servers that have the same content and configuration. When there is more than one, the RAD-Series Server will load balance the request across all the defined LDAP servers. There is no default value, if not present.

Parameter	Description
URL	The URL (Universal Record Locator) defines the server, port and type of connection that should be used. The connection *`type`* can be `ldap` (un-encrypted) or `ldaps` (SSL encrypted). If you configure it as `ldaps` then the certificates need to match the IP address or fully qualified domain name for it to work. If using ldaps (SSL), also specify the RAD-Series Server's TLS CA certificate path and file in the "**ProLDAP** Connection Properties" on page 185. The *`server`* can be a fully qualified domain name that maps to an IPv4 or IPv6 address. It also can be an IPv4 or IPv6 address. If it is an IP address then it should be enclosed in "[ ]". If it is an IPv6 address and there is a *`port`*, then it **must** be enclosed in "[ ]". The aaa.config ipv6_enabled parameter controls whether IPv6 is enabled or disabled for RADIUS messages but it has no effect on LDAP communications. If the *`server`* is configured with an IPv6 address, or a domain name that maps to an IPv6 address, the LDAP communications will be IPv6. The TCP *`port`* is optional and defaults to 389 for a *`type`* of `ldap` and to 636 for a *`type`* of `ldaps`. There is no default value, if the URL is not present. Here are several sample URLs (the double-quotes are optional): URL"ldap://srvr.ldap.com" URL"ldap://ipv6.ldap.com:389" URL"ldap://[192.168.3.1]:389" URL"ldap://[2003:34::42]:389" URL"ldaps://ipv4.ldap.com" URL"ldaps://ipv4.ldap.com:636" URL"ldaps://[192.168.3.1]:636" URL"ldaps://[2003:34::42]:636"
Administrator	*Distinguished Name (dn)* of the administrative user permitted to search the LDAP directory. This ID should match the ID you set up on your directory for the RAD-Series Server. This user must have read access to all the users to be authenticated by the RAD-Series Server and their passwords. If this field is omitted, the server will perform a bind to the directory using the user credentials from the Access-Request if it is necessary to validate the user's password. There is no default value, if not present.
Password	The clear text password is used for the `Administrator` bind to the LDAP directory server. `Password` is not required if there is no `Administrator`. If an `Encrypted-Password` is configured it takes precidence as the password to use.There is no default value, if not present.

Parameter	**Description**
Encrypted-Password	The encrypted password is used for `Administrator` bind to the LDAP directory server. `Encrypted-Password` is not required if there is no `Administrator`. The encrypted password can only be used if the `secrets1` and `secrets2` are defined in `aatv.ProLDAPX{}` block in the `aaa.config` file. All encrypted passwords must be generated using `saltencrypt` utility located in the binary directory using the secrets mentioned above. The encrypted password take precedence over the clear text password. There is no default value, if not present.
SearchBase	Required. Pointer into the directory where the RAD-Series Server will begin to search for users in this realm. Enter a comma-delimited list of attribute-value pairs that represent the directory levels, no spaces. There is no default value, if not present.
Authenticate	Determines the mode to be used to access this LDAP server for this LDAP directory. There are three mode: Bind, Search and Auto. The default access mode is Auto. • **`Bind`** When binding as the user for authentication is desired. No Policy-Pointers, check items, or reply items will be returned to the RAD-Series Server when Bind is specified. • **`Search`** When a LDAP search as the configured administrator is desired. The RAD-Series Server expects the user's password in the search result. The RAD-Series Server must perform an administrator search on the LDAP server when Policy-Pointers, check items, or reply items must be returned. • **`Auto`** When a LDAP search as the configured administrator (search anonymously if no configured administrator) is desired. After the search the RAD-Series Server expects the password to be returned in the search results. The RAD-Series Server binds as the user if the password is not available. Policy-Pointers, check items, and deny items will not be returned by the LDAP server if the RAD-Series Server reverts to bind. This mode can affect performance, since two LDAP operations may occur for one authentication. IMPORTANT: You must use Auto with a Microsoft Active Directory.
Filter	`Filter` specifies which LDAP attribute to use in the lookup. The three values are `User-Id, UID` and `sAMAccountName`. `User-ID` is case sensitive in LDAP and `UID` is case insensitive in LDAP. `sAMAccountName` MUST be used when accessing Active Directory via its LDAP interface. The default value is `UID`.

Example ProLDAPX authfile configuration

```
# This realm uses an LDAP database
realm3.com                  ProLDAPX        “LDAP lookup”
{
     Request-Attribute-For-Search          User-Id
     Filter-Type                           CIS
     Directory-Select-Mode                 Round-Robin
     Directory             “Directory 1”
     {
          URL                   “ldap://ldap1.ispx.com:389”
          Administrator         “cn=...,ou=...,ou=...,o=radius”
          Password              “password”
          SearchBase            “...,ou=...,o=radius”
          Authenticate          Search
          Filter                uid
     }
}
```

Hunt Group RADIUS Proxies Authentication Type

In the `authfile`, the `HGRADIUS` authentication type requires extended parameters to configure the list of servers to proxy requests to.

- A hunt group may be managed as a round-robin (load-balanced) hunt group which round-robins the requests over the members of the group.
- Or a hunt group may be managed as a first-available hunt group where all the requests go to the first member of the group unless it is down, at which time all requests go the 2nd member until the 1st member comes back up.
- A member of a hunt group is declared down if it has not responded to N consecutive requests (N is configurable).
- When a server goes down, it stays down for a while (a configurable hold down timer).
- When all servers in a group are down, the hunt group itself is declared down and all new received requests for this realm's hunt group are rejected. There is also a configurable hold down timer for the hunt group as a whole.
- A home server may be a member of multiple hunt groups (for multiple realms). A home server may be configured for both HGRADIUS for realm X, and as a regular old RADIUS for realm Y.
- There may be more than two members in the hunt group, up to 20.
- A given home server may belong to multiple hunt groups, up to 20.

The extended parameters look like this (required subset is in bold):

```
<realm_name>  HGRADIUS  "<realm_comment>"
{
     server <server_name>
     server <server_name>
     server <server_name>

     Acct-Timeout                <integer>
     Auth-Timeout                <integer>
     Destination-Select-Mode     <string>
     Failure-Limit               <integer>
     Group-Downtime              <integer>
     Request-Tries               <integer>
     Server-Downtime             <integer>
}
```

Parameter	Description
Server (FQDN or IP address)	This defines one server name or IP address of this proxy hunt group. This field is must be present and can be repeated for up to a total of 20 servers.
acct-timeout (integer)	Time (in seconds) that HGRADIUS waits for an accounting reponse before trying next server. The default value is 5.
auth-timeout (integer)	Time (in seconds) that HGRADIUS waits for an authentication reponse before trying next server. The default value is 5.
destination-select-mode (string)	Sets the Hunt Group type to Round-Robin or First-Available. The default value in Round-Robin.
failure-limit (integer)	Number of consecutive RADIUS request failures to declare a server down. The default value is 3.
group-downtime (integer)	Time (in seconds) that a HGRADIUS Group is down before resetting its status to up and thus start sending requests again to that group. The default value is 300.
request-tries (integer)	Maximum number of servers in a HGRADIUS Group to try before giving up on sending a particular request. The default value is 2.
server-downtime (integer)	Time (in seconds) that a HGRADIUS Server is down before resetting its status to up and thus start sending requests again to that server. The default value is 900.

Allow Authentication Type

In the `EAP.authfile`, the `Allow` authentication type is used to specify a realm whose users require no authentication. The *`auth-parameter`* field is empty. For example:

```
fred.com    Allow       ""
```

Specifies that realm fred.com users are just allowed with no authentication.

Note: This should only be used in very special circumstances. Please consult with technical support for guidence.

Deny Authentication Type

In the `EAP.authfile`, the `Deny` authentication type is used to specify a realm whose users are not allowed to authenticate. The *`auth-parameter`* field is empty. For example:

```
fred.com    Deny        ""
```

Specifies that realm fred.com users are **NOT** allowed to authenticate.

Note: This should only be used in very special circumstances. Please consult with technical support for guidence.

Prefixed users and authfiles

In the `clients` file, you may optionally specify a prefix for a client device. For any users connecting through this client, the RAD-Series Server will search for the user's profile in a file called *`prefix`*`.users`. If the user's profile is not found there the server will use the files *`prefix`*`.authfile` and *`prefix`*`.EAP.authfile` to authenticate the user. This allows you to specify different authentication methods for users in the same domain who may connect via different clients under different circumstances, for example: password authentication when connecting through a wired NAS, vs. EAP authentication when connecting through a wireless AP.

If a prefixed file does not exist or no prefix is specified then the regular non-prefixed file will be used. The "`EAP.`" prefixed `authfile` and any other FSM specified prefixed files are mandatory files.

clients

The `clients` file defines the access devices and RADIUS servers that this RAD-Series Server communicates with.

Clients Entry Syntax

***Name* *Secret* type=*Vendor*:NAS** *Options* [*Optional-Fields*]

or

Name:*Auth-Port* ***Secret* type=*Vendor*:PROXY** Options [*Optional-Fields*]

or

Name:*Auth-Port*:*Acct-Port* ***Secret* type=***Vendor*:**PROXY** Options [*Optional-Fields*]

The keywords "`type=`", "`srcip=`", and "`reply_holdtime=`" are case-insensitive.

Examples

```
j.flatland.org      f52tl         type=Ascend:NAS
216.27.61.137       secret        type=Ascend+USR:NAS   v1   west.
a1.flatlink.com     f25lt         type=Merit:Proxy
10.2.7.7:1666:1667  real-secret   type=none:proxy  v1
192.168.1.*         secret        type=Cisco:NAS   v1
192.168.2.0/24      secret        type=Cisco:NAS   v1
2001:3::42          newsecret     type=Ascend:NAS  reply_holdtime=5
[2001:3::69]:1566   somesecret    type=3com:NAS    srcip=[::]:2223
10.10.10.1          realSecret    type=none:proxy  srcip=10.1.1.2
2001:333:2::2222    aSecret       type=none:proxy  srcip=10.1.1.2  set1.
2001:444::/64       bigSecret     type=NAS
```

Parameter	**Description**
Name	Device fully qualified domain name, IPv4 address, IPv6 address or a wildcard pattern (see "Using Wildcards for IP Addresses" on page 27). To use IPv6 addresses you must enable IPv6 in the aaa.config file. See `ipv6_enabled`, "General Server Properties" on page 171.
Auth-Port	Optional authentication relay port to use when proxying authentication requests to this RADIUS home server. This parameter overrides the port specified by the -pp option on the radiusd startup command.
Acct-Port	Optional accounting relay port to use when proxying accounting requests to this RADIUS home server. This parameter overrides the port specified by the -qq option on the radiusd startup command.
Secret	The shared secret between this device and the RAD-Series Server is a required entry. No spaces are allowed.
Vendor	Vendors whose attributes should be returned in reply messages. A list can be entered by combining vendors with a "+" between them, e.g.: Type=Microsoft+Interlink. Enter NONE to prune all VSAs. See vendors file for a list of defined vendors.
{NAS|PROXY}	Device type: NAS or Proxy server. Use NAS for wireless access points. If a NAS fails the RFC conformance checks then you may wish to configure it as a PROXY which will cause the RAD-Series Server to skip the conformance tests. This may require that you add prune, see below.

Parameter	Description
Options	Additional criteria for messages. A list can be entered by combining options with a "+" between them, e.g.: Type=Cisco:NAS+oldchap+debug. No spaces are allowed. The set of options for a NAS are: • noencaps - Do not encapsulate vendor response (if the access device requires non-encapsulated A-V pairs). • oldchap - For access devices that perform pre-RFC CHAP. • debug - Dump packet traces for this client into the RAD-Series Server's debug output file if the server is running at debug level 1 or greater. The set of options for a PROXY are: • prune - Force pruning as if the response were being returned to an NAS. With this option the generic vendor prunes all vendor-specific attributes before a message is returned to the proxy server. This may be used to help prevent problems that might occur if unencapsulated vendor attributes are not correctly mapped in the vendors file. • no_append - This is useful when a remote server does not return all of the A-V pairs that it received in the order they were received. If it is not set, the RAD-Series Server will append all the A-V pairs received from a remote server to the new A-V pairs sent in the response message. • debug - Dump packet traces for this client into the RAD-Series Server's debug output file if the server is running at debug level 1 or greater.
Optional-Field	All other optional fields can be in any order. Each one can appear at most once. The available optional fields are below.
V1	Only needs to be added to a clients entry if you wish to specify a `Prefix`. In that case specify this '`V1`' or '`v1`' followed by the `Prefix` option.
Prefix	Prefix to identify the `users` files and/or `authfile` and/or `EAP.authfile` to use for requests from this client. The prefix is prepended to the normal file name to get the name of the file to use.For example, a prefix of "west." yields `west.users`, `west.authfile` and `west.EAP.authfile` which would be used in place of `users`, `authfile` and `EAP.authfile`. If one of the prefixed files does not exist then its corresponding standard non-prefixed file will be used. This option can only be configured by manually adding it to a device entry in the clients file.

Parameter	Description
Reply_holdtime	After a request has been replied to, it is held for a period of time in case a retransmission is necessary. This option specifies the number of seconds to hold on to a request after it has been replied to. It also is used to determine the time to wait for the Accounting-Start. The value should be twice the default retransmission period of the device involved. This option can only be configured by manually adding it to a device entry in the clients file.
SrcIP	This option specifies the source IP address for proxying requests to this client. The source port may additionally be specified, as an aid for home servers whose firewall allows communications with only certain IP addresses and/or ports. If the client has both IPv4 and IPv6 addresses, this option controls whether the proxied requests are sent as IPv4 or as IPv6 requests. The srcip address may be a specific IPv4 address, a specific IPv6 address, the IPv4 ANY address [0.0.0.0], or the IPv6 ANY address [::]. If the IPv4 ANY address is specified, the operating system assigns the specific IPv4 address. Likewise for the IPv6 ANY address, the operating system assigns the specific IPv6 address. The `srcip` option solves the problem where a home server has both an IPv4 and an IPv6 address but the home server's RADIUS application is listening only on the IPv4 (or IPv6) interface, see “IPv6 OperationIPv6 Operation” on page 269. The srcip can also address the situation where the proxy server has multiple IPv4 or multiple IPv6 interfaces and needs to specify the correct interface from which to send the request to the home server. If IPv6 is not enabled and if the srcip address is IPv6, an appropriate logfile message is generated and the `srcip` option is ignored. If the DNS lookup determines that the client is IPv4 only (or IPv6 only) and the srcip value is IPv6 (or IPv4), an appropriate logfile message is generated and the `srcip` option is ignored.

Realm files (.users) and default users file

Realm files provide local storage of user profiles. Create a separate file for each defined realm to store user profiles for authentication. All realm file names **must** end with the extension `.users`.

The default "users" file is read into memory at startup/HUP and the (indexed) lookups are fast. The realm files are sequentially searched via file I/O for each authentication and changes take affect as soon as the file is updated.

User Entry Syntax

For the default users file the first line of each user entry consists of User-Name, optionally followed by configuration items and check item fields. Subsequent lines may contain an indented list of reply item A-V pairs. All reply item lines except for the last are followed by a comma:

```
User-Name check-items
      reply-item,
      reply-item,
      ...
```

Example

```
guest@library.org Password = "public", Simultaneous-Use = 20
      session-timeout = 3600,
      idle-timeout = 300
```

Note: For realm files the one difference is that the User-name is replaced with the User-Id. The realm name is omitted

Parameter	Description
User-Name	In the default users file this is the user's Network Access Identifier (NAI) format login string. In realm files this is the *userid* portion of user's NAI format login string. Omit the realm name. The *userid* portion must contain only the letters A-Z and a-z, the digits 0-9, and these printable graphic characters: ASCII code 33 = ! (Exclamation mark) ASCII code 35 = # (Number sign) ASCII code 36 = $ (Dollar sign) ASCII code 37 = % (Percent sign) ASCII code 38 = & (Ampersand) ASCII code 39 = ' (Single quote or Apostrophe) ASCII code 42 = * (Asterisk) ASCII code 43 = + (Plus sign) ASCII code 45 = - (Hyphen, minus sign) ASCII code 46 = . (Period, Dot, full stop) ASCII code 47 = / (Slash, forward slash, fraction bar, division slash) ASCII code 61 = = (Equals sign) ASCII code 63 = ? (Question mark) ASCII code 94 = ^ (Circumflex accent or Caret) ASCII code 95 = _ (underscore, understrike, underbar or low line) ASCII code 96 = ` (Grave accent) ASCII code 123 = { (braces or curly brackets, opening braces) ASCII code 124 = | (vertical-bar, vbar, vertical line or vertical slash) ASCII code 125 = } (curly brackets or braces, closing curly brackets) ASCII code 126 = ~ (Tilde or swung dash)
check-items	Any check or deny items to be matched by the Access-Requests before authorizing a user. See the dictionary files for a list of valid attributes. Configuration attributes used for Interlink-specific functions may also appear here. See Configuration Attributes below for a list of valid attributes.
reply-items	Any reply items to be returned to the access device to provision the user session. Each line, except for the last, should be followed by a comma. See the dictionary files for a list of valid attributes. Configuration attributes used for Interlink-specific functions may also appear here and will not be returned to the access device. See the dictionary files for a list of valid configuration attributes.

General Configuration Attributes Used as Check Items

Configuration attributes provide user-level information for Interlink-specific functions. Only the User-Name attribute is required in a user entry. All other configuration attributes are optional.

See the `dictionary` files for a list of valid values for each attribute.

Attribute	Description
Comment	Allows you to provide any necessary explanation for the entry.
Deny-Message	Specifies a string that would be returned to the user in the Access-Reject if any deny item for this user caused a rejection. The Deny-Message is only sent when a deny item comparison fails, not when a check item comparison fails. You may use an asterisk wildcard: Deny-Message = "*" This wildcard sends the message “Access denied” and the deny item that triggered the rejection. For example: Access denied, NAS-Port != 3160
Expiration	In date format, specifies when an entry expires. For example: Expiration = ”Dec 01 2004” When the specified date has been reached, the user will receive an Access-Reject with the message, “Password has expired,” in response to all Access-Requests. See “Date Attribute Value Formats” on page 168 for a description of the acceptable date formats.
Xvalue	Provides a means to pass an integer value to a module on a per user basis. Usually used in Advanced Policy and custom applications.
Xstring	Provides a means to pass a string value to a module on a per user basis. Usually used in Advanced Policy and custom applications.

Attribute	Description
Password	Specifies the value to compare to the RADIUS User-Password or the user's input in response to an Access-Challenge. Do not use the \ character. To store hashed passwords in the realm file, add the Encryption-type to the password value by using the syntax: Password = {*Encryption-type*}*password* Where *Encryption-type* is the hashing algorithm and *password* is the user password. For example: Password = "{md5}CY9rzUYh03PK3k6DJie09g==" Encryption-type can be: • clear • crypt • md5 • sha-1 • ssha • x-nthash • x-lmhash Choose the Encryption-type based on the inner realm authentication method used. The list of Encryption-types can be extended through the Software Developers Kit (SDK).

LAS Configuration Attributes Used as Check Items

These attributes provide information for the RAD-Series Server's Local Authorization Service (LAS) function. To activate this feature, you must enable Session Tracking for the user's realm. The following attribute may be added to a user entry to override the global default.

Attribute	Description
Simultaneous-Use	The maximum number of active sessions the user may have. If Session Tracking is enabled for the user's realm, then the global value set in Server Properties is the default value. Any user specific configuration of Simultaneous-Use will be used instead of the default value. A value of -1 removes the simultaneous session constraint for the user, whose simultaneous session limit is then bound by the RAD-Series Server's overall licensed limit. The value 0 will deny access to the user.

Check and Reply Items

A user entry may include check and reply items to control access and provision service.

Check items are A-V pairs that are compared to A-V pairs in a received RADIUS Access-Request packet. There are two types of check items: **regular check items** and **deny items**. Regular check items are compared to the attribute value in the Access-Request message: the attributed must be present with the matching value, only if so is the user authorized. A deny item is similar: the attribute must either be not present in the Access-Request, or present but with a different value, only if so is the user authorized. Deny items use the operator != (not equal to) instead of = (equal to).

Note: The RAD-Series Server compares a check/deny item in the user profile with the first value that appears for that attribute in an Access-Request. The server will disregard any additional instances of the same attribute in the request. This limitation also applies to tagged attributes, like those used to establish VPN tunnels.

A **reply item** is an A-V pair that is returned in an Access-Accept, Access-Challenge, or Access-Reject message to provide instruction to the access device for provisioning the user. Some of these attributes, such as Session-Timeout, can be used to enforce some simple authorization policies.

Note: The RAD-Series Server handles multiple instances of a reply item in the user profile based on the clients file entry for the recipient. The server will consult the pruning rules in the `dictionary` files to determine which instances to send based on the NAS type. See “clients” on page 224 for details

Some attributes may be used as either check or reply items. In these instances, the attribute may appear in an Access-Request as a hint from the client for a value to assign to the attribute. With the exception of Service-Type and the tunneling attributes, the RAD-Series Server does not resolve hints, but you can use a check item to control access based on hints. For example, if you only wish to authenticate users requesting a Framed service, you could add `Service-Type = Framed` as a check item for those users.

Most check and reply item attributes are defined in the standard RADIUS RFC documentation and the `dictionary` files.

Date type attributes used for check/deny/reply attributes can be entered in many formats. See “Date Attribute Value Formats” on page 168 for a complete description of the rules.

Interlink-specific Attributes

Interlink-specific attributes that may be used as check or reply items are:

Attribute	Description
Day-Of-Week	An integer representing the day of the week, where 0 is Sunday and 6 is Saturday. This attribute is derived from to the current system clock of the machine hosting the RAD-Series Server.
Reply-If-Ack-Message	Allows you to specify a message to return to the user if authentication succeeds. Similar to Reply-Message, but only sent in an Access-Accept packet. The normal pruning operation concentrates all Reply-Messages and Reply-If-Ack-Messages into as few Reply-Messages as possible.

las.conf

The `las.conf` file contains a list of configuration items for the RAD-Series Server's Local Authorization Service function. There are configuration sections for realms and LAS session items. These sections do not have to be maintained in a particular order; however, an object (a realm, for example) must be defined before it may be referenced.

The parameters described here can be manually edited in `las.conf`.

LAS Session Configuration

These parameters let you override the RAD-Series Server’s default values related to session timing and session limits. Timing parameters are specified in seconds.

Syntax

attribute value

Session Attributes

Attribute	Description
Session-Table-Checkpoint-Interval	The interval, in seconds, between saves of the session table if there are any changes. The default value is 300 seconds (5 minutes).

Attribute	Description
Session-Checkpoint-File-Lifetime	This parameter specifies the age of the oldest session.las file that will be accepted by the RAD-Series Server at startup time. The age is relative to the startup time. The default is 28800 seconds (8 hours). By default, the server will ignore a session.las file written 8 or more hours before the server startup time. A value of zero indicates the server should accept the session.las file no matter how old.
Session-Checkpoint-Fork-Threshold	If the number of sessions in the session table is greater or equal this threshold, the RAD-Series will fork a separate process when writing the session checkpoint file at the configured interval. This is a performance tuning parameter to prevent excessive delays to normal request processing during session checkpoint writing. The default is 5000 sessions, and the valid range is 1000 to 2147483647 session entries.
Session-Pending-Timeout	Time in seconds the RAD-Series Server waits for an Accounting-Start before moving a PENDING session into the UNCONFIRMED state. The default is 15 seconds.
Session-Unconfirmed-Timeout	Time in seconds the RAD-Series Server waits before removing a session in the UNCONFIRMED state. The default is 15 seconds.
Session-Collision-Timeout	Time in seconds the RAD-Series Server waits before removing a session in the COLLISION state. The default is 300 seconds (5 minutes).
Session-MIA-Timeout	Time in seconds the RAD-Series Server awaits the next Interim-Accounting message before moving an ACTIVE session into the MIA state, or before removing a session already in the MIA state. The default is 0, a special value that indicates the server will measure the time interval between received Interim-Acct messages, and use that measured value to time out subsequent Interim-Acct messages. A positive value represents a fixed time interval which the server will use to time out the next expected Interim-Acct message, rather than using a measured interval. A large positive value, e.g. 2147483647 seconds (over 68 years) will effectively cause the server to not terminate a session, or move an ACTIVE session into MIA state, due to the absence of an Interim-Accounting message.

Attribute	Description
Session-Dropped-Timeout	Time in seconds the RAD-Series Server waits before removing a session in the DROPPED state. The default is 300 seconds (5 minutes).
Session-Finished-Timeout	Time in seconds the LAS waits before removing a session in the FINISHED state. The default is 45 seconds.
Session-Table-Limit	The maximum number of sessions that can be held in the Session Table. When this number is met, authentication requests that would normally result in a new session are ignored. This should not be confused with the session limit of your license. That limits the number of active sessions. The session table has to hold the active sessions and sessions in various other states waiting to be released. The default is 2147483647 (maximum allowed).
Session-Table-Update-Interval	Time in seconds between updates to the status of sessions. From one to four session timer lists can be created, of increasingly longer update intervals. A given session will reside on at most one timer list. The update interval for the first timer list must be from 1 to 10 seconds, with a default of 5 seconds. Each subsequent timer interval, if any, must be greater than its predecessor. The longest interval cannot exceed 86400 seconds (24 hours). Note: the value of the first timer determines the granularity (precision) of the session management timers. The default is one timer list which is updated every 5 seconds.
Simultaneous-Use	The maximum number of active sessions users may have, unless a user-specific value is configured that overrides this number. The default value is 1. The value -1 removes the simultaneous session constraint for all user and the total number of simultaneous sessions is then bound by the RAD-Series Server's overall licensed limit. The value 0 prevents access to any user that does not have a specific value configured.
Simultaneous-Use-States	This parameter defines which session states count towards a user's Simultaneous-Use limit. The default value is "Pending-Active-Unconfirmed-Expired-MIA", which means that a user session in state PENDING, ACTIVE, UNCONFIRMED, EXPIRED, or MIA will count against the user's Simultaneous-Use limit. These represent states that are ACTIVE or could become ACTIVE. The other supported value is "Pending-Active", which means that a user session in state PENDING or ACTIVE will count against the user's Simultaneous-Use limit; this is compatible with previous versions of the RAD-Series Server.

Attribute	Description
Minimum-Interim-Accounting-Timeout	This parameter sets a floor for the RAD-Series Server's timeout for the next expected Interim-Acct request message, even if the server's measured value is less than this value. The default is 60 seconds.
Interim-Accounting-Grace-Period	Time in seconds the RAD-Series Server will wait for an Accounting message (Interim or Stop) before removing a session in the MIA state. The default is 15 seconds.
Max-Number-Of-Acct-Records-Per-Second	The RAD-Series Server will internally generate an Acct-Stop record for a session which is terminating without benefit of having received an Acct-Stop request from the NAS, such as sessions which terminate in COLLISION or MIA state. The parameter limits the server's rate of generating such accounting records, to avoid overtaxing of resources. For example, a server with thousands of sessions might be suddenly tasked with writing many accounting records upon receiving an Accounting-Off request from a NAS; this parameter can be used to distribute that workload over several seconds. The default is 100 server-generated accounting records per second.
Session-Id-Prefix	The Session-Id-Prefix is an alphanumeric character string, 2 to 8 characters in length. The RAD-Series Server generates a session-id which begins with this character string. The default value is "AAA". The Session-Id Prefix helps distinguish the Class attribute which contains the RAD-Series Server's session-id from other Class attributes that may be present in a RADIUS message.
Accounting-OnOff-Support	When this option is set to Yes, the RAD-Series Server will clear all sessions for a NAS who sends an Accounting Request message with Acct-Status-Type=Accounting-On or Acct-Status-Type=Accounting-Off. When set to No, the Accounting Request will be acknowledged, but the server will take no action regarding the NAS's sessions. The default is Yes.

Attribute	Description
Roaming Accounting	When this option is set to Yes, the RAD-Series Server will, when processing an Acct-Interim for a given user+session, check if there is another ACTIVE session for the same user which has the same Acct-Session-Id. Such a session, if found, is treated as the previous subsession of a roaming session. This associated earlier subsession is transitioned into EXPIRED state, and after a short grace period, is ended and a server-generated Accounting-Stop record is produced. This option will thus ensure the termination of each subsession of a roaming session, and will ensure the generation of an accounting record for each subsession of a roaming session. When this option is set to No, the RAD-Series Server will not, when processing an Acct-Interim for a given user+session, check if there is another ACTIVE session for the same user which has the same Acct-Session-Id. Thus each earlier subsession of a roaming session will remain ACTIVE until either [a] an Accounting-Stop is received for that subsession, or [b] the subsession times out due to the expiration of the Acct-Interim timer. The default is No.
Extra-Logging	If set to Yes, the RAD-Series Server will log additional information regarding session events. For example, the RAD-Series Server will always log notifications of unexpected session transitions, such as transitions into COLLISION or UNCONFIRMED states. If this parameter is set to Yes, the server will also log normal transitions, such as from ACTIVE to FINISHED state. This parameter is intended for those wanting closer monitoring of session events.
Session-Collision-Checking	When this option is set to Yes, the LAS will check if a newly-received authentication request comes from the same NAS (as identified by the NAS-Identifier or NAS-IP-Address or NAS-IPv6-Address attribute) and port (as identified by the NAS-Port attribute) as an existing active session. If so, the existing session is put into an non-active state and the newly-received authentication request becomes the new sole owner of that NAS/Port. When set to No, the RAD-Series Server will not check for concurrent usage of the same NAS and port. This parameter is intended for use by NASes which distinguish their ports and allow only one active session on a given port, to recognize and clear out previous sessions on the given NAS/Port for which the LAS did not receive a termination notice (normally an Accounting-Stop message). The default is Yes.

Generic Tokenpool Configuration

This section describes the configuration and use of generic token pools.

Generic token pools can be used to limit or monitor access to resources. The RAD-Series Server tracks the number of tokens currently in use, as well as the high-water mark of tokens in use, for each token pool. The server tokenpool statistics are reported by the radcheck utility.

During user authentication, a token can be requested for a user in one of two ways:

- A user can be explicitly configured with one or more Token=<poolname> reply-items, where Token is an internal dictionary attribute and <poolname> is the name of a token pool matching a configured Tokenpool, or
- A policy step can evaluate the user’s Access-Request and determine if one or more token requests should be made for this request, and if so appending Token=<poolname> attribute(s) to the request.

If a token is requested for a session-tracked session from a non-empty token pool, a token is allocated from the pool and assigned for the duration of the session. When the session ends, the token is returned to the pool for re-use by another session. The accounting logfile will contain attributes reflecting the tokens used by the session. A session may request tokens from multiple pools. Configuring a large number of tokens allows resource usage to be monitored without being limited.

If a token is requested for a session that isn’t tracked, or if the specified token pool is empty, the user’s access request is rejected.

Syntax

```
Tokenpool     pool-name    number-of-tokens
```

Tokenpool Attributes

Attribute	Description
pool-name	The name of a defined token pool
number-of-tokens	The number of tokens in that pool

LAS Realm Configuration

This section lists session-tracked realms by name. To turn on Session Tracking you must add the realm entry to `las.conf`.

Syntax

```
Realm realm-name
End-Realm
```

Realm Attributes

Attribute	Description
Realm	Required, defines the name of a realm whose sessions will be tracked.
End-Realm	Optional, defines the end of the Realm block

Logging LAS Realms

The default RAD-Series Server behavior is to log accounting messages locally, whether the server processes Access-Request messages locally or sends them to a remote server. If a realm entry exists in `las.conf`, the server will NOT send accounting messages to the server that processed the authentication for the corresponding user.

vendors

The `vendors` file contains a list of entries defining vendors whose attributes are recognized by the RAD-Series Server. Each vendor entry contains a vendor name and vendor number. The vendor numbers are SMI Network Management Private Enterprise Code numbers (integers), as managed by IANA.

Entries may optionally contain an interim way of mapping external (with respect to the RADIUS server) attribute numbers to internal vendor-specific attributes. This optional mapping is used in RADIUS requests and responses. This is used primarily to support very old NAS implementations. It should not be necessary when adding new vendors.

Version Tracking

It is possible to track different versions of the vendors file by changing the following line in the file:

%VENDORSID *Version-String*

`Version-String` is the version identifier. If you need to modify the vendors file, it is recommended that you modify the version string by adding to the end of the current string. This string will appear in `radcheck` output.

Vendor Entry Syntax

```
Attribute-String   Value-String Vendor-Code Vendor-Name   Type-Field-Size=n
Length-Field-Size=m
{
    Standard-Value  Vendor-Specific-Value
    . . .
}
```

Parameter	Description
Attribute-String	An optional string that defaults to ATTRIBUTE when not specified. When adding vendor-specific attributes to the dictionary files, enter the string that will be used to identify the attributes. You must also specify Value-String.
Value-String	An optional string that defaults to VALUE when not specified. When adding vendor-specific values to the dictionary files, enter the string that will be used to identify the values.
Vendor-Code	The private enterprise number assigned by IANA.
Vendor-Name	Vendor identifier that will appear in the clients file as a type=vendor:nas entry or in the dictionary files and .users files in vendor-specific attributes.
Type-Field-Size	Type-Field-Size is the length in octets of the <type> field in this vendors's VSAs as transmitted in RADIUS messages. The valid range is 1 to 4. This parameter is optional and defaults to 1.
Length-Field-Size	Length-Field-Size is the length in octets of the <length> field in this vendors's VSAs as transmitted in RADIUS messages. The valid range is 0 to 4. his parameter is optional and defaults to 1.
Standard-Value	The external or common attribute number as seen in RADIUS requests on the network. Must be mapped to Vendor-Specific-Value.
Vendor-Specific-Value	The internal attribute number.

Standard-Value and Vendor-Specific-Value are used to map attributes from the common attribute space defined in the RADIUS RFC to internal non-conflicting vendor-specific attributes. They address backward compatibility issues for very old NASes. You should not have to use these parameters for new VSA definitions.

Examples

```
Shiva.attr  Shiva.value   166   Shiva
(
     91     91
     92     92
       . . .
)
```

```
MS.attr     MS.value      311   Microsoft

USR.attr    USR.value     429   USR Type-Field-Size=4 Length-Field-Size=0
```

dictionary

The `dictionary` files contains a list of dictionary translations that the RAD-Series Server uses to parse incoming requests and to generate outgoing responses. It includes definitions of all attributes and their permitted values.

Version Tracking

It is possible to track different versions of the dictionary by changing the following line in the file:

`%DICTID` *`Version-String`*

`Version-String` is the version information. If you need to modify the dictionary, it is recommended that you modify the version string by adding to the end of the current string. This string will appear in `radcheck` output.

Partitioning of dictionary into Multiple Files

The dictionary files support an `%INCLUDE` directive, allowing the dictionary to be partitioned into multiple parts. This facilitates the isolation of custom modifications while allowing standard updates of other attributes.

The dictionary provided by Interlink Networks is distributed over multiple dictionary files. The main dictionary file contains the standard RADIUS attribute definitions. Each vendor's VSAs are defined in a separate vendor-specific dictionary file, for example Interlink attributes are defined within a `dictionary.Interlink` file. The Interlink VSAs are incorporated into the server's complete internal dictionary by the presence of a `%INCLUDE dictionary.Interlink` directive within the `dictionary.VSAs` file. All vendors VSAs are incorporated into the server’s complete internal dictionary from the `dictionary.VSAs` file.

A `dictionary.custom` file is provided for defining customer-specific VSA definitions and any other additions that they need. Use the `%INCLUDE dictionary.`*`<vendor name>`* in the `dictionary.custom` file to add a new vendor’s VSAs.

Attribute Entry Syntax

{ATTRIBUTE|*Attribute-String*} *Attribute-name Integer-encoding Type (pruning) # comment*

Parameter	Description
Attribute-String	Vendor-specific attribute identifier defined in the vendors file. If no Attribute-String is defined, use ATTRIBUTE.
Attribute-name	Unique name of an attribute.

Parameter	Description
Integer-Encoding	Actual attribute number code.
Type	The attribute type. May be: • unsigned8: 8-bit unsigned integer value • unsigned16: 16-bit unsigned integer value • integer: 32-bit value in big endian order (high byte first) • signed32: signed 32-bit value in big endian order (high byte first) • signed64: signed 64-bit value in big endian order (high byte first) • unsigned64: unsigned 64-bit value in big endian order (high byte first) • date: 32-bit value in big endian order (seconds since 00:00:00 GMT, Jan. 1, 1970) • octets: 0-253 undistinguished octets • abinary: 0-253 Ascend binary filter octets • string: 0-253 octets • vendor: 0-253 octets with octets 0-3 representing the IANA number • ipaddr: IPv4 address: 4 octets in network byte order • ipv6addr: IPv6 address: 16 octets, in network byte order • ip46addr: IPv4 or IPv6 address: 4 or 16 octets, in network byte order • ipv6prefix: 2 octets of prefix length followed by 0-16 octets of IPv6 prefix, in network byte order • interfaceid: Interface id: 8 octets in network byte order • tag-int: single octet followed by three octets of integer value (used for tunneling attribute) • tag-str: single octet followed by 0-252 octets (used for tunneling attribute) • tlv: A variable number of octets (binary), where the value is structured as a collection of subattributes
Pruning	Optional pruning expression that controls: • Sending A-V -pair to NAS • Logging the attribute • Encapsulation • Internal attribute identification
Comment	Optional comment about the entry.

Pruning Expressions and Flags

Pruning is a feature that allows the RAD-Series Server to remove A-V pairs from an Access-Accept, Access-Reject, or Access-Challenge message before sending the message to a client (generally a NAS) that has been configured for pruning in the clients file (Type=NAS or Type=proxy+prune). The pruning is defined by optional pruning expressions in the dictionary attribute entries.

Pruning is expressed in an attribute entry as follows:

```
(ack, nak, chall, NOLOG, NOLOG_MERIT, CONFIG, FRAGMENTABLE, INTERNAL,
ENCRYPT_SALT, MASK_VALUE)
```

If the `ack`, `nak`, or `chall` value is omitted, but the comma is present for that expression, the default value of 0 is used. If no pruning expressions are specified, all defaults are used.

`ack, nak, chall`: How many instances of the dictionary attribute to add to an Access-Accept, Access-Reject, or Access-Challenge, respectively. May be one of the following values:

- 0—No attributes of this kind (default value)
- 1—One attribute of this kind (last occurrence on the reply list)
- *—any number of attributes of this kind

`NOLOG`: Do not add the attribute to RAD-Series Server log file or session logs.

`NOLOG_MERIT`: Do not add the attribute to RAD-Series Server Merit accounting log file.

`CONFIG`: This option means the attribute is only for the internal use only. The CONFIG pruning option may not be combined with any other pruning option.

`INTERNAL`: This option means the attribute is for internal use only and the attribute will not be sent or received.

`FRAGMENTABLE`: The values of all occurrences of the attribute are internally concatenated to a single occurrence with a long value when received, and fragmented (if necessary) into multiple occurrences when transmitted. An EAP-Message is an example of a FRAGMENTABLE attribute.

`ENCRYPT_SALT`: The attribute's value is encrypted using the generic salt encryption algorithm when transmitted in RADIUS messages.

`MASK_VALUE`: The attribute's value is masked when logged.

Note: Always specify pruning expressions for vendor-specific attributes or they will not be returned to the client in any replies.

Examples

```
ATTRIBUTE         Framed-Protocol         7     integer (1, 0, 0)

ATTRIBUTE         Password                133   string  (CONFIG,MASK_VALUE)
ATTRIBUTE        Tunnel-Password         69    tag-str (*, 0, 0,ENCRYPT_SALT)

Merit.ATTRIBUTE User-Realm               223   string  (*, 0, 0, INTERNAL)

Interlink.Attr   Address-Pool            1     string  (0, 0, 0, INTERNAL)

WiMAX.attr       WiMAX-Time-Of-Day-Time  20    TLV     (*,0,0)

    # Subattributes of WiMAX-Time-Of-Day-Time

    WiMAX.attr  Hour                     20.1  unsigned8

    WiMAX.attr  Minute                   20.2  unsigned8

    WiMAX.attr  UTC-Offset               20.3  signed32
```

Generic Salt Encryption

The values of some attributes are always transmitted in an encrypted form, using the generic salt encryption algorithm. The `Tunnel-Password` attribute (*RFC2548*) and the `MS-MPPE-Recv-Key` and `MS-MPPE-Send-Key` attributes (*RFC2868*) are examples of such attributes. The generic salt encryption algorithm is defined in these RFCs. Some vendor specific attributes, e.g. `Unisphere-Med-Ip-Address`, are also transmitted in encrypted form, using the same generic salt encryption algorithm.

The RAD-Series server recognizes an attribute as needing generic salt encryption/decryption if the attribute is configured with the ENCRYPT_SALT flag in the dictionary. A second flag, MASK_VALUE, can be additionally configured, which causes the RAD-Series server to mask the value of sensitive attributes when displayed in logfiles.

The RAD-Series Server supports the salt-encryption of all attribute types, i.e. abinary, date, integer, interfaceid, ipaddr, ip46addr, ipv6addr, ipv6prefix, octets, signed32, signed64, string, tag- int, tag-str, unsigned8, unsigned16, and unsigned64.

The `radpwtst` utility supports the sending and receiving of salt-encrypted attributes. The attributes to be sent should be specified unencrypted and `radpwtst` will encrypt them before sending e.g. "`radpwtst -:Tunnel-Password=:1:mypassword ...`"

Value Entry Syntax

Valid values for integer-type attributes may be defined in the `dictionary` files. Define each value on a separate line.

{*Value-String*|VALUE} *Attribute-Name Value-Name Integer-Encoding*

Parameter	Description
Value-String	Vendor-specific value identifier string defined in the vendors file. If no value-string is defined, use VALUE.
Attribute-Name	The name of the attribute associated with this value.
Value-Name	The name/descriptor of the value.
Integer-Encoding	The actual value used in the A-V pair data format.

Examples

```
VALUE              Framed-Protocol  PPP    1
VALUE              Framed-Protocol  SLIP   2

Interlink.VALUE    EAP-Type         TLS    13
Interlink.VALUE    EAP-Type         PEAP   25
```

log.config

The `log.config` file specifies how accounting logs are generated in the RAD-Series Server. It allows you to configure multiple logging streams, which can be used with sophisticated Finite State Machine (FSM) tables. For most applications, configuration of the default stream will suffice.

For any stream, it is possible to configure log file format, location, name, and how often streams are switched between files.There are six possible entry types that control special logging features and options for a stream.

Default Entry

The `default` entry specifies the name of the stream to use as the default and follows the syntax:

```
default stream-name
```

Default Path Entry

The `default-path` entry specifies the default path for session log files. The `default-path` does not apply to the `*default*` stream. The `default-path` follows the syntax:

```
default-path pathname
```

Default-dateformat Entry

The `default-dateformat` entry specifies the default format for printing date type attributes in the accounting log files and follows the syntax:

```
default-dateformat date-format
```

The *`date-format`* is a strftime format string that defines how the date type attributes should be displayed. The conversion specifications of strftime can be found by doing a ‘`man strftime`’. An example strftime format is `‘%Y-%m-%d %H:%M:%S’` which would yield a date in this format: ‘`2016-01-30 16:12:59`’. If this optional entry is not present then all the date type attributes in the accounting logfiles will be displayed in the original format as follows:

LOG_V2_0 (= default Merit) accounting displays its date value as yyyy-mm-dd,
e.g. “`2016-02-24`” in local time zone.
LOG_ACCT (=Livingston accounting) displays a date value “mmm d yyyy”,
e.g. “`Feb 24 2016`", in local time zone.

Default-dateformat-timezone Entry

The `default-dateformat-timezone` entry specifies the default time zone used for printing date type attributes in the accounting log files and follows the syntax:

```
default-dateformat-timezone { gmt | utc | local }
```

The `default-dateformat-timezone` parameter controls whether the date type attributes displayed with the `default-dateformat` string are presented as UTC or in the local time zone. It does not cause the time zone mnemonic to be displayed. If you want the time zone displayed, then use the %Z in `default-dateformat` or `dateformat` if it works on your system or just add the correct characters to `default-dateformat` or `dateformat`, like this `'%Y-%m-%d %H:%M:%S EDT'`.

The default, if `default-dateformat-timezone` is not present, is the local time zone. The `default-dateformat-timezone` parameter applies only to the configured `default-dateformat` and the `dateformat` parameter, defined below. If neither of these is configured, then `default-dateformat-timezone` is ignored.

The values 'gmt' and 'utc' are synonyms.

Nodefault Entry

The one-keyword `nodefault` entry turns off the defaulting feature. When this entry is used, the default stream name must be identified by the `STRVALUE=keyword` in the finite state machine. Don't use this unless you have some idea of how to manipulate the finite state machine.

Stream Entry

```
stream name {
     aatv                   AATV_NAME
     aatv-value             integer
     filename               string
     buffer                 integer
     chmod                  octal-value
     close                  {on|off}
     dateformat             string
     dateformat-timezone    {gmt|utc|local}
     debug                  integer
     dont                   attribute attribute . . .
     gmt|local
     join                   joined_stream
     header                 {none|type|full}
     on-endfile             shell-command
     path                   pathname
     update                 seconds
     wrap                   integer
}
end
```

Parameter	**Description**
stream	The name following this keyword identifies the stream.
aatv	This required parameter for any stream is the AATV to use for logging the stream. • LOG_ACCT (Livingston/Lucent/RABU style call detail format) • LOG_ALL (logs all streams defined in log.config) • LOG_BRIEF (simple session format) • LOG_BY_ATTRIBUTE • LOG_BY_NAS • LOG_BY_REALM • LOG_TACACS+ (Cisco TACACS+ accounting record format) • LOG_V2_0 (default Merit style logging)
aatv-value	The Xinteger value to pass to the AATV. The default is 7.
buffer	The number of records to buffer before flushing the buffers to disk. The default is 1.
chmod	The value is an octal UNIX permission value; must have a leading 0. The default is 0640.
close	The value of `on` causes the RAD-Series Server to open/close file for each flush. The value of `off` causes the server to not open/close file for each flush. The default is off.
dateformat	This strftime format string works just like `default-dateformat`, defined above, except that it applies to this logging stream only. The default, if not present, is to use the `default-dateformat` string if it is present, else to use the old format of 'yyyy-mm-dd' (LOG_V2_0) or 'mmm dd yyyy' (LOG_ACCT)

Parameter	**Description**
dateformat-timezone	This controls whether the date attributes displayed with the `dateformat` string are presented as UTC or in the local time zone. It works just like `default-dateformat-timezone`, defined above, except that it applies to this logging stream only. The default, if not present, is `default-dateformat-timezone` if it is present, else it is the local time zone. The `dateformat-timezone` parameter applies only to the configured `dateformat` parameter. If `dateformat` is not configured, then `dateformat-timezone` is ignored.
debug	The value specifies the logging debug level for accounting messages. If the RAD-Series Server is running at least at debug level 1, then if the logging debug level is > the current debug level, raise the debug level to the logging debug level while we are logging the accounting message. Valid values are 0 to 4. The default is 0.
dont	The value is a list of one or more RADIUS accounting attributes that should not be included in the session logfile. The default is none.
filename	The file name format to use for this stream. The default is "session.%Y-%m-%d.log".
utc|gmt|local	Determines whether to evaluate the session filename format string using UTC (GMT) time or local time. UTC and GMT are synonyms. The default is local.
header	Specifies the amount of header information to be written to the session logfile. `none` adds no header lines, `type` adds just the first 3 lines of header information and `full` adds all the header lines. The default is full.
join	Specifies the stream name of another stream to join to this stream. The joined stream must already be defined. This will allow logging the same accounting message to multiple streams (i.e.: produce multiple outputs.). There is no default.

Parameter	Description
on-endfile	This specifies a shell command to run when the session logfile is rolled to a new filename. Do not enclose the command in quotes. If the command has a “!” in it, then, before executing the command, the “!” will be replaced by the file name in use before the rollover occurred. A common usage is to compress the accounting log at rollover time via on-endfile gzip -9 ! There is no default.
path	The value specifies the path to be used for session log files. Do not enclose the path in quotes. The default is the -a option used when starting radiusd or `/var/opt/aaa/acct` if no -a option was used.
update	The value is the interval in seconds between flushes of the buffer to the session logfile. The default is 900 seconds (15 minutes).
wrap	The number of A-V pairs to put on a line before wrapping to a new line. The default is 3.
end	This required parameter for any stream is the keyword that tells the RAD-Series Server to stop reading the configuration file, allowing subsequent text to be ignored.

Logging Multiple Streams

To log multiple streams, define a stream named `*default*` with the `aatv` subcommand set to `LOG_ALL`. All other streams defined in log.config will also generate accounting logs.

```
stream *default* {
                aatv  log_all
}
stream livingston {
                aatv  log_acct
                buffer      1
                close on
                filename    livingston.%Y%m%d.log
}
stream new {
                aatv  log_v2_0
                aatv-value  7
                buffer      1
                close on
                filename    merit.%Y%m%d.log
}
end
```

iaaaAgent.conf

The `iaaaAgent.conf` file contains one parameter, `agentxPingInterval`, which is set to 30 seconds by default. This setting specifies how often the RAD-Series Server's SNMP subagent will check to see if a master agent is active. If one is detected and the subagent has not already registered with the master agent, the subagent will register with the master agent and begin processing SNMP requests.

Finite State Machine (FSM)

The main component of the RAD-Series Server's software engine is the Finite State Machine and a few associated routines. At RAD-Series Server startup the finite state machine loads state table instructions from an `.fsm` file. It will load the `radius.fsm` file, unless it is missing or another .fsm file is specified by the `radiusd -f` option.

The .fsm file defines a state table that includes the states, events, and actions that determine how a request is processed.

Version Tracking

It is possible to track different versions of state tables by adding the following line to the file:

```
%FSMID Version-String
```

- The `%` **must** be in the first column.
- `Version-String` is the version information. This string will appear in `radcheck` output.

States

Each state defined in a finite state table starts with a line containing just the name of the state, followed by a colon character. There can be up to 1,024 states defined. Each subsequent line is an event handler with three required and two optional fields. The fields on the line can be separated by spaces and/or tabs:

```
State-name:
     Event-1   Action-1  Next-state-1   intattr=value   stringattr=value
     ...  ...  ...
     Event-n   Action-n  Next-state-n   intattr=value   stringattr=value
```

- Every `State-name` **must** start in column 1.
- Every `Event-n` **must not** start in column 1.
- Every `State-name` referenced in an event handler must be defined only once in the state table.
- Every `State-name`, except for the `End`, must have at least one associated event handler.
- Every `State-name`, except for `Start`, must be referenced by at least one event handler in another state as its next state.

The parameters for any event are:

Parameter	Description
State-name	An arbitrary string which represents a state in the FSM. It may be composed of any printable ASCII character except space, new line, carriage return, tab, and colon (:) characters. • Every *`State-name`* must start in column 1. • Every *`State-name`*, except for `Start`, must be referenced by at least one event handler in any state as its next state. • Every *`State-name`*, except for `End`, must have at least one associated event handler. • Every *`State-name`* referenced in an event handler must be defined only once in the finite state machine.
Event	Three-tuple with each part separated by a period in the form: *`Last-state.Last-action.Event-name`* • *`Last-state`* must not start in column 1. • *`Last-state`* is the name of the state that generated the event or an asterisk character (*)—that will match any state—if there is no last state for the event or if the last state does not matter. • *`Last-action`* is the name of the AATV that generated the event or an arbitrary string (found in the code or arrived in a packet), prefixed with a plus character. This may be an asterisk character (*)—that will match any action—if there is no last action or if the last action does not matter. • *`Event-name`* is the name of the event-code returned from Last-action.
Action	Name of the module to call in this event. The called module will return a value that will be used as the next event. For reference, you may refer to a list of "Predefined Actions" in the SDK documentation. Typically, the RAD-Series Server invokes iaaaRealm upon receipt of an authentication request. iaaaRealm in turn invokes the proper authentication module (ProLDAP, SecurID, etc.), depending on the configuration of the request in question. This process is specific to the server's default state table.
Next-state	Name of next state in the AAA transaction. The current *`State-name`*, Action, and the value returned from the Action (an event) determine which event listed under *`Next-state`* should be processed next.

Parameter	Description
intattr=value	An optional integer attribute A-V pair whose value is passed to an Action as an argument. Value may be either the integer or the symbolic value defined for the attribute in the `dictionary`. Only one integer argument may be specified for each event.
stringattr=value	An optional string attribute A-V pair whose value is passed to an Action as an argument. Value may be any string. The string must be less than 64 characters long. Only one string argument may be specified for each event. With the POLICY module, use the `Xstring` attribute to specify an URL that identifies where the policy definitions are stored. See “**Using Advanced Policy**” on page 110 for instructions.

Events

After an action completes its task, it returns an event code name to the FSM. The previous state and action and the event code name determine the current event, which in turn determines the next action of the FSM. The event code names returned by the standard AAA actions are predefined. New names can be created by modifying the .fsm file.

Predefined Event Names

An action may return one of the following predefined events:

Event	Description
ACC_CHAL	The incoming proxy reply is an Access-Challenge or an Access-Challenge should be sent in response to an Access-Request.
ACCT	The incoming request is an Accounting-Request.
ACCT_ALIVE	The incoming Accounting-Request is an interim accounting message.
ACCT_OFF	Received accounting message has a Status-Type of Accounting-Off.
ACCT_ON	Received accounting message has a Status-Type of Accounting-On.
ACCT_START	Received accounting message has a Status-Type of Start.
ACCT_STATUS_TYPE_UNKNOWN	Received accounting message is missing the [required] Status-Type attribute.
ACCT_STOP	Received accounting message has a Status-Type of Stop.
ACCT_TUNNEL_LINK_REJECT	The incoming Accounting-Request is a message that the user has been denied access to an established tunnel.
ACCT_TUNNEL_LINK_START	The incoming Accounting-Request is a message to start a session through an established tunnel.
ACCT_TUNNEL_LINK_STOP	The incoming Accounting-Request is a message to end a session through an established tunnel.
ACCT_TUNNEL_REJECT	The incoming Accounting-Request indicates that a requested tunnel could not be established.
ACCT_TUNNEL_START	The incoming Accounting-Request is a message to establish a tunnel.
ACCT_TUNNEL_STOP	The incoming Accounting-Request is a message to eliminate a tunnel.

Event	Description
ACK	Acknowledgment of the previous action.
AUTHEN	The incoming request is an Access-Request.
AUTHENTICATE	The incoming request is a proxied Access-Request.
AUTH_ONLY	Received Access-Request has a Service-Type of Authenticate-Only.
CHAL_WAIT	The incoming proxy reply is an Access-Challenge or an Access-Challenge should be sent in response to an Access-Request.
CONTINUE	The incoming Access-Request is a continuation of an in-progress EAP conversation. Generally, you should allow the RAD-Series Server to handle these events without modification. This event is not pre-defined, it must be defined in the FSM file.
DROP	The request should be dropped, no further processing will occur. This event is not pre-defined, it must be defined in the FSM file with a value that matches the value of DROP for the Interlink-Reply- Status attribute in the dictionary.
DUP	The incoming Access-Request is a duplicate. Generally, you should allow the RAD-Series Server to handle these events without modification.
ERROR	The previous action generated an error. Generally, you should allow the RAD-Series Server to handle these events without modification.
LAS_ACCT	The incoming request is a LAS proxied Accounting-Request.
NAK	Negative acknowledgment of the previous action.
NO_SUCH_REALM	This value may be returned by the iaaaRealm action when a user's realm cannot be found in the file that iaaaRealm checked (authfile by default).
PROXY_EGRESS	This value may be returned by the RAD2RAD AATV (RADIUS proxy) module to indicate that a request is about to be forwarded. In the default FSM this invokes the proxy post-processing policy. This event is not pre-defined, it must be defined in the FSM file.
PW_EXPIRED	This value may be returned by the iaaaAuthenticate AATV module if the user profile includes an out-of-date value for the Expiration configuration attribute.

Event	Description
RETRIEVE_DEFAULT	This value is returned by the iaaaRealm or iaaaUsers action when the search for a user's profile returns the DEFAULT entry.
RETRIEVE_ERROR	This value may be returned by the user profile retrieval (iaaaRealm, iaaaUsers) when a user's profile cannot be found.
RETRIEVE_SUCCESS	This value may be returned by the iaaaRealm or iaaaUsers action when the search for a user's profile is successful.
RETRY_LIMIT	The number of received duplicate requests has exceeded the retry limit.
SEND	Typically used after a post-processing policy to cause the request to be forwarded or the reply to be sent. This event is not pre-defined, it must be defined in the FSM file.
TIMEOUT	The request has timed out due to inactivity.
TIMER	The timer value has expired.
WAIT	The previous action generated a pending event. Generally, you should allow the RAD-Series Server to handle these events without modification.

Creating New Event Names

To create custom event names, add to the .fsm file, prior to any States:

`%event` *`name`*

- The % must be in column 1.
- *`name`* can be any alphanumeric string and may include underscores (_). The RAD-Series Server will internally assign a unique numeric code for this *`name`*.
- You may define a new event anywhere in the .fsm file, but it must be defined before it is referenced.

Actions

The actions in the state table correspond to the RAD-Series Server's AATV actions. These actions perform discrete functions, such as initiating an authentication request, replying to an authentication request, or logging an accounting record. For any action used in the .fsm file, there must be a corresponding AATV of the same name.

The following table lists some of the available actions.

Action	Description
ACCT	Writes Livingston call detail records
ACCT_SWITCH	Direct FSM to next state based on reason code of the Accounting-Request
ACC_CHAL	Returns an ACC_CHAL event
ACK	Signifies success
CHK_DENY	Verifies check items in user profile.

Action	Description
CONTINUE	Resume processing of an in-progress EAP conversation
iaaaAuthenticate	Performs authentication (following profile retrieval)
iaaaFile	Attempts to retrieve a user profiles from realm users files
iaaaRealm	Attempts a realm-based dispatch using the specified authfile to locate where a user profile is stored or the authentication type for the realm extracted from a user request
iaaaUsers	Retrieves user profiles from the default users file
LAS	Evaluates realm-based authorization for L.A.S.
LAS_ACCT	Initial action to handle an Accounting-Request for L.A.S.
localFile	Retrieves user profiles from realm files based on the Request-Attribute-For-Search option which defaults to User-ID
LOG	Writes accounting logs as defined in log.config
NAK	Returns an NAK event
NULL	No action placeholder.
PASSWD	Verifies password against Unix password system
POLICY	Evaluates advanced policies
POSTLAS	Saves, in the session entry, the IP address allocated by the DHCP-Relay, if one was assigned.
ProLDAP	Retrieves user profile from an LDAP server
ProxySend	Forwards proxy requests
RAD2RAD	Prepares to send RADIUS proxy requests
ReplyDispatch	Translates the Interlink-Reply-Status attribute to an FSM event
ReplyPrep	Prepares to generate reply messages prior to post-processing policy
ReplySend	Generates reply messages after post-processing policy
RequestDispatch	Translates the Interlink-Proxy-Action attribute to an FSM event

Action	Description
SECURID	Performs authentication with a RSA SecurID Authentication Manager versions 6.1.2 and later, 7.1 SP2, 7.1 SP3 and 8.1 SP2 and later.
SRV_STATUS	For Status-Server (Management-Poll) requests
TIMEOUT	Performs timeout logging and provides feedback to other actions that have pending events
TUNNELING	Encrypts Tunnel-Password and resolves hints from client

Predefined .fsm Files

The following FSM tables can be read by the radiusd program at RAD-Series Server startup:

.FSM	Description
radius.fsm	Basic functions. This is the default read by the RAD-Series Server at startup, unless another .fsm is specified.
logall.fsm	Logs all accounting messages in the format specified in the log.config file.
DAC.fsm	A Dynamic Access Control (time-based) finite state machine table. See “**Dynamic Access Control**” on page 160 for details.
DNIS.fsm	A DNIS routing finite state machine table. See “**DNIS Routing**” on page 162 for details.
proxyacct.fsm	The default finite state machine table modified to proxy all accounting requests while maintaining session state locally.

The default `radius.fsm` file is created from the predefined `/etc/opt/aaa/default.fsm` during RAD-Series Server installation. Unless you choose to use an alternate state table with the ‘`-f` *<fsmtable>*’ startup option, always edit `radius.fsm` and have the `radiusd` program read `radius.fsm`.

Managing RADIUS Sockets

The RAD-Series Server can listen for requests on a particular local IP address and port, or on multiple IP addresses and ports. Similarly the RAD-Series Server can proxy requests from a particular IP address and port, or proxy requests using different source IP addresses and ports for different servers.

Managing RADIUS Listen Sockets

The RAD-Series Server can open up to eight sockets on which to listen for authentication requests, and up to eight sockets on which to listen for accounting requests. By default, the RAD-Series Server will listen for authentication requests on one IPv4 socket and will listen for accounting requests on one IPv4 socket

The RAD-Series Server is configured with a set of zero or more IPv4 addresses, and zero or more IPv6 addresses, on which to listen. Each listen IP address is further configured with one or two listen ports: an authentication port and/or an accounting port. The RAD-Series Server opens a separate listen socket for each IP address/Port pair. The RAD-Series Server may be configured to open only authentication ports and no accounting ports, or vice versa.

The listen sockets are configured as a set of `radius_socket{}` configuration blocks. See “RADIUS Listen Socket Properties” on page 199 for a detailed description of this configuration block.

For example, the following configuration would listen for IPv4 authentication and accounting requests on interface 192.168.2.2, and listen for IPv6 authentication requests (but not accounting requests) on interface 2001:abc::4.

```
radius_socket
{
        ipaddr          192.168.2.2
        authport        1812
        acctport        1813
}
radius_socket
{
        ipaddr          2001:abc::4
        authport        1812
}
```

The configured listen IP address can be a specific IPv4 address, a specific IPv6 address, the IPv4 ANY address [0.0.0.0], or the IPv6 ANY address [::]. If the IPv4 ANY address is specified, the RAD-Series Server will receive requests which are sent to any of its IPv4 interfaces. If the IPv6 ANY address is specified, the server will receive IPv4 and IPv6 requests which are sent to any of its interfaces. It is recommended that you configure specific IPv4 and IPv6 addresses instead of the IPv6 ANY address.

The following provides some details involved in the handling of the configured listen blocks: There are interactions with (x)inetd. The `-p` and `-q` parameters sometimes play a role. And the RAD-Series Server has a default behavior if no `radius_socket{}` blocks are configured.

The following describes the opening of authentication listen sockets, and of how the RAD-Series Server builds the internal list of authentication listen sockets. Opening of accounting listen sockets is done in a similar manner:

- The RAD-Series Server first checks if it has been started via (x)inetd. If so, the server temporarily remembers the port number opened by (x)inetd.
- If (x)inetd has started the RAD-Series Server, and if the port number opened by (x)inetd matches the default authentication port (usually specified via the '-p' command-line parameter) then the socket opened by (x)inetd is considered an authentication socket and is added to the server's (initially empty) set of authentication listen sockets.
- The RAD-Series Server then processes the list of `ipaddr/authport pairs` configured in the `radius_socket{}` blocks. For each such `ipaddr/authport` pair, the server checks if the `authport` matches the port number, if any, opened by (x)inetd. If so, the server concludes that a socket for listening on this `ipaddr/authport` has already been opened by (x)inetd, and so the server skips over the opening of that `ipaddr/authport` configuration. Otherwise the server binds a new socket to the specified `ipaddr/authport`, and adds the new socket to the list of authentication listen sockets. If the `authport` is configured with a value of zero, then the default authentication port, specified by the '-p' command-line parameters, is substituted.

  A note on the above - If IPv6 is disabled and if the `radius_socket{}` block's ipaddr is an IPv6 address, then the `ipaddr/authport` for that `radius_socket{}` block is bypassed, and an appropriate logfile message is generated, indicating that the RAD-Series Server will not listen on the specified IPv6 interface.
- If, after processing the collection of zero or more `radius_socket{}` blocks, no authentication sockets have yet been opened, and if no `radius_socket{}` blocks (either accounting or authentication) have been configured, then the RAD-Series Server opens one authentication listen socket, using the IPv4 ANY address (0.0.0.0) as the local interface and the default authentication port (usually specified with '-p') as the local authport. This is done for backwards compatibility with previous versions of the RAD-Series Server, which did not support `radius_socket{}` blocks. The reason for first checking that no `radius_socket{}` blocks of any kind, with neither `authport` nor `acctport`, are configured, is that this allows the server to be configured via `radius_socket{}` blocks as an authentication only or accounting only server.
- It is the configurer's responsibility to ensure that there are no `ipaddr/authport` conflicts (i.e. overlaps) in the collection of configured `radius_socket{}` blocks. If the RAD-Series Server, when opening a new listen socket, encounters a bind error, because the address/port is already in use, the server will terminate with an appropriate logfile message. This could happen, say, if `radius_socket{ ipaddr 0.0.0.0 authport 1812 }` and `radius_socket{ ipaddr 192.168.3.25 authport 1812 }` were both configured.

A note on IP Address/Port conflicts - If one configures two `radius_socket{}` blocks with the same listen `authport`, one for the IPv4 ANY address (0.0.0.0) and one for an IPv6 specific address (e.g. 2001:abc::4), this theoretically creates no conflicts. However some operating systems treat this as a conflict and fail with a "bind error" when opening the second socket. Therefore the RAD-Series Server will bypass any IPv4 and IPv6 address configurations found in `radius_socket{}` blocks which have the same port as the port opened by (x)inetd.

Managing RADIUS Proxying Sockets

The RAD-Series Server can open up to eight sockets on which to proxy requests to other RADIUS servers.

By default, the RAD-Series Server will open one proxying socket, for proxying IPv4 requests to IPv4 servers; this socket is opened with the IPv4 ANY address and an ephemeral port. If IPv6 is enabled (see "ipv6_enabled" on page 175), then by default the server will open a second proxying socket, for proxying IPv6 requests to IPv6 servers; this socket is opened with the IPv6 ANY address and an ephemeral port. Binding to the local ANY address causes the operating system to select the local IP address for proxied requests.

By configuration, this default behavior can be modified. If the OS-chosen source IP addresses are not acceptable, then a default specific IPv4 source address for proxied IPv4 requests can be explicitly configured with the `default_source_ipv4_address` parameter (see "default-source-ipv4-address" on page 173), and a default specific IPv6 source address for proxied IPv6 requests can be configured with the `default_source_ipv6_address` parameter (see "default-source-ipv6-address" on page 173). An example of configuration of these two parameters, located in the aaa.config file:

```
# Proxy IPv4 requests with this source IP address
default_source_ipv4_address      192.168.2.2

# Proxy IPv6 requests with this source IP address
default_source_ipv6_address      2001:abc::4
```

The RAD-Series Server can further be configured to utilize multiple IPv4 proxy sockets and multiple IPv6 proxy sockets. A given client (actually a RADIUS peer server in this case) can be configured with a source IP address and a source port for proxied requests to that client (see SrcIP in "clients" on page 224 which describes the SrcIP=<ipaddr>:port parameter of the clients file), and this client-specific IP address will override the default proxy source IP address. An example of configuration of this parameter, located in the clients file:

```
# Proxy requests to abc.com with:
#  a source IP address of 192.168.1.2 and a source port of 2222

abc.com          aSecret          type=PROXY
      srcip=[192.168.1.2]:2222
```

Closing and Reopening Listen Sockets and Proxy Sockets upon a HUP

The RAD-Series Server, when processing a HUP signal, will close all RADIUS listen sockets other than the (x)inetd opened socket (if any), and close all proxy sockets, then reread the configurations, and then open new listen and proxy sockets representing the (possibly-updated) configuration. There may or may not have been configuration changes affecting the set of listen IP address/ports and proxy IP address/ports, and IPv6 operation may or may not have been changed from enabled to disabled (or vice versa).

IPv6 Operation

The RAD-Series Server can always send and receive RADIUS messages via IPv4 packets. The RAD-Series Server can also send and receive RADIUS messages via IPv6 packets, though by default it does not do so. The following steps indicate how the RAD-Series Server can be configured to send and receive IPv6 RADIUS messages.

- There are some very preliminary steps not described here, such as updating one's local DNS with IPv6 mappings, and updating the RAD-Series Server's `/etc/hosts` file with local IPv6 address(es).
- Interlink Networks provides a utility for assessing a machine's readiness for IPv6 UDP communications. One should first run this readiness tool, and repair any revealed IPv6 deficiencies. See “The IPv6 Readiness Tool” on page 270 for details on this utility.
- The RAD-Series Server should be configured with IPv6 enabled. By default, IPv6 is disabled. See “The “ipv6_enabled” configuration parameter” on page 270 for details on what this means. Configure the server's aaa.config file with a “`ipv6_enabled yes`” parameter.
- Configure any IPv6 sockets on which the RADIUS server needs to listen for requests. See “Managing RADIUS Listen Sockets” on page 265 and “RADIUS Listen Socket Properties” on page 199.
- If the default IPv6 proxying behavior (by default, the source IPv6 address for proxied IPv6 requests is chosen by the OS, and the source port is ephemeral) is not adequate, then configure the IPv6 proxy sockets as needed. See “Managing RADIUS Proxying Sockets” on page 267.
- Configure any IPv6 clients and servers in the clients file. See “clients” on page 224.
- Configure the RAD-Series Server to communicate with an LDAP server using TCP/IPv6, if desired. See “ProLDAP Authentication Type” on page 211.
- Start the RAD-Series Server. Check the server's logfile for any messages indicating a problem. Do a crude test of the server's configuration by having the radcheck utility send an IPv6 status request. See “radcheck” on page 90 for more details. The radcheck output should confirm that IPv6 is enabled. Do another crude test of the server's configuration by having the radpwtst utility send an IPv6 RADIUS request. See “radpwtst” on page 92 for more details.

The IPv6 Readiness Tool

Interlink Networks provides an "IPv6 Readiness Tool" utility for assessing a machine's readiness for IPv6 UDP communications. This utility is installed with the RAD-Series Server (`/opt/aaa/bin/ipv6check/ipv6check.sh`) and can be installed separately for testing. This tool checks:

- If the operating system is supported by the RAD-Series Server.
- If IPv6 support is enabled in the kernel.
- If certain utilities (ifconfig, route, netstat, ip, tcpdump) are IPv6-ready.
- If certain IPv6 utilities (ping6, tracepath6, traceroute6) are available.
- If the DNS infrastructure of the operating system is IPv6 ready.
- If a Link-Local address is configured, and if so, is pingable.
- If an IPv6 Link-Local route exists.
- If an IPv6 default route exist.
- If a specified IPv6 address pingable.
- If the readiness tool's test server can bind to the [::] address
- If non-Link-Local IPv6 addresses are configured, and if so, if the readiness tool's test server can bind to the non-Link-Local IPv6 addresses.

The "ipv6_enabled" configuration parameter

The RAD-Series Server maintains a boolean configuration parameter which enables or disables sending and receiving RADIUS messages over UDP/IPv6. This parameter is initialized so that IPv6 communications are by-default disabled.

This "`ipv6_enabled Yes/No`" parameter is configured in the `aaa.config` file. When the RAD-Series Server reads the `aaa.config` file at startup, or when processing a HUP signal, the parameter's setting, initially off, is updated.

If IPv6 is disabled (`ipv6_enabled=No`), the RAD-Series Server will utilize only IPv4 sockets and internally call IPv4 only operating system routines. If enabled (`ipv6_enabled=Yes`), the server will open IPv6 sockets and call IPv6 operating system routines when needed.

The state of `ipv6_enabled` can be changed when re-reading `aaa.config` during HUP processing. The effect is that communications with IPv6 only clients will be shut down or opened up. This permits a quick disabling of IPv6 messages without extensive configuration changes: `ipv6_enabled=No` acts as a master IPv6 shutdown switch. If `ipv6_enabled=No`, then any clients which have only IPv6 addresses will be ignored. Any listen sockets configured with IPv6 addresses will not be opened. At startup and following a HUP, logfile messages are generated when IPv6 configuration pieces are being ignored.

The specific effects of `ipv6_enabled` are these:

- No matter whether `ipv6_enabled` is Yes or No:
  - IP address fields in most data structures are of a size capable of holding either IPv4 or IPv6 addresses.
  - The RAD-Series Server can send and receive IPv6-specific RADIUS attributes, such as `NAP-IPv6-Address`. These IPv6 related attributes, defined in RFC 3162: "RADIUS and IPv6", are defined in the default dictionary.
  - For IPv4 communications the RAD-Series Server will open IPv4 only sockets.
  - Communications with the LDAP server are not under the control of the ipv6_enabled switch. So if the LDAP server name maps to an IPv6 address, or if the LDAP server is configured with an explicit IPv6 address, then the RAD-Series Server will send IPv6 messages to the LDAP server.
- If `ipv6_enabled` is `Yes`:
  - IPv6 networking is additionally enabled. The RAD-Series Server will open IPv6 sockets for receiving and sending IPv6 messages. The server can communicate with IPv6 clients.
  - DNS lookups are done with an operating system routine which returns both IPv4 and IPv6 addresses.
- If `ipv6_enabled` is `No`:
  - IPv6 networking is specifically disabled. The RAD-Series Server will not communicate with any peer which has only an IPv6 address.
  - DNS lookups are done with an operating system routine which returns only IPv4 addresses.

Configuring EAP-AKA

Configuration information for EAP-AKA is placed in several files, with some default values built into the RAD-Series Server. There is a precedence for the values that can be found in multiple places. The server starts with the built-in default values and overrides them with values in `aaa.config` (global values). The resulting values can be overridden by values in the `EAP.authfile` on a per realm basis. Finally the results of the user credential lookup may override some of these values.

Which file to configure a particular piece of information in is best made on the basis of where it will appear the least number of times. Based on this, the choice is first to use the `aaa.config` file, then `EAP.authfile`, and finally in the user credential datastore.

Some items would not be reasonable to configure on a global basis since they are user-specific, like the user password. These need to be unique, so the user credential datastore is the correct place for them. However some items, such as the `AKA-Algorithm`, may be common to all the users in a given realm so they could be configured in the `EAP.authfile` in just one place instead of each user's entry in the datastore for that realm. If all the realms used the same `AKA-Algorithm` then putting the `AKA-Algorithm` in the `aaa.config` file would be the best option. These are just some basic guidelines for grouping the common values in the best place.

EAP-AKA Features

The Interlink EAP-AKA RAD-Series Server is fully compliant with RFC4187, 2006. The RFC's "MUSTs" "SHOULDs", and "RECOMMENDEDs" are implemented. It supports the following features:

- IMSI permanent identities, supportable on a per realm basis
- Non-IMSI permanent identities, supportable on a per realm basis
- Protected success indications, supportable on a per realm basis
- Fast re-authentication, supportable on a per realm basis
- Protected Identity Exchanges using `AT_CHECKCODE`, supportable on a per realm basis
- Authentication Management Field (**`AMF`**) supportable on a per realm basis
- Pseudonym support via algorithmically generated or random pseudonyms, supportable on a per realm basis
- To ensure that permanent usernames, pseudonym usernames, and fast re-authentication usernames are separate and recognizable from each other, the RAD-Series Server generates pseudonyms with a leading "4" character and fast re-authentication usernames with a leading "5" character. Per the RFC, permanent usernames derived from the IMSI are prefixed with a leading "0" character.
- Many EAP-AKA parameters are configurable on a per realm basis
- A user's subscriber key (Ki), sequence number, mode, and the name of the appropriate AKA

algorithms, may be stored in an external database or local file. If so, the RAD-Series Server will automatically generate the authentication vector from this information.

- An authentication vector may be stored in a local file. This is intended for use in a lab environment, and requires no additional user-written plug-ins.
- The user credentials may be retrieved from an AuC if the customer implements an AATV which communicates with the AuC.
- AKA 3GPP Milenage algorithms are provided with configurable parameters.
- The Milenage AKA algorithm can be customized with a simple plug-in.
- Additional customer supplied AKA algorithms may be plugged into the RAD-Series Server.
- Occurrences and values of received AKA attributes are validated.
- Support for pseudonym and fast reauthentication identity mapping is built-in, without the need of an external database.

EAP-AKA User Credential Lookup Configuration

The RAD-Series Server inherently supports configuration of EAP-AKA user credentials as Reply Items in two forms:

One form is the configuration of the user's `Subscriber-Key` (Ki), `AKA-Sequence-Number` (SQN), `AKA-Mode` (Authentication Management Field, AMF), and `AKA-Algorithm`. See "A3, A8 and AKA Algorithms" on page 311 for a description of the algorithm. The RAD-Series Server uses these AVPs as input to generate an authentication vector.

- `Subscriber-Key` is a string attribute containing the binary encoded 128-bit user secret key often referred to as Ki. The encoding should be in network byte order (big-endian).
- `AKA-Sequence-Number` is a string attribute containing the binary encoded 48-bit user sequence number often referred to as SQN. The encoding should be in network byte order (big- endian).
- `AKA-Mode` is a string attribute containing the binary encoded 16-bit user authentication management field often referred to as AMF. The encoding should be in network byte order (big-endian).
- `AKA-Algorithm` is a string attribute indicating the name of the AKA algorithm to be applied in AKA vector generation. Most lines in the configuration files are limited to 1023 characters which places a limit on the length of this string. The value is case-sensitive.

The second form is the configuration of an AKA vector. An `AKA-Vector` is a fixed length binary string (octets) attribute which holds an EAP-AKA authentication vector. The attribute value is a 576-bit binary string (72 bytes) partitioned as follows:

- `RAND =` The first 128 bits (16 bytes) of value
- `XRES =` The next 64 bits (8 bytes) of value
- `CK =` The next 128 bits (16 bytes) of value

- `IK` = The next 128 bits (16 bytes) of value
- `AUTN` = The last 128 bits (16 bytes) of value

Configuration of an AKA-Vector is intended for a test lab environment, when the user's `Subscriber-Key, AKA-Sequence-Number, AKA-Mode`, and `AKA-Algorithm` are not known but AKA vectors have been retrieved from the device to be used in the testing.

The user credentials can be stored in any supported data repository: a local realm users file, a LDAP database, or a customer supplied database.

Additionally, a customer-specific plug-in may provide EAP-AKA user credentials. This plug-in may access a customer supplied database or may contact an authentication center. The plug-in's `Authentication-Type` needs to be defined in the `dictionary` file and may be configured with an `Xstring` parameter in the `authfile`.

Example realm file content

The following is an example of a local realm users file. See "User Entry Syntax" on page 228 for the a description of the format and allowable content. What follows are two EAP-AKA users with reply items used to authenticate the user:

```
# IMSI configured with 128 bit Subscriber-Key, 16 bit AKA-Mode,
#                       48 bit AKA-Sequence-Number, and AKA-Algorithm
801448005551000
  Subscriber-Key
="\x6d\x37\x71\x8a\xcc\xec\x37\x01\x4e\xdb\xf0\xf0\x3b\xe5\x77\xda",
  AKA-Mode = "\xa1\xb2",
  AKA-Sequence-Number = "\x01\x02\x03\x04\x05\x06",
  AKA-Algorithm = "3GPP-Milenage"

# Generic (i.e. non-IMSI permanent identity) user
# configured with a 576 bit AKA vector
fred
  AKA-Vector="\x11\x11\x11\x11\x11\x11\x11\x11\x11\x11\x11\x11\x11\x12
\x22\x22\x22\x22\x22\x22\x22\x22\x22\x22\x22\x22\x22\x33\x33\x33\x33\x
33\x33\x33\x33\x33\x33\x33\x33\x33\x34\x44\x44\x44\x44\x44\x44\x44\x44
\x44\x44\x44\x44\x44\x55\x55\x55\x55\x55\x55\x55\x55\x55\x55\x55\x55\x
55\x55\x55\x55\x55\x55"
```

Note: `Subscriber-Key, AKA-Mode, AKA-Sequence-Number` and `AKA-Vector` values are binary strings and are configured as quoted strings of hex escaped octets.

Note: If a user's `Subscriber-Key` is configured, but the `AKA-Algorithm` is not configured, the default `AKA-Algorithm` specified in the realm's configuration or the globally configured value will be used.

EAP-AKA Realm-Based Configurations

Many EAP-AKA parameters can be configured on a per realm basis. These parameters are configured in realm entries stored in the `authfile` and `EAP.authfile` (these files are possibly prefixed). The realm names may be wildcarded.

Realm-Based EAP-AKA Configuration Information in authfile

The user's AKA credentials lookup information is configured in the `authfile` on a realm by realm basis

The EAP-AKA realm must be configured with the `-AKA` switch (the `-AKA` switch is new as of version 7.4), which shields this realm entry from undesired `iaaaRealm` actions.

If the realm utilizes a local users file, then the `localFile` AATV will be used.

localFile Authentication Type

The `localFile` AATV is an enhanced version of `iaaaFile`. Whereas `iaaaFile` always looks up the user record specified by the `User-Id` attribute value, `localFile` can search based on a specified attribute value. The configuration of a realm which employs `localFile` is followed by a required `{}` block. The `{}` block allows any or all of these three parameters: `Request-Attribute-For-Search, Filter-Type`, and `Policy-Pointer`. Another difference from iaaaFile is that the `Filter-Type` is not controlled by the `-BIN` and `-CIS` flags but by the `Filter-Type` specification in the `{}` block.

Parameter	Description
Request-Attribute-For-Search	This indicates the search attribute to use for a user lookup. The attribute must be a string-type (string, tag-str, octets). The `localFile` AATV is not restricted to use by EAP-AKA, it may be used generically. When `localFile` is used for EAP-AKA, the `Request-Attribute-For-Search` attribute must be configured with a value of `Real-Username`. The default value, if not present, is `User-Id`.
Filter-Type	This is used to specify case-sensitive or case-insensitive treatment of the value of the `Request-Attribute-For-Search` attribute. A value of "`BIN`" causes a case-sensitive lookup. A value of "`CIS`" causes a case-insensitive lookup performed by taking the contents of the attribute specified for searching and converting it to uppercase before the datastore lookup. Therefore the local realm file needs to store the identity in uppercase. The default value, if `Filter-Type` is not present, is `BIN`.

Policy-Pointer	For information on `Policy-Pointer`, see “Authorization” on page 8.

Example localFile authfile configuration for credentials lookup

```
# This realm uses a local realm users file “realm1.users”
eapakarealm1.com     -AKA     localFile     realm1
{
    Request-Attribute-For-Search    Real-Username
}

# This set of wildcarded realms use realm users file “ispx.users”
*.ispx.com           -AKA     localFile                ispx
{
    Request-Attribute-For-Search    Real-Username
}
```

ProLDAP Authentication Type

The `ProLDAP` AATV has, as of version 7.3, been enhanced to support the `Request-Attribute-For-Search` configuration parameter.

Parameter	Description
Request-Attribute-For-Search	This indicates the search attribute to use for a user lookup. The attribute must be a string-type (string, tag-str, octets). The default value, if not present, is `User-Id`. When ProLDAP is used for EAP-AKA, the `Request-Attribute-For-Search` attribute must be configured with a value of `Real-Username`.

Example ProLDAP authfile configuration for credentials lookup

```
# This realm uses an LDAP database
eapakarealm3.com     -AKA     ProLDAP     “LDAP lookup”
{
      Request-Attribute-For-Search    Real-Username
      Filter-Type                     CIS
      Directory   “Directory 1”
      {
            URL              “ldap://ldap1.ispx.com:389”
            Administrator    “cn=...,ou=...,ou=...,o=radius”
            Password         “password”
            SearchBase       “...,ou=...,o=radius”
            Authenticate     Search
      }
}
```

See "Local Storage in Default users File

Local storage of user's credentials can be stored in a flat file on server wide basis. `users` is the file that is used to store these credentials. There is no entry required in the `authfile` to invoke this lookup. All authentications attempt to find the user's credentials in this file before using any other authentication type.

Note: The user's entry in the "`users`" file must include the full users' identity, including any realm name.

Local Storage – iaaaFile

Local storage of user's credentials can be stored in a flat file on a realm by realm basis. In the `authfile`, the `iaaaFile` authenication type is used to specify the realm and the flat file name in the *`auth-parameter`* field. For example:

```
fred.com    iaaaFile    fred.com
```

Specifies that realm fred.com users are to be looked up in the file, `fred.com.users`.

Note: The flat file name in the `authfile` does not include the required extension of "`.users`".

Unix Password

User's credentials can be stored in the Unix password file. In the `authfile`, the `PASSWD` authenication type is used to specify a realm whose users should be looked up in the Unix passwd file. The *`auth-parameter`* field is empty. For example:

```
fred.com    PASSWD      ""
```

Specifies that realm fred.com users are to be looked up in the Unix password file.

SecurID

User's credentials can be stored in RSA SecuID authentication server which does two factor authentication. In the `authfile`, the `SecurID` authenication type is used to specify a realm whose users should be authenticated using a SecurID ACE server. The *`auth-parameter`* field is empty. For example:

```
fred.com    SecurID     ""
```

Specifies that realm fred.com users are to be authenticated via the RSA SecurID ACE server. You will also need to export two files, `securid` and `sdconf.rec`, from the ACE server to allow secure communication between the RAD-Series AAA server and the SecurID ACE server. They need to be put in the configuration directory. There are other configuration parameters in the `aaa.config` file which affect the communications with the RSA SecurID ACE server.

ProLDAP Authentication Type" on page 210 for details on all the parameters.

Realm-Based EAP-AKA Configuration Information in EAP.authfile

The `EAP.authfile` entry for a realm which supports EAP-AKA can contain an optional {} configuration block following the '`EAP-Type AKA`' specification. This block contains realm specific EAP-AKA configuration information. Some of these configuration lines are EAP-AKA parameters and the AATVs for lookup and update of pseudonym and fast reauthentication identity mappings. See "Pseudonym and Fast Re-Authentication Data Base AATVs" on page 316 for a description of the provided implementation.

Some of the parameters, if not specified in the `EAP-Type AKA{}` configuration block, will be assigned default values from the `aatv.EAP-AKA{}` configuration block in `aaa.config`. Other parameters do not have a default and must be specified if the capability is to be supported.

The following rules apply to the `EAP-Type AKA{}` configuration block parameters

- The parameter names are case insensitive.
- For parameters with on/off binary values, the values "`enabled`", "`yes`", "`on`", and "`true`" are synonyms and the values "`disabled`", "`no`", "`off`", and "`false`" are synonyms.
- String parameter values should be enclosed in single or double quotes.
- When configuring a lookup or update for a fast reauthentication identity or pseudonym, the configuration parsing requires that a string parameter must be specified. It may be an empty string, i.e. "". This string is used as the `Xstring` value for the lookup and update AATV calls. The Interlink provided AATVs do not require an `Xstring` and so the "" should be used. If a custom AATV is written for lookup or update then that AATV may require a string parameter. See the Software Developers Kit for more details on AATVs and `Xstring`.
- If there is no AATV for the lookup or update for a fast reauthentication identity or pseudonym, then the parameter (e.g. `Fast-Reauth-Lookup`) may be simply not configured at all. Alternatively, the configuration may be explicitly configured with an AATV value of "`NULL`" or "`NONE`".
- Pseudonym lookup is disabled if the realm's configuration specifies that pseudonyms should be algorithmically generated (i.e. "`Generate-Random-Character-Pseudonyms No`"), but NO pseudonym encryption keys are configured in the `aatv.SIMAKA{}` section of the `aaa.config` file.
- Generation of new pseudonyms is disabled if the realm's configuration specifies that pseudonyms should be generated but NO `Pseudonym-Algorithm-Current-Key` is configured in the `aatv.SIMAKA{}` section of the `aaa.config` file.

The `EAP-Type AKA{}` configuration block can contain any subset, including the empty subset, of the following named parameters:

Parameter	Description
AKA-Algorithm	This parameter specifies the default AKA Algorithm for the realm. If the profile for a user in this realm does not specify an AKA Algorithm and if an AKA Algorithm is needed to produce the AKA vector for this user's authentication, then the AKA Algorithm specified by this parameter is used. See “A3, A8 and AKA Algorithms” on page 311 for details on available algorithms. If not explicitly configured, the default value is specified in the `AKA-Algorithm` parameter in the `aatv.EAP-AKA{}` section of the `aaa.config` file.
AKA-Mode	`AKA-Mode` is the user authentication management field often referred to as AMF. It is an input to the functions f1 and f1*, See 3GPP documents for details. The value is a 16-bit binary string (2 bytes) entered as “0x” followed by 2 two digit hex values. “dots” are optional and are just for readability. The encoding should be in network byte order (big-endian). See examples below. If not explicitly configured, the default value is specified in the `AKA-Mode` parameter in the `aatv.EAP-AKA{}` section of the `aaa.config` file.
Prefixed-IMSI-Permanent-IDs	This parameter indicates whether or not the RAD-Series Server should, for this realm, accept permanent identities of the form '0' + IMSI. The EAP-AKA RFC 4187 indicates that the permanent identity SHOULD be derived from the IMSI, but alternatively, an implementation MAY choose a permanent identity that is not based on the IMSI. The server supports both options. The valid values are "`Enabled`" and "`Disabled`". If not explicitly configured, the default value is specified in the `Prefixed-IMSI-Permanent-IDs` parameter in the `aatv.EAP-AKA{}` section of the `aaa.config` file.

Parameter	Description
Generic-Permanent-IDs	This parameter indicates whether or not the RAD-Series Server should, for this realm, accept generic permanent identities not based on an IMSI, e.g. "fred". The EAP-AKA RFC 4187 indicates that the permanent identity SHOULD be derived from the IMSI, but alternatively, an implementation MAY choose a permanent identity that is not based on the IMSI. The server supports both options. The valid values are "`Enabled`" and "`Disabled`". If not explicitly configured, the default value is specified in the `Generic-Permanent-IDs` parameter in the `aatv.EAP-AKA{}` section of the `aaa.config` file.
Minimum-Length-IMSI and Maximum-Length-IMSI	These parameters specify the minimum and maximum length of IMSIs that the RAD-Series Server will accept. The server performs sanity checks on a permanent identity that is offered as an IMSI to ensure that the identity is neither too short nor too long to be an IMSI. The EAP-AKA RFC 4187 explicitly states that 15 is the maximum. The minimum length is 6 based on a 3 digit MCC plus a 2 digit MNC plus a 1 digit MSIN. This is a theoretical absolute minimum length for an IMSI. Therefore the check made is: `6 <= Minimum-Length-IMSI <= Maximum-Length-IMSI <= 15` If not explicitly configured, the default values are specified in the `Minimum-Length-IMSI` and `Maximum-Length-IMSI` parameters in the `aatv.EAP-AKA{}` section of the `aaa.config` file.
Protected-Success-Indications	Protected success indications are an optional EAP-AKA feature which the Interlink EAP-AKA RAD-Series Server supports. This parameter indicates whether the server will offer protected success indications to the peer. The valid values are "`Enabled`" and "`Disabled`". If not explicitly configured, the default value is specified in the `Protected-Success-Indications` parameter in the `aatv.EAP-AKA{}` section of the `aaa.config` file.

Parameter	Description
Protected-Identity-Exchanges	The use of the `AT_CHECKCODE` attribute is an optional feature in EAP-AKA. This parameter determines if the RAD-Series Server should use the `AT_CHECKCODE` attribute. The attribute is used in order to allow protection of the EAP/AKA-Identity messages and any future extensions to them. The implementation of `AT_CHECKCODE` is RECOMMENDED. The valid values are "`Yes`" and "`No`". If not explicitly configured, the default value is specified in the `Protected-Identity-Exchanges` parameter in the `aatv.EAP-AKA{}` section of the `aaa.config` file.
Resync-Update	The EAP-AKA protocol requires support for two features related to the management of sequence numbers (SQN). This parameter specifies an AATV which provides one of the features and an `Xstring` parameter for this AATV. This AATV is invoked to inform the authentication center (AuC) about synchronization failures. The reception of an EAP-Response/AKA/Synchronization-Failure message from the client will trigger the call to this AATV. This parameter does not need to be configured if your implementation does not require this feature. There is no default.
Auth-Result-Update	The EAP-AKA protocol requires support for two features related to the management of sequence numbers (SQN). This parameter specifies an AATV which provides one of the features and an Xstring parameter for this AATV. This AATV is invoked to inform the authentication center (AuC) about the results of an authentication attempt. The completion of an EAP-AKA authentication sequence (successful or not) will trigger the call to this AATV. This parameter does not need to be configured if your implementation does not require this feature. There is no default.

Parameter	**Description**
Fast-Reauth-Lookup	Fast re-authentication is an optional EAP-AKA feature which the Interlink EAP-AKA RAD-Series Server supports. This parameter specifies an AATV and an `Xstring` parameter for this AATV. This AATV is invoked to map a fast reauthentication identity to the user's real identity and full authentication context. If this parameter is not configured, then fast re-authentication support is disabled for the realm. The Interlink server provides an AATV, `SIMAKA-ReauthCacheLookup`, for this function. See "Fast Re-Authentication" on page 318 for details. There is no default.
Fast-Reauth-Update	Fast re-authentication is an optional EAP-AKA feature which the Interlink EAP-AKA RAD-Series Server supports. This parameter specifies an AATV and an `Xstring` parameter for this AATV. This AATV is invoked to update the mapping of a fast reauthentication identity to a user's real identity. If this parameter is not configured, then fast re-authentication support is disabled for the realm. The Interlink server provides an AATV, `SIMAKA-ReauthCacheUpdate`, for this function. See "Fast Re-Authentication" on page 318 for details. There is no default.

Parameter	Description
Pseudonym-Lookup	Pseudonyms are an optional EAP-AKA feature which the Interlink EAP-AKA RAD-Series Server supports. This parameter specifies an AATV and an `Xstring` parameter for this AATV. This AATV is invoked to map a pseudonym to the user's real identity. If this parameter is not configured, then pseudonym support is disabled for the realm. If this parameter specifies the Interlink-provided AATV `SIMAKA-PseudonymDecrypt`, see "Pseudonym" on page 316, then: • The server forces non-random pseudonym generation for this realm. • If no `Pseudonym-Algorithm-Key-*` parameters are defined in the `aatv.SIMAKA{}` section of the `aaa.config` file, then pseudonym support is disabled. • If at least one of the above mentioned keys is defined and the `Pseudonym-Algorithm-Current-Key` is not defined in the `aatv.SIMAKA{}` section of the `aaa.config` file or does not refer to a defined key, then generation of new pseudonyms is disabled but existing pseudonyms can be looked up. There is no default.
Pseudonym-Update	Pseudonyms are an optional EAP-AKA feature which the Interlink EAP-AKA RAD-Series Server supports. This parameter specifies an AATV and an `Xstring` parameter for this AATV. This AATV is invoked to update the mapping of a pseudonym to a user's real identity. The Interlink provided pseudonym support using an algorithm, does not require an `Pseudonym-Update` AATV. See "Pseudonym" on page 316. There is no default.
Max-Number-Of-Reauths-Before-Full-Auth-Is-Required	Fast re-authentication is an optional EAP-AKA feature which the Interlink EAP-AKA RAD-Series Server supports. This parameter specifies an upper limit for the number of subsequent fast re-authentications allowed before a full authentication needs to be performed. The valid range is 1 to 65,535. If not explicitly configured, the default value is specified in the `Max-Number-Of-Reauths-Before-Full-Auth-Is-Required` parameter in the `aatv.EAP-AKA{}` section of the `aaa.config` file.

Parameter	**Description**
Fast-Reauth-Realm	When providing a fast reauthentication identity, the RAD-Series Server also includes a realm to help ensure that the subsequent fast re-authentication be targeted to this server, the only server which holds the full authentication context if internal caching, rather than an external database, is used to save the fast re-authentication context. This parameter specifies such a realm. Since the maximum length of a fast reauth NAI cannot exceed 253 characters and since the length of the fast reauth user name is 10 characters, the Fast Reauth Realm value must not exceed 242 characters. If the fast reauthentication identity should be generated with NO realm name then this would be configured as "`NULL`". The empty string entry, using just two quotes, indicates that the server should generate a fast reauthentication identity with the same realm name as the permanent identity. If not explicitly configured, the default value is specified in the `Fast-Reauth-Realm` parameter in the `aatv.EAP-AKA{}` section of the `aaa.config` file.
Fast-Reauth-Id-Lifetime	Fast re-authentication is a mechanism for a AKA user to freshen his keys periodically. A fast re-authentication, if it is going to take place, will happen in a "short" time after a full authentication or a previous fast re-authentication. This parameter specifies a lifetime for a fast reauthentication identity, in seconds. If the fast reauthentication identity is assigned and isn't used within this period of time, the fast reauthentication identity and the associated full authentication context is purged. The valid range is 1 to 14400 (1 second to 4 hours). If not explicitly configured, the default value is specified in the `Fast-Reauth-Id-Lifetime` parameter in the `aatv.EAP-AKA{}` section of the `aaa.config` file.
Pseudonym-Lifetime	A random character pseudonym, when generated for a user, is placed in an external database. This parameter specifies the lifetime of such a generated random character pseudonym. After the specified length of time has elapsed since the pseudonym was first assigned, the pseudonym is invalidated, independent of how many times the pseudonym was used, if ever. The valid range is 1 to 31,622,400 (1 second to 366 days). If not explicitly configured, the default value is specified in the `Pseudonym-Lifetime` parameter in the `aatv.EAP-AKA{}` section of the `aaa.config` file.

Parameter	Description
Generate-Random-Character-Pseudonyms	The Interlink RAD-Series Server provides a mechanism, using configured encryption keys, by which pseudonyms can be generated as an encrypted form of the permanent identity, which can be subsequently decrypted to reproduce the permanent identity. Alternatively, the server can generate pseudonyms as a string of random characters, similar to the fast reauthentication identity. In this latter case, an external database is required to store the pseudonym to permanent identity mappings. This parameter indicates whether the server generates pseudonyms by algorithm (value=no) or if the server generates random character pseudonyms (value=yes). The valid values are "`Yes`" and "`No`". If not explicitly configured, the default value is specified in the `Generate-Random-Character-Pseudonyms` parameter in the `aatv.EAP-AKA{}` section of the `aaa.config` file.

Example EAP.authfile configuration file

The following `EAP.authfile` configures the EAP-AKA protocol for an AKA realm:

```
eapaka.com    -EAP   EAP     "comment"
{
  EAP-Type AKA
  {
    AKA-Algorithm                               "3GPP-Milenage"
    Prefixed-IMSI-Permanent-IDs                 "Enabled"
    Generic-Permanent-IDs                       "Enabled"
    Minimum-Length-IMSI                         6
    Maximum-Length-IMSI                         15
    Protected-Success-Indications               "Disabled"
    Protected-Identity-Exchanges                No
    AKA-Mode                                    0x12ab
    Resync-Update            null               "null"
    Auth-Result-Update       NULL               "none"

    #            Temporary identity datastores
    Fast-Reauth-Lookup    SIMAKA-ReauthCacheLookup   ""
    Fast-Reauth-Update    SIMAKA-ReauthCacheUpdate   ""
    Pseudonym-Lookup      SIMAKA-PseudonymDecrypt    ""
    Pseudonym-Update      NULL                       ""

    #            Fast Reauth configuration:
    Max-Number-Of-Reauths-Before-Full-Auth-Is-Required  2
    Fast-Reauth-Realm                           "this.server.com"
    Fast-Reauth-Id-Lifetime                     3600

    #            Pseudonym configuration:
    Pseudonym-Lifetime                          1209600
    Generate-Random-Character-Pseudonyms    "No"
  }
}
```

Global EAP-AKA Configuration in aaa.config

The `aatv.EAP-AKA{}` configuration block, located within the `aaa.config` file, contains global EAP-AKA configuration information. Some of the parameters represent realm default values for those not specified in the realm configuration. Other parameters represent global defaults which do not correspond to any realm based parameter. For the global parameters common to EAP-SIM and EAP-AKA, see "EAP-AKA and EAP-SIM Common Global Configurations" on page 291.

The following rules apply to the `aatv.EAP-AKA{}` configuration block parameters

- The parameter names are case insensitive.
- For parameters with on/off binary values, the values "`enabled`", "`yes`", "`on`", and "`true`" are synonyms and the values "`disabled`", "`no`", "`off`", and "`false`" are synonyms.
- String parameter values should be enclosed in single or double quotes.

The `aatv.EAP-AKA{}` configuration block, in `aaa.config` file, can contain any subset, including the empty subset, of the following named parameters:

The following parameters are global. No realm configuration overrides.

Parameter	Description
Statistics	This parameter enables/disables the output of EAP-AKA statistics to the logfile when the RAD-Series Server shuts down. The valid values are "`Enabled`" and "`Disabled`". If not explicitly configured, the default value is "`Enabled`".

The following parameters specify RAD-Series Server-wide realm defaults.
These are overridable by the realm configuration.

Parameter	Description
AKA-Algorithm	This parameter specifies the global default AKA Algorithm. If the profile for a user does not specify an AKA Algorithm and if the realm configuration does not specify an AKA Algorithm and if an AKA Algorithm is needed to produce the AKA Vector for this user's authentication, then the AKA Algorithm specified by this parameter is used. See “A3, A8 and AKA Algorithms” on page 311 for details on available algorithms. If not explicitly configured, there is NO default value.
AKA-Mode	`AKA-Mode` is the user authentication management field often referred to as AMF. It is an input to the functions f1 and f1*, See 3GPP documents for details. The value is a 16-bit binary string (2 bytes) entered as “`0x`” followed by 2 two digit hex values. “dots” are optional and are just for readability. The encoding should be in network byte order (big-endian). See examples below. If not explicitly configured, there is NO default value.
Prefixed-IMSI-Permanent-IDs	This parameter indicates whether or not the RAD-Series Server should accept permanent identities of the form '0' + IMSI for a realm, if the realm configuration does not specify this parameter. The EAP-AKA RFC 4187 indicates that the permanent identity SHOULD be derived from the IMSI, but alternatively, an implementation MAY choose a permanent identity that is not based on the IMSI. The valid values are "`Enabled`" and "`Disabled`". If not explicitly configured, the default value is "`Enabled`".
Generic-Permanent-IDs	This parameter indicates whether or not the RAD-Series Server should accept generic permanent identities not based on an IMSI, e.g. “fred”, for a realm where the realm configuration does not specify this parameter. The EAP-AKA RFC 4187 indicates that the permanent identity SHOULD be derived from the IMSI, but alternatively, an implementation MAY choose a permanent identity that is not based on the IMSI. The valid values are "`Enabled`" and "`Disabled`". If not explicitly configured, the default value is "`Disabled`".

Parameter	Description
Minimum-Length-IMSI and Maximum-Length-IMSI	These parameters specify the minimum and maximum length of IMSIs that the RAD-Series Server will accept. The server performs sanity checks on a permanent identity that is offered as an IMSI to ensure that the identity is neither too short nor too long to be an IMSI. The EAP-AKA RFC 4187 explicitly states that 15 is the maximum. The minimum length is 6, based on a 3 digit MCC plus a 2 digit MNC plus a 1 digit MSIN. This is a theoretical absolute minimum length for an IMSI. Therefore the check made is: `6 <= Minimum-Length-IMSI <= Maximum-Length-IMSI <= 15` If not explicitly configured, the default values are 6 and 15.
Protected-Success-Indications	Protected success indications are an optional EAP-AKA feature which the Interlink EAP-AKA RAD-Series Server supports. This parameter indicates whether the server will offer protected success indications to the peer. The valid values are "`Enabled`" and "`Disabled`". If not explicitly configured, the default value is "`Enabled`".
Protected-Identity-Exchanges	The use of the `AT_CHECKCODE` attribute is an optional feature in EAP-AKA. This parameter determines if the RAD-Series Server should use the `AT_CHECKCODE` attribute. The attribute is used in order to allow protection of the EAP/AKA-Identity messages and any future extensions to them. The implementation of `AT_CHECKCODE` is RECOMMENDED. The valid values are "`Ye`s" and "`No`". If not explicitly configured, the default value is "`Yes`".
Max-Number-Of-Reauths-Before-Full-Auth-Is-Required	Fast re-authentication is an optional EAP-AKA feature which the Interlink EAP-AKA RAD-Series Server supports. This parameter specifies an upper limit for the number of subsequent fast re-authentications allowed before a full authentication needs to be performed. The valid range is 1 to 65,535. If not explicitly configured, the default value is 4.

Parameter	Description
Fast-Reauth-Realm	When providing a fast reauthentication identity, the RAD-Series Server also includes a realm to ensure that the subsequent fast re-authentication be targeted to this server, the only server which holds the full authentication context if internal caching, rather than an external database, is used to save the fast re-authentication context. This parameter specifies such a realm. Since the maximum length of a fast reauth NAI cannot exceed 253 characters and since the length of the fast reauthentication identity is 10 characters, the Fast Reauth Realm value must not exceed 242 characters. If the fast reauthentication identity should be generated with NO realm name then this would be configured as "`NULL`". The empty string entry, using just two quotes, indicates that the server should generate a fast reauthentication identity with the same realm name as the permanent identity. If not explicitly configured, the default value is the empty string entry which indicates the server should generate a fast reauthentication identity with the same realm name as the permanent identity.
Fast-Reauth-Id-Lifetime	Fast re-authentication is a mechanism for an EAP-AKA user to freshen his keys periodically. A fast re-authentication, if it is going to take place, will happen in a "short" time after a full authentication or a previous fast re-authentication. This parameter specifies a lifetime for a fast reauthentication identity, in seconds. If the fast reauthentication identity is assigned and isn't used for this period of time, the fast reauthentication identity and the associated full auth context is purged. The valid range is 1 to 14400 (1 second to 4 hours). If not explicitly configured, the default value is 3600 (one hour),
Generate-Random-Character-Pseudonyms	Pseudonyms can be generated as an encrypted form of the permanent identity, which can be subsequently decrypted to reproduce the permanent identity. Alternatively, pseudonyms can be generated as a string of random characters, similar to the fast reauthentication identity. In this case, an external database is required to store the pseudonym to permanent identity mappings. This parameter indicates whether the RAD-Series Server generates pseudonyms by algorithm (value = "no") or if the server generates random character pseudonyms (value = "yes"). The valid values are "`Ye`s" and "`No`". If not explicitly configured, the default value is "`No`".

Parameter	Description
Pseudonym-Lifetime	A random character pseudonym, when generated for a user, is placed in an external database. This parameter specifies the lifetime of such a generated random character pseudonym. After the specified length of time has elapsed since the pseudonym was first assigned, the pseudonym is invalidated, independent of how may times the pseudonym was used, if ever used. The valid range is 1 to 31,622,400 (1 second to 366 days). If not explicitly configured, the default value is 1,209,600 (14 days).

Example aaa.config configuration file

```
aatv.EAP-AKA
{
  #   ======================================================================
  #   The following parameters are global. No realm configuration overrides.
  #   ======================================================================

  Statistics                              "Enabled"  # Enabled or Disabled

  #   ======================================================================
  #   All parameters that follow specify server-wide realm defaults.
  #   These are overridable by the realm configuration.
  #   ======================================================================

  Protected-Success-Indications           "Enabled"  # Enabled or Disabled
  Protected-Identity-Exchanges            "Yes"      # Yes or No
  Prefixed-IMSI-Permanent-IDs             "Enabled"  # Enabled or Disabled
  Generic-Permanent-IDs                   "Disabled" # Enabled or Disabled

  Minimum-Length-IMSI 6  # 6<=Minimum-Length-IMSI <= Maximum-Length-IMSI <=15
  Maximum-Length-IMSI 15 # 6<=Minimum-Length-IMSI <= Maximum-Length-IMSI <=15

  AKA-Algorithm                           "3GPP-Milenage"

  #   ======================================================================
  #   The following group of parameters configure the PSEUDONYM support
  #   ======================================================================

  Generate-Random-Character-Pseudonyms    "No"       # yes or no
  Pseudonym-Lifetime                      1209600    # 14 days, in seconds

  #   ======================================================================
  #   The following group of parameters configure FAST RE-AUTHENTICATION
  #   ======================================================================

  Max-Number-Of-Reauths-Before-Full-Auth-Is-Required  4  # range [1 to 65535]

  Fast-Reauth-Realm                       ""         # use perm ID's realm
  Fast-Reauth-Id-Lifetime                     3600       # range [1 to 14400]
}
```

EAP-AKA and EAP-SIM Common Global Configurations

Some parameters common to EAP-AKA and EAP-SIM can ONLY be configured on a global basis. These parameters are configured in the `aatv.SIMAKA{}` configuration block, located within the `aaa.config` file. These parameters represent global defaults which do not correspond to any realm based parameter.

The following rules apply to the `aatv.SIMAKA{}` configuration block parameters

- The parameter names are case insensitive.
- For parameters with on/off binary values, the values "`enabled`", "`yes`", "`on`", and "`true`" are synonyms and the values "`disabled`", "`no`", "`off`", and "`false`" are synonyms.
- String parameter values should be enclosed in single or double quotes.

The `aatv.SIMAKA{}` configuration block, in `aaa.config` file, can contain any subset, including the empty subset, of the following named parameters:

The following parameters are global. There are no realm based configuration overrides.

Parameter	Description
Maximum-Fast-Reauth-Cache-Size	This parameter specifies the maximum size of the in-memory Fast Re-authentication table, in terms of the number of entries. For a given user, the RAD-Series Server needs to save the full authentication context for subsequent fast re-authentications. A bound must be placed on the number of entries in this table to protect the server's memory. The valid range is 0 to 1,000,000. A value of zero means that no new fast reauth identities should be added to the cache, but existing non-expired entries will be honored. This value is intended to phase out fast reauth support following a HUP. If not explicitly configured, the default value is the number of licensed users up to a maximum of 1,000,000.
Pseudonym-Algorithm-Key-*n*	Pseudonyms can be generated as an encrypted form of the permanent identity, which can be subsequently decrypted to reproduce the permanent identity. This set of parameters (n = 1 to 16) can used to specify up to 16 encryption keys that can be used for this encryption/decryption.

	The key value is a 128-bit binary string (16 bytes) entered as “0x” followed by 16 two digit hex values. “dots” are optional and are just for readability. See examples below. Pseudonym generation will be disabled for a realm if no keys have been defined and the generation of random character pseudonyms is disabled (`Generate-Random-Character-Pseudonyms "no"`). If not explicitly configured, there are NO default values.
Pseudonym-Algorithm-Current-Key	The key number specified by this parameter is used to select the `Pseudonym-Algorithm-Key` to use to encrypt the permanent identity when generating a new pseudonym. The other keys are used for decryption of pseudonyms previously generated with the other keys, but are not used for generation of new pseudonyms. The valid range is 1 to 16. If not explicitly configured, there is NO default value.
Statistics	This parameter enables/disables the output of common SIM and AKA statistics (`SIMAKA statistics:`) to the logfile when the RAD-Series Server shuts down. The valid values are "`Enabled`" and "`Disable`d". If not explicitly configured, the default value is "`Enabled`".

Example aaa.config configuration file

```
aatv.SIMAKA
{
  #   ===================================================================
  #   The following parameters are global.
  #   They apply to the shared features of EAP-SIM and EAP-AKA.
  #   There are no realm configuration overrides.
  #   ===================================================================

  Maximum-Fast-Reauth-Cache-Size   4096       # range [0 to 1,000,000]

  Pseudonym-Algorithm-Key-1    0x00010203.04050607.08090a0b.0c0d0e0f
  Pseudonym-Algorithm-Key-11   0xa0a1a2a3.a4a5a6a7.a8a9aaab.acadaeaf
  Pseudonym-Algorithm-Key-16   0xf0f1f2f3.f4f5f6f7.f8f9fafb.fcfdfeff

  Pseudonym-Algorithm-Current-Key  11

  Statistics                       "Enabled"  # Enabled or Disabled
}
```

Configuring EAP-SIM

Configuration information for EAP-SIM is placed in several files, with some default values built into the RAD-Series Server. There is a precedence for the values that can be found in multiple places. The server starts with the built-in default values and overrides them with values in `aaa.config` (global values). The resulting values can be overridden by values in the `EAP.authfile` on a per realm basis. Finally the results of the user credential lookup may override some of these values.

Which file to configure a particular piece of information in is best made on the basis of where it will appear the least number of times. Based on this, the choice is first to use the `aaa.config` file, then `EAP.authfile`, and finally in the user credential datastore.

Some items would not be reasonable to configure on a global basis since they are user-specific, like the user password. These need to be unique, so the user credential datastore is the correct place for them. However some items, such as the `A8-Algorithm`, may be common to all the users in a given realm so they could be configured in the `EAP.authfile` in just one place instead of each user's entry in the datastore for that realm. If all the realms used the same `A8-Algorithm` then putting the `A8-Algorithm` in the `aaa.config` file would be the best option. These are just some basic guidelines for grouping the common values in the best place.

EAP-SIM Features

The Interlink EAP-SIM RAD-Series Server is fully compliant with RFC4186, 2006. The RFC's "MUSTs" "SHOULDs", and "RECOMMENDEDs" are implemented. It supports the following features:

- IMSI permanent identities, supportable on a per realm basis
- Non-IMSI permanent identities, supportable on a per realm basis
- Protected success indications, supportable on a per realm basis
- Fast re-authentication, supportable on a per realm basis
- Pseudonym support via algorithmically generated or random pseudonyms, supportable on a per realm basis
- To ensure that permanent usernames, pseudonym usernames, and fast re-authentication usernames are separate and recognizable from each other, the RAD-Series Server generates pseudonyms with a leading "2" character and fast re-authentication usernames with a leading "3" character. Per the RFC, permanent usernames derived from the IMSI are prefixed with a leading "1" character.
- Many EAP-SIM parameters are configurable on a per realm basis
- A user's subscriber key (Ki), along with the names of the appropriate A3 and A8 algorithms, may be stored in an external database or local file. If so, the RAD-Series Server will automatically generate GSM authentication triplets from this information.

- A set of GSM authentication triplets may be stored in a local file. This is intended for use in a lab environment, and requires no additional user-written plug-ins.
- The user credentials may be retrieved from an AuC if the customer implements an AATV which communicates with the AuC.
- A3/A8 3GPP Milenage algorithms are provided with configurable parameters.
- The Milenage A3/A8 algorithm can be customized with a simple plug-in.
- Additional customer supplied A3/A8 algorithms may be plugged into the RAD-Series Server.
- Occurrences and values of received SIM attributes are validated.
- Support for pseudonym and fast reauthentication identity mapping is built-in, without the need of an external database.

EAP-SIM User Credential Lookup Configuration

The RAD-Series Server inherently supports configuration of EAP-SIM user credentials as Reply Items in two forms:

One form is the configuration of the user's `Subscriber-Key` (Ki), `A3-Algorithm` and `A8-Algorithm`. See "A3, A8 and AKA Algorithms" on page 311 for a description. The RAD- Series Server uses these AVPs as input to generate authentication vectors.

- `Subscriber-Key` is a string attribute containing the binary encoded 128-bit user secret key often referred to as Ki. The encoding should be in network byte order (big-endian).
- `A3-Algorithm` is a string attribute indicating the name of the A3 algorithm to be applied in GSM triplet generation. Most lines in the configuration files are limited to 1023 characters which places a limit on the length of this string. The value is case-sensitive.
- `A8-Algorithm` is a string attribute indicating the name of the A8 algorithm to be applied in GSM triplet generation. Most lines in the configuration files are limited to 1023 characters which places a limit on the length of this string. The value is case-sensitive.

The second form is the configuration of a collection of GSM triplets. A `GSM-Triplet` is a fixed length binary string (octets) attribute which holds an EAP-SIM authentication vector. The attribute value is a 224-bit binary string (28 bytes) partitioned as follows:

- `RAND` = The first 128 bits (16 bytes) of value
- `Kc` = The next 64 bits (8 bytes) of value
- `SRES` = The last 32 bits (4 bytes) of value

Configuration of GSM triplets is intended for a test lab environment, when the user's `Subscriber-Key, A3-Algorithm`, and `A8-Algorithm` are not known but GSM triplets have been retrieved from the device to be used in the testing.

The user credentials can be stored in any supported data repository: a local realm users file, an LDAP database, or a customer supplied database.

Additionally, a customer-specific plug-in may provide EAP-SIM user credentials. This plug-in may access a customer supplied database or may contact an authentication center. The plug-in's `Authentication-Type` needs to be defined in the `dictionary` file and may be configured with an `Xstring` parameter in the `authfile`.

Example realm file content

The following is an example of a local realm users file. See “User Entry Syntax” on page 228 for the a description of the format allowable content. What follows are two EAP-SIM users with reply items used to authenticate the user:

```
# IMSI configured with 128 bit Subscriber-Key, A3-Algorithm, and A8-Algorithm
801448005551000
  Subscriber-Key
="\x6d\x37\x71\x8a\xcc\xec\x37\x01\x4e\xdb\xf0\xf0\x3b\xe5\x77\xda",
  A3-Algorithm = "3GPP-Milenage",
  A8-Algorithm = "3GPP-Milenage"

# Generic (i.e. non-IMSI permanent identity) user
# configured with a collection of 224 bit triplets
fred
  GSM-Triplet="\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1
a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a",
  GSM-Triplet="\x2b\x2b\x2b\x2b\x2b\x2b\x2b\x2b\x2b\x2b\x2b\x2b\x2b\x2
b\x2b\x2b\x2b\x2b\x2b\x2b\x2b\x2b\x2b\x2b\x2b\x2b\x2b\x2b",
  GSM-Triplet="\x3c\x3c\x3c\x3c\x3c\x3c\x3c\x3c\x3c\x3c\x3c\x3c\x3c\x3
c\x3c\x3c\x3c\x3c\x3c\x3c\x3c\x3c\x3c\x3c\x3c\x3c\x3c\x3c",
  GSM-Triplet="\x4c\x4c\x4c\x4c\x4c\x4c\x4c\x4c\x4c\x4c\x4c\x4c\x4c\x4
c\x4c\x4c\x4c\x4c\x4c\x4c\x4c\x4c\x4c\x4c\x4c\x4c\x4c\x4c",
  GSM-Triplet="\x5c\x5c\x5c\x5c\x5c\x5c\x5c\x5c\x5c\x5c\x5c\x5c\x5c\x5
c\x5c\x5c\x5c\x5c\x5c\x5c\x5c\x5c\x5c\x5c\x5c\x5c\x5c\x5c",
  GSM-Triplet="\x6c\x6c\x6c\x6c\x6c\x6c\x6c\x6c\x6c\x6c\x6c\x6c\x6c\x6
c\x6c\x6c\x6c\x6c\x6c\x6c\x6c\x6c\x6c\x6c\x6c\x6c\x6c\x6c",
  GSM-Triplet="\x7c\x7c\x7c\x7c\x7c\x7c\x7c\x7c\x7c\x7c\x7c\x7c\x7c\x7
c\x7c\x7c\x7c\x7c\x7c\x7c\x7c\x7c\x7c\x7c\x7c\x7c\x7c\x7c",
  GSM-Triplet="\x8c\x8c\x8c\x8c\x8c\x8c\x8c\x8c\x8c\x8c\x8c\x8c\x8c\x8
c\x8c\x8c\x8c\x8c\x8c\x8c\x8c\x8c\x8c\x8c\x8c\x8c\x8c\x8c",
  GSM-Triplet="\x9c\x9c\x9c\x9c\x9c\x9c\x9c\x9c\x9c\x9c\x9c\x9c\x9c\x9
c\x9c\x9c\x9c\x9c\x9c\x9c\x9c\x9c\x9c\x9c\x9c\x9c\x9c\x9c"
```

Note: `Subscriber-Key` and `GSM-Triplet` values are binary strings and are configured as quoted strings of hex escaped octets.

Note: If a user's `Subscriber-Key` is configured, but the `A3-Algorithm` and/or `A8-Algorithm` is not configured, the default `A3-Algorithm` and/or `A8-Algorithm` specified in the realm's configuration or the globally configured value will be used.

Note: Under some conditions more GSM triplets may be configured than are necessary for a user's authentication and the RAD-Series Server will select the necessary number of GSM triplets from the collection in a manner described in “Realm-Based EAP-SIM Configuration Information in authfile” on page 296.

EAP-SIM Realm-Based Configurations

Many EAP-SIM parameters can be configured on a per realm basis. These parameters are configured in realm entries stored in the `authfile` and `EAP.authfile` (these files are possibly prefixed). The realm names may be wildcarded.

Realm-Based EAP-SIM Configuration Information in authfile

The user's SIM credentials lookup information is configured in the `authfile` on a realm by realm basis

The EAP-SIM realm must be configured with the `-SIM` switch (the `-SIM` switch is new as of version 7.3), which shields the realm from undesired `iaaaRealm` actions.

If the realm utilizes a local users file, then one of the two new (as of version 7.3) AATVs should be used: `localFile` or `SIM-TripletFile`.

localFile Authentication Type

The `localFile` AATV is an enhanced version of `iaaaFile`. Whereas `iaaaFile` always looks up the user record specified by the `User-Id` attribute value, `localFile` can search based on a specified attribute value. The configuration of a realm which employs `localFile` is followed by a required `{}` block. The `{}` block allows any or all of these three parameters: `Request-Attribute-For-Search`, `Filter-Type`, and `Policy-Pointer`.

Parameter	**Description**
Request-Attribute-For-Search	This indicates the search attribute to use for a user lookup. The attribute must be a string-type (string, tag-str, octets). The `localFile` AATV is not restricted to use by EAP-SIM, it may be used generically. When `localFile` is used for EAP-SIM, the `Request-Attribute-For-Search` attribute must be configured with a value of `Real-Username`. The default value, if not present, is `User-Id`.
Filter-Type	This is used to specify case-sensitive or case-insensitive treatment of the value of the `Request-Attribute-For-Search` attribute. A value of "`BIN`" causes a case-sensitive lookup. A value of "`CIS`" causes a case-insensitive lookup performed by taking the contents of the attribute specified for searching and converting it to uppercase before the datastore lookup. Therefore the local realm file needs to store the identity in uppercase. The default value, if `Filter-Type` is not present, is `BIN`.
Policy-Pointer	For information on `Policy-Pointer`, see "Authorization" on page 8.

When `localFile` retrieves GSM triplets, there may be more configured triplets than are necessary for the user authentication. The EAP-SIM AATV will use the first *N* retrieved triplets, where *N* is the number of triplets this realm requires for a EAP-SIM authentication (i.e. either 2 or 3).

Example localFile authfile configuration for credentials lookup

```
# This realm uses a local realm users file "realm1.users"
eapsimrealm1.com     -SIM     localFile     realm1
{
    Request-Attribute-For-Search    Real-Username
}

# This set of wildcarded realms use realm users file "ispx.users"
*.ispx.com           -SIM     localFile     ispx
{
    Request-Attribute-For-Search    Real-Username
}
```

SIM-TripletFile Authentication Type

The `SIM-TripletFile` AATV is very similar to `localFile` with an identical {} block, but is specialized for the retrieval of `GSM-Triplet` credentials.

Parameter	Description
Request-Attribute-For-Search	This indicates the search attribute to use for a user lookup. The attribute must be a string-type (string, tag-str, octets). The default value, if not present, is `Real-Username`.
Filter-Type	This is used to specify case-sensitive or case-insensitive treatment of the value of the `Request-Attribute-For-Search` attribute. A value of "`BIN`" causes a case-sensitive lookup. A value of "`CIS`" causes a case-insensitive lookup performed by taking the contents of the attribute specified for searching and converting it to uppercase before the datastore lookup. Therefore the local realm file needs to store the identity in uppercase. The default value, if `Filter-Type` is not present, is `BIN`.
Policy-Pointer	For information on `Policy-Pointer`, see "Authorization" on page 8.

If `SIM-TripletFile` retrieves `GSM-Triplets` and if there are more configured triplets than are necessary for the user authentication, then `SIM-TripletFile` will return N configured triplets, where N is the number of triplets this realm requires for a SIM authentication. Rather than returning the first N triplets, `SIM-TripletFile` will pick a random starting point in the list of configured triplets and return the next N triplets, starting at the random starting point and wrapping to the beginning of the list if necessary.

If the datastore will be returning `GSM-Triplets`, then the use `SIM-TripletFile` would be the first choice. If you need a predicable set of `GSM-Triplets`, then use `localFile`.

Example SIM-TripletFile authfile configuration for credentials lookup

```
# This realm uses a local realm users file "realm2.users"
eapsimrealm2.com   -SIM   SIM-TripletFile   realm2
{
    Filter-Type                         CIS
    Request-Attribute-For-Search        Real-Username
    Policy-Pointer                      "decisionfile://policy-1.policy"
}
```

ProLDAP Authentication Type

The `ProLDAP` AATV has, as of version 7.3, been enhanced to support the `Request-Attribute-For-Search` configuration parameter.

Parameter	Description

Request-Attribute-For-Search	This indicates the search attribute to use for a user lookup. The attribute must be a string-type (string, tag-str, octets). The default value, if not present, is User-Id. When ProLDAP is used for EAP-SIM, the `Request-Attribute-For-Search` attribute must be configured with a value of `Real-Username`.

Example ProLDAP authfile configuration for credentials lookup

```
# This realm uses an LDAP database
eapsimrealm3.com            -SIM    ProLDAP    "LDAP lookup"
{
     Request-Attribute-For-Search   Real-Username
     Filter-Type                    CIS
     Directory   "Directory 1"
     {
          URL              "ldap://ldap1.ispx.com:389
          Administrator    "cn=...,ou=...,ou=...,o=radius"
          Password         "password"
          SearchBase       "...,ou=...,o=radius"
          Authenticate     Search
     }
}
```

See "ProLDAP Authentication Type" on page 276 for details on all the parameters.

Realm-Based EAP-SIM Configuration Information in EAP.authfile

The `EAP.authfile` entry for a realm which supports EAP-SIM can contain an optional {} configuration block following the '`EAP-Type SIM`' specification. This block contains realm specific EAP-SIM configuration information: EAP-SIM parameters and the AATVs for lookup and update of pseudonym and fast reauthentication identity mappings. See "Pseudonym and Fast Re-Authentication Data Base AATVs" on page 316 for a description of the provided implementation.

Some of the parameters, if not specified in the `EAP-Type SIM{}` configuration block, will be assigned default values from the `aatv.EAP-SIM{}` configuration block in `aaa.config`. Other parameters do not have a default and must be specified if the capability is to be supported.

The following rules apply to the `EAP-Type SIM{}` configuration block parameters

- The parameter names are case insensitive.
- For parameters with on/off binary values, the values "`enabled`", "`yes`", "`on`", and "`true`" are synonyms and the values "`disabled`", "`no`", "`off`", and "`false`" are synonyms.
- String parameter values should be enclosed in single or double quotes.
- When configuring a lookup or update for a fast reauthentication identity or pseudonym, the configuration parsing requires that a string parameter must be specified. It may be an empty string, i.e. "". This string is used as the `Xstring` value for the lookup and update AATV calls. The Interlink provided AATVs do not require an `Xstring` and so the "" should be used. If a custom AATV is written for lookup or update then that AATV may require a string parameter. See the Software Developers Kit for more details on AATVs and `Xstring`.
- If there is no AATV for the lookup or update for a fast reauthentication identity or pseudonym, then the parameter (e.g. `Fast-Reauth-Lookup`) may be simply not configured at all. Alternatively, the configuration may be explicitly configured with an AATV value of "`NULL`" or "`NONE`".
- Pseudonym lookup is disabled if the realm's configuration specifies that pseudonyms should be algorithmically generated (i.e. "`Generate-Random-Character-Pseudonyms  No`"), but NO pseudonym encryption keys are configured in the `aatv.SIMAKA{}` section of the `aaa.config` file.
- Generation of new pseudonyms is disabled if the realm's configuration specifies that pseudonyms should be generated but no Pseudonym-Algorithm-Current-Key is configured in the `aatv.SIMAKA{}` section of the `aaa.config` file.

The `EAP-Type SIM{}` configuration block can contain any subset, including the empty subset, of the following named parameters:

Parameter	Description
A3-Algorithm	This parameter specifies the default A3 Algorithm for the realm. If the profile for a user in this realm does not specify an A3 Algorithm and if an A3 Algorithm is needed to produce the GSM triplets for this user's authentication, then the A3 Algorithm specified by this parameter is used. See "A3, A8 and AKA Algorithms" on page 311 for details on available algorithms. If not explicitly configured, the default value is specified in the `A3-Algorithm` parameter in the `aatv.EAP-SIM{}` section of the `aaa.config` file.
A8-Algorithm	This parameter specifies the default A8 Algorithm for the realm. If the profile for a user in this realm does not specify an A8 Algorithm and if an A8 Algorithm is needed to produce the GSM triplets for this user's authentication, then the A8 Algorithm specified by this parameter is used. See "A3, A8 and AKA Algorithms" on page 311 for details on available algorithms. If not explicitly configured, the default value is specified in the `A8-Algorithm` parameter in the `aatv.EAP-SIM{}` section of the `aaa.config` file.
Prefixed-IMSI-Permanent-IDs	This parameter indicates whether or not the RAD-Series Server should, for this realm, accept permanent identities of the form '1' + IMSI. The EAP-SIM RFC 4186 indicates that the permanent identity SHOULD be derived from the IMSI, but alternatively, an implementation MAY choose a permanent identity that is not based on the IMSI. The server supports both options. The valid values are "`Enabled`" and "`Disabled`". If not explicitly configured, the default value is specified in the `Prefixed-IMSI-Permanent-IDs` parameter in the `aatv.EAP-SIM{}` section of the `aaa.config` file.
Generic-Permanent-IDs	This parameter indicates whether or not the RAD-Series Server should, for this realm, accept generic permanent identities not based on an IMSI, e.g. "fred". The EAP-SIM RFC 4186 indicates that the permanent identity SHOULD be derived from the IMSI, but alternatively, an implementation MAY choose a permanent identity that is not based on the IMSI. The server supports both options. The valid values are "`Enabled`" and "`Disabled`". If not explicitly configured, the default value is specified in the `Generic-Permanent-IDs` parameter in the `aatv.EAP-SIM{}` section of the `aaa.config` file.

Parameter	**Description**
Minimum-Length-IMSI and Maximum-Length-IMSI	These parameters specify the minimum and maximum length of IMSIs that the RAD-Series Server will accept. The server performs sanity checks on a permanent identity that is offered as an IMSI to ensure that the identity is neither too short nor too long to be an IMSI. The EAP-SIM RFC 4186 explicitly states that 15 is the maximum. The minimum length is 6 based on a 3 digit MCC plus a 2 digit MNC plus a 1 digit MSIN. This is a theoretical absolute minimum length for an IMSI. Therefore the check made is: `6 <= Minimum-Length-IMSI <= Maximum-Length-IMSI <= 15` If not explicitly configured, the default values are specified in the `Minimum-Length-IMSI` and `Maximum-Length-IMSI` parameters in the `aatv.EAP-SIM{}` section of the `aaa.config` file.
Number-Of-Triplets-For-Authentication	This parameter indicates how many GSM triplets are needed for authentication. The EAP-SIM RFC 4186 indicates this value MUST be 2 or 3. If not explicitly configured, the default value is specified in the `Number-Of-Triplets-For-Authentication` parameter in the `aatv.EAP-SIM{}` section of the `aaa.config` file.
Protected-Success-Indications	Protected success indications are an optional EAP-SIM feature which the Interlink EAP-SIM RAD-Series Server supports. This parameter indicates whether the server will offer protected success indications to the peer. The valid values are "`Enabled`" and "`Disabled`". If not explicitly configured, the default value is specified in the `Protected-Success-Indications` parameter in the `aatv.EAP-SIM{}` section of the `aaa.config` file.
Fast-Reauth-Lookup	Fast re-authentication is an optional EAP-SIM feature which the Interlink EAP-SIM RAD-Series Server supports. This parameter specifies an AATV and an Xstring parameter for this AATV. This AATV is invoked to map a fast reauthentication identity to the user's real identity and full authentication context. If this parameter is not configured, then fast re-authentication support is disabled for the realm. The Interlink server provides an AATV, `SIMAKA-ReauthCacheLookup`, for this function. See "Fast Re-Authentication" on page 318 for details. There is no default.

Parameter	Description
Fast-Reauth-Update	Fast re-authentication is an optional EAP-SIM feature which the Interlink EAP-SIM RAD-Series Server supports. This parameter specifies an AATV and an Xstring parameter for this AATV. This AATV is invoked to update the mapping of a fast reauthentication identity to a user's real identity. If this parameter is not configured, then fast re-authentication support is disabled for the realm. The Interlink server provides an AATV, `SIMAKA-ReauthCacheUpdate`, for this function. See "Fast Re-Authentication" on page 318 for details. There is no default.
Pseudonym-Lookup	Pseudonyms are an optional EAP-SIM feature which the Interlink EAP-SIM RAD-Series Server supports. This parameter specifies an AATV and an Xstring parameter for this AATV. This AATV is invoked to map a pseudonym to the user's real identity. If this parameter is not configured, then pseudonym support is disabled for the realm. If this parameter specifies the Interlink-provided AATV `SIMAKA-PseudonymDecrypt`, see "Pseudonym" on page 316, then: • The server forces non-random pseudonym generation for this realm. • If no `Pseudonym-Algorithm-Key-*` parameters are defined in the `aatv.SIMAKA{}` section of the `aaa.config` file, then pseudonym support is disabled. • If at least one of the above mentioned keys is defined and the `Pseudonym-Algorithm-Current-Key` is not defined in the `aatv.SIMAKA{}` section of the `aaa.config` file or does not refer to a defined key, then generation of new pseudonyms is disabled but existing pseudonyms can be looked up. There is no default.
Pseudonym-Update	Pseudonyms are an optional EAP-SIM feature which the Interlink EAP-SIM RAD-Series Server supports. This parameter specifies an AATV and an Xstring parameter for this AATV. This AATV is invoked to update the mapping of a pseudonym to a user's real identity. The Interlink provided pseudonym support using an algorithm, does not require a `Pseudonym-Update AATV`. See "Pseudonym" on page 316. There is no default.

Parameter	Description
Max-Number-Of-Reauths-Before-Full-Auth-Is-Required	Fast re-authentication is an optional EAP-SIM feature which the Interlink EAP-SIM RAD-Series Server supports. This parameter specifies an upper limit for the number of subsequent fast re-authentications allowed before a full authentication needs to be performed. The valid range is 1 to 65,535. If not explicitly configured, the default value is specified in the `Max-Number-Of-Reauths-Before-Full-Auth-Is-Required` parameter in the `aatv.EAP-SIM{}` section of the `aaa.config` file.
Fast-Reauth-Realm	When providing a fast reauthentication identity, the RAD-Series Server also includes a realm to help ensure that the subsequent fast re-authentication be targeted to this server, the only server which holds the full authentication context if internal caching, rather than an external database, is used to save the fast re-authentication context. This parameter specifies such a realm. Since the maximum length of a fast reauth NAI cannot exceed 253 characters and since the length of the fast reauth user name is 10 characters, the Fast Reauth Realm value must not exceed 242 characters. If the fast reauthentication identity should be generated with NO realm name then this would be configured as "`NULL`". The empty string entry, using just two quotes, indicates that the server should generate a fast reauthentication identity with the same realm name as the permanent identity. If not explicitly configured, the default value is specified in the `Fast-Reauth-Realm` parameter in the `aatv.EAP-SIM{}` section of the `aaa.config` file.
Fast-Reauth-Id-Lifetime	Fast re-authentication is a mechanism for an EAP-SIM user to freshen his keys periodically. A fast re-authentication, if it is going to take place, will happen in a "short" time after a full authentication or a previous fast re-authentication. This parameter specifies a lifetime for a fast reauthentication identity, in seconds. If the fast reauthentication identity is assigned and isn't used within this period of time, the fast reauthentication identity and the associated full authentication context is purged. The valid range is 1 to 14400 (1 second to 4 hours). If not explicitly configured, the default value is specified in the `Fast-Reauth-Id-Lifetime` parameter in the `aatv.EAP-SIM{}` section of the `aaa.config` file.

Parameter	Description
Generate-Random-Character-Pseudonyms	The Interlink RAD-Series Server provides a mechanism, using configured encryption keys, by which pseudonyms can be generated as an encrypted form of the permanent identity, which can be subsequently decrypted to reproduce the permanent identity. Alternatively, the server can generate pseudonyms as a string of random characters, similar to the fast reauthentication identity. In this latter case, an external database is required to store the pseudonym to permanent identity mappings. This parameter indicates whether the server generates pseudonyms by algorithm (value=no) or if the server generates random character pseudonyms (value=yes). The valid values are "`Yes`" and "`No`". If not explicitly configured, the default value is specified in the `Generate-Random-Character-Pseudonyms` parameter in the `aatv.EAP-SIM{}` section of the `aaa.config` file.
Pseudonym-Lifetime	A random character pseudonym, when generated for a user, is placed in an external database. This parameter specifies the lifetime of such a generated random character pseudonym. After the specified length of time has elapsed since the pseudonym was first assigned, the pseudonym is invalidated, independent of how many times the pseudonym was used, if ever. The valid range is 1 to 31,622,400 (1 second to 366 days). If not explicitly configured, the default value is specified in the `Pseudonym-Lifetime parameter` in the `aatv.EAP-SIM{}` section of the `aaa.config` file.

Example EAP.authfile configuration file

The following `EAP.authfile` configures the EAP-SIM protocol for a SIM realm:

```
eapsim.com    -EAP   EAP     "comment"
{
  EAP-Type SIM
  {
    A3-Algorithm                           "3GPP-Milenage"
    A8-Algorithm                           "3GPP-Milenage"
    Prefixed-IMSI-Permanent-IDs            "Enabled"
    Generic-Permanent-IDs                  "Enabled"
    Minimum-Length-IMSI                    6
    Maximum-Length-IMSI                    15
    Number-Of-Triplets-For-Authentication  2
    Protected-Success-Indications          "Disabled"

    #           Temporary identity datastores
    Fast-Reauth-Lookup   SIMAKA-ReauthCacheLookup   ""
    Fast-Reauth-Update   SIMAKA-ReauthCacheUpdate   ""
    Pseudonym-Lookup     SIMAKA-PseudonymDecrypt    ""
    Pseudonym-Update     NULL                       ""

    #           Fast Reauth configuration:
    Max-Number-Of-Reauths-Before-Full-Auth-Is-Required  2
    Fast-Reauth-Realm                      "this.server.com"
    Fast-Reauth-Id-Lifetime                3600

    #           Pseudonym configuration:
    Pseudonym-Lifetime                     1209600
    Generate-Random-Character-Pseudonyms   "No"
  }
}
```

Global EAP-SIM Configuration in aaa.config

The `aatv.EAP-SIM{}` configuration block, located within the `aaa.config` file, contains global EAP-SIM configuration information. Some of the parameters represent realm default values for those not specified in the realm configuration. Other parameters represent global defaults which do not correspond to any realm based parameter. For the global parameters common to EAP-SIM and EAP-AKA, see “EAP-AKA and EAP-SIM Common Global Configurations” on page 291.

The following rules apply to the `aatv.EAP-SIM{}` configuration block parameters

- The parameter names are case insensitive.
- For parameters with on/off binary values, the values "`enabled`", "`yes`", "`on`", and "`true`" are synonyms and the values "`disabled`", "`no`", "`off`", and "`false`" are synonyms.
- String parameter values should be enclosed in single or double quotes.

The `aatv.EAP-SIM{}` configuration block, in `aaa.config` file, can contain any subset, including the empty subset, of the following named parameters:

The following parameters are global. No realm configuration overrides.

Parameter	Description
Statistics	This parameter enables/disables the output of EAP-SIM statistics to the logfile when the RAD-Series Server shuts down. The valid values are "`Enabled`" and "`Disabled`". If not explicitly configured, the default value is "`Enabled`"

The following parameters specify server-wide realm defaults.
These are overridable by the realm configuration.

Parameter	Description
A3-Algorithm	This parameter specifies the global default A3 Algorithm. If the profile for a user does not specify an A3 Algorithm and if the realm configuration does not specify an A3 Algorithm and if an A3 Algorithm is needed to produce the GSM triplets for this user's authentication, then the A3 Algorithm specified by this parameter is used. See “A3, A8 and AKA Algorithms” on page 311 for details on available algorithms. If not explicitly configured, there is NO default value.
A8-Algorithm	This parameter specifies the global default A8 Algorithm. If the profile for a user does not specify an A8 Algorithm and if the realm configuration does not specify an A8 Algorithm and if an A8 Algorithm is needed to produce the GSM triplets for this user's authentication, then the A8 Algorithm specified by this parameter is used. See “A3, A8 and AKA Algorithms” on page 311 for details on available algorithms. If not explicitly configured, there is NO default value.
Prefixed-IMSI-Permanent-IDs	This parameter indicates whether or not the RAD-Series Server should accept permanent identities of the form '1' + IMSI for a realm, if the realm configuration does not specify this parameter. The EAP-SIM RFC 4186 indicates that the permanent identity SHOULD be derived from the IMSI, but alternatively, an implementation MAY choose a permanent identity that is not based on the IMSI. The valid values are "`Enabled`" and "`Disabled`". If not explicitly configured, the default value is "`Enabled`".

Parameter	Description
Generic-Permanent-IDs	This parameter indicates whether or not the RAD-Series Server should accept generic permanent identities not based on an IMSI, e.g. "fred", for a realm where the realm configuration does not specify this parameter. The EAP-SIM RFC 4186 indicates that the permanent identity SHOULD be derived from the IMSI, but alternatively, an implementation MAY choose a permanent identity that is not based on the IMSI. The valid values are "`Enabled`" and "`Disabled`". If not explicitly configured, the default value is "`Disabled`".
Minimum-Length-IMSI and Maximum-Length-IMSI	These parameters specify the minimum and maximum length of IMSIs that the RAD-Series Server will accept. The server performs sanity checks on a permanent identity that is offered as an IMSI to ensure that the identity is neither too short nor too long to be an IMSI. The EAP-SIM RFC 4186 explicitly states that 15 is the maximum. The minimum length is 6, based on a 3 digit MCC plus a 2 digit MNC plus a 1 digit MSIN. This is a theoretical absolute minimum length for an IMSI. Therefore the check made is: `6 <= Minimum-Length-IMSI <= Maximum-Length-IMSI <= 15` If not explicitly configured, the default values are 6 and 15.
Number-Of-Triplets-For-Authentication	This parameter indicates how many GSM triplets are needed for authentication. The EAP-SIM RFC 4186 indicates this value MUST be 2 or 3. If not explicitly configured, the default value is 2.
Protected-Success-Indications	Protected success indications are an optional EAP-SIM feature which the Interlink EAP-SIM RAD-Series Server supports. This parameter indicates whether the server will offer protected success indications to the peer. The valid values are "`Enabled`" and "`Disabled`". If not explicitly configured, the default value is "`Enabled`".

Parameter	Description
Max-Number-Of-Reauths-Before-Full-Auth-Is-Required	Fast re-authentication is an optional EAP-SIM feature which the Interlink EAP-SIM RAD-Series Server supports. This parameter specifies an upper limit for the number of subsequent fast re-authentications allowed before a full authentication needs to be performed. The valid range is 1 to 65,535. If not explicitly configured, the default value is 4.
Fast-Reauth-Realm	When providing a fast reauthentication identity, the RAD-Series Server also includes a realm to ensure that the subsequent fast re-authentication be targeted to this server, the only server which holds the full authentication context if internal caching, rather than an external database, is used to save the fast re-authentication context. This parameter specifies such a realm. Since the maximum length of a fast reauth NAI cannot exceed 253 characters and since the length of the fast reauthentication identity is 10 characters, the Fast Reauth Realm value must not exceed 242 characters. If the fast reauthentication identity should be generated with NO realm name then this would be configured as "`NULL`". The empty string entry, using just two quotes, indicates that the server should generate a fast reauthentication identity with the same realm name as the permanent identity. If not explicitly configured, the default value is the empty string entry which indicates the server should generate a fast reauthentication identity with the same realm name as the permanent identity.
Fast-Reauth-Id-Lifetime	Fast re-authentication is a mechanism for a SIM user to freshen his keys periodically. A fast re-authentication, if it is going to take place, will happen in a "short" time after a full authentication or a previous fast re-authentication. This parameter specifies a lifetime for a fast reauthentication identity, in seconds. If the fast reauthentication identity is assigned and isn't used for this period of time, the fast reauthentication identity and the associated full auth context is purged. The valid range is 1 to 14400 (1 second to 4 hours). If not explicitly configured, the default value is 3600 (one hour),

Parameter	Description
Generate-Random-Character-Pseudonyms	Pseudonyms can be generated as an encrypted form of the permanent identity, which can be subsequently decrypted to reproduce the permanent identity. Alternatively, pseudonyms can be generated as a string of random characters, similar to the fast reauthentication identity. In this case, an external database is required to store the pseudonym to permanent identity mappings. This parameter indicates whether the RAD-Series Server generates pseudonyms by algorithm (value = "no") or if the server generates random character pseudonyms (value = "yes"). The valid values are "`Yes`" and "`No`". If not explicitly configured, the default value is "`No`".
Pseudonym-Lifetime	A random character pseudonym, when generated for a user, is placed in an external database. This parameter specifies the lifetime of such a generated random character pseudonym. After the specified length of time has elapsed since the pseudonym was first assigned, the pseudonym is invalidated, independent of how may times the pseudonym was used, if ever used. The valid range is 1 to 31,622,400 (1 second to 366 days). If not explicitly configured, the default value is 1,209,600 (14 days).

Example aaa.config configuration file

```
aatv.EAP-SIM
{
  #   ======================================================================
  #   The following parameters are global. No realm configuration overrides.
  #   ======================================================================

  Statistics                                "Enabled"  # Enabled or Disabled

  #   ======================================================================
  #   All parameters that follow specify server-wide realm defaults.
  #   These are overridable by the realm configuration.
  #   ======================================================================

  Number-Of-Triplets-For-Authentication   2          # range [2 to 3]
  Protected-Success-Indications           "Enabled"  # Enabled or Disabled
  Prefixed-IMSI-Permanent-IDs             "Enabled"  # Enabled or Disabled
  Generic-Permanent-IDs                   "Disabled" # Enabled or Disabled
  Minimum-Length-IMSI 6  # 6<=Minimum-Length-IMSI <= Maximum-Length-IMSI <=15
  Maximum-Length-IMSI 15 # 6<=Minimum-Length-IMSI <= Maximum-Length-IMSI <=15

  A3-Algorithm                            "3GPP-Milenage"
  A8-Algorithm                            "3GPP-Milenage"
```

```
    #   =====================================================================
    #   The following group of parameters configure the PSEUDONYM support
    #   =====================================================================

    Generate-Random-Character-Pseudonyms    "No"       # yes or no

    Pseudonym-Lifetime                      1209600    # 14 days, in seconds

    #   =====================================================================
    #   The following group of parameters configure FAST RE-AUTHENTICATION
    #   =====================================================================

    Max-Number-Of-Reauths-Before-Full-Auth-Is-Required  4  # range [1 to 65535]

    Fast-Reauth-Realm                        ""        # use perm ID’s realm

    Fast-Reauth-Id-Lifetime                  3600      # range [1 to 14400]
  }
```

A3, A8 and AKA Algorithms

If authentication vectors are not retrieved from a datastore or supplied by an external AuC, then they must be generated using A3 and A8 algorithms for EAP-SIM or the AKA algorithm for EAP-AKA.

GSM A3 and A8 algorithms can be used in EAP-SIM. [GSM-03.20] specifies the general GSM authentication procedure and the external interface (inputs and outputs) of the A3 and A8 algorithms. The operation of these functions falls completely within the domain of an individual network operator, and therefore, the functions are specified by each operator rather than being fully standardized. The GSM-MILENAGE algorithm, specified publicly in [3GPP-TS-55.205], is an example algorithm set for A3 and A8 algorithms.

The AKA algorithm can also use the GSM functions that are used to implement A3 and A8 algorithms mentioned above.

The A3/A8/AKA algorithm plug-ins are located in the AATV directory (`/opt/aaa/aatv` by default). There may be multiple named A3/A8/AKA algorithms used by the RAD-Series Server. They can be specified in the global configuration file, `aaa.config`, in the realm-based configurations, or in an users’ profile. See the SDK documentation for information on how to modify the examples or create your own A3/A8/AKA algorithm plug-ins.

3GPP Milenage A3/A8/AKA Algorithm

An implementation of the 3GPP Milenage A3/A8 algorithm functions for EAP-SIM authentication and the AKA algorithm for EAP-AKA are included with the RAD-Series Server. The 3GPP Milenage A3/A8/AKA algorithm plug-in module has configuration parameters which allows it to be customized for a specific operator. The A3, A8 and AKA algorithm names, in this plug-in, are `3GPP-Milenage.`

See the relevant 3GPP documents for complete details on 3GPP Milenage f1, f1*, f2, f3, f4, f5, f5* algorithms:

- 3GPP TS 35.205 v6.0.0 - General Information
- 3GPP TS 35.206 v6.0.0 - Algorithm Specification
- 3GPP TS 35.207 v6.0.0 - Implementors' Test Data
- 3GPP TS.35.208 v6.0.0 - Design Conformance Test Data
- 3GPP TS.35.909 v6.0.0 - Summary and results of design and evaluation
- 3GPP TS.55.205 v6.2.0 - Authentication and Key Generation functions for A3 and A8

The 3GPP Milenage A3/A8/AKA algorithms are based on the 3GPP Milenage functions:

f1(), f1*(), f2(), f3(), f4(), f5(), f5*()

A total of 12 parameters are required to fully specify the function set:

- Ek 128-bit kernel function
- OP 128-bit operator specific value
- C1-C5 128-bit values used to compute f1, f1*, f2, f3, f4, f5, f5*
- R1-R5 integer rotation constants used to compute f1, f1*, f2, f3, f4, f5, f5*

The Ek kernel function specified by 3GPP Milenage is 128-bit AES (Rijndeal). This plug-in module does not allow the Ek kernel function to be changed. The underlying implementation provides support for replacing Ek. If replacing the Ek kernel function is required, see the SDK documentation for further details.

The 3GPP Milenage A3 algorithm has two variants, corresponding to recommended SRES derivation function #1 and recommended SRES derivation function #2. The A3 function is affected by the choice, while the A8 function is unaffected. The selection of A3 variant #1 or #2 constitutes another parameter, A3-Variant. The AKA algorithm is unaffected by this parameter.

The selection of parameter values must match the characteristics of the client devices to be authenticated.

Configuration parameters available in `aatv.3GPP-Milenage{}` block in `aaa.config` file:

Parameter	**Description**
OP	This is a 128-bit operator-specific constant. The `OP` value MUST be specified by each operator. Milenage specifies no default for this value. If not explicitly configured, the default value is 0x00000000.00000000.00000000.00000000. Use of this value will generate a warning message in the logfile.
C1	This is a 128-bit computation constant. `C1` SHOULD have even parity. Use of a value with odd parity will generate a warning message in the logfile. Milenage specifies the default for this value. If not explicitly configured, the default value is 0x00000000.00000000.00000000.00000000.
C2	This is a 128-bit computation constant. `C2` SHOULD have odd parity. Use of a value with even parity will generate a warning message in the logfile. Milenage specifies the default for this value. If not explicitly configured, the default value is 0x00000000.00000000.00000000.00000001.
C3	This is a 128-bit computation constant. `C3` SHOULD have odd parity. Use of a value with even parity will generate a warning message in the logfile. Milenage specifies the default for this value. If not explicitly configured, the default value is 0x00000000.00000000.00000000.00000002.
C4	This is a 128-bit computation constant. `C4` SHOULD have odd parity. Use of a value with even parity will generate a warning message in the logfile. Milenage specifies the default for this value. If not explicitly configured, the default value is 0x00000000.00000000.00000000.00000004.
C5	This is a 128-bit computation constant. `C5` SHOULD have odd parity. Use of a value with even parity will generate a warning message in the logfile. Milenage specifies the default for this value. If not explicitly configured, the default value is 0x00000000.00000000.00000000.00000008.
R1	This is a rotation constant. The valid range is 0 to 127.

Parameter	**Description**
	Milenage specifies the default for this value. If not explicitly configured, the default value is 64.
R2	This is a rotation constant. The valid range is 0 to 127. Milenage specifies the default for this value. If not explicitly configured, the default value is 0.
R3	This is a rotation constant. The valid range is 0 to 127. Milenage specifies the default for this value. If not explicitly configured, the default value is 32.
R4	This is a rotation constant. The valid range is 0 to 127. Milenage specifies the default for this value. If not explicitly configured, the default value is 64.
R5	This is a rotation constant. The valid range is 0 to 127. Milenage specifies the default for this value. If not explicitly configured, the default value is 96.
A3-Variant	This plug-in module supports the selection of Milenage variant #1 or #2. `A3-Variant` MUST be 1 or 2. If an alternative SRES derivation function is required, see the SDK documentation for further details. The AKA algorithm is unaffected by this parameter. If not explicitly configured, the default value is 1.

Note: The Ci,Ri pairs **MUST** be unique.
It must **NOT** be the case that both Ci = Cj and Ri = Rj for $i \neq j$.
For instance, it must not be the case that both C2=C4 and R2=R4.

Example aatv.3GPP-Milenage block in aaa.config File

```
aatv.3GPP-Milenage
{

    # OP   128-bit operator-specific constant: CONFIGURATION RECOMMENDED

    OP 0x00000000.00000000.00000000.00000000

    # C1   128-bit computation constant: CONFIGURATION OPTIONAL.

    C1 0x00000000.00000000.00000000.00000000

    # C2   128-bit computation constant: CONFIGURATION OPTIONAL.

    C2 0x00000000.00000000.00000000.00000001

    # C3   128-bit computation constant: CONFIGURATION OPTIONAL.

    C3 0x00000000.00000000.00000000.00000002

    # C4   128-bit computation constant: CONFIGURATION OPTIONAL.

    C4 0x00000000.00000000.00000000.00000004

    # C5   128-bit computation constant: CONFIGURATION OPTIONAL.

    C5 0x00000000.00000000.00000000.00000008

    # R1   rotation constant: CONFIGURATION OPTIONAL.

    R1   64

    # R2   rotation constant: CONFIGURATION OPTIONAL.

    R2   0

    # R3   rotation constant: CONFIGURATION OPTIONAL.

    R3   32

    # R4   rotation constant: CONFIGURATION OPTIONAL.

    R4   64

    # R5   rotation constant: CONFIGURATION OPTIONAL.

    R5   96

    # A3-Variant   algorithm variant: CONFIGURATION OPTIONAL.

    A3-Variant   1

}
```

Pseudonym and Fast Re-Authentication Data Base AATVs

Pseudonym

The Interlink RAD-Series Server provides a mechanism for EAP-AKA and EAP-SIM, using configured encryption keys, by which pseudonyms can be generated as an encrypted form of the permanent identity, which can be subsequently decrypted to reproduce the permanent identity. Alternatively, the server can generate pseudonyms as a string of random characters, similar to the fast reauthentication identity. In this latter case, an external database is required to store the pseudonym to permanent identity mappings. For many customers, the algorithm based pseudonyms will be the easiest and most efficient choice. The use of random pseudonyms would only be required if the network operator felt the algorithm did not provide adequate hiding of the permanent identity.

In order for pseudonyms to be used, the realm configuration in `EAP-Type AKA{}` and `EAP-Type SIM{}` must specify the `PseudonymLookup` parameter with its value being the name of an AATV which maps the Pseudonym to the permanent identity. The configured AATV may be the Interlink provided one, `SIMAKA-PseudonymDecrypt,` which does local decryption of a pseudonym using the pseudonym encryption key that was used to encrypt the pseudonym. The configured AATV may also be a customer provided AATV which accesses an external database to map the Pseudonym to the real identity. The requirements of a customer implementation are defined in the Software Developers Kit.

Random Pseudonyms

The RAD-Series Server, when operating in an environment where a central database is used for saving the pseudonym to permanent identity mappings, may be configured to generate a pseudonym as a string of random characters, and store the last-used and last-assigned pseudonyms in this database. The EAP-AKA RFC 4187 and EAP-SIM RFC 4186 recommends saving at least two pseudonyms, the last used and the last assigned. In order for random pseudonyms to work, the realm configuration in `EAP-Type AKA{}` block and the `EAP-Type SIM{}` block in the `EAP.authfile` must specify the `Pseudonym-Lookup` and `Pseudonym-Update` parameters with their values being the names of the AATV which maps the pseudonym to the permanent identity and the AATV which stores the random pseudonym in the database. In this case, the pseudonym algorithm would not be employed and the pseudonym would look just like the fast reauthentication identity, with a different prefix. That is: the random pseudonym identity is 10 characters long, consisting of the pseudonym prefix '2' or '4', followed by 9 random characters from the character set {BCDFGHJKLMNPQRSTVWXYZ0123456789}. The random pseudonym has the benefit that there is no way to reverse engineer the permanent identity. It also has the drawback that there must be a database implemented to store and retrieve the mapping of pseudonym to permanent identity.

Algorithm Based Pseudonyms

Interlink has provided an option in the RAD-Series Server that generates a pseudonym by encrypting the real username using an algorithm and an AATV, `SIMAKA-PseudonymDecrypt,` that decrypts a pseudonym to reproduce the real username. The algorithmic approach has these features/benefits, as specified by Ericsson[1] and submitted to the 3GPP TSG SA WG3 working group:

- No external database is needed to store all the assigned pseudonyms.
- A pseudonym generated on one RADIUS server can be processed by a second RADIUS server.
- No user state is kept in the RADIUS server between WLAN sessions.
- Pseudonyms are not stored in the HSS/HLR.
- Any secret keys used in RADIUS server for the generation of pseudonyms are infeasible to recover (even for an attacker that has available a number of matching permanent identities and pseudonyms).
- Given a pseudonym (or even a number of correlated pseudonyms), it is infeasible for an attacker to recover the corresponding permanent identity.
- It is infeasible for an attacker to determine whether or not two pseudonyms correspond to the same permanent identity.
- No random forgery: It is infeasible for an attacker to generate a valid pseudonym (irrespective of the underlying permanent identity).
- No targeted forgery: It is infeasible for an attacker to generate a valid pseudonym corresponding to a given permanent identity.

In order for the Interlink provided algorithm to be used, the global configuration in `aatv.SIMAKA{}` must specify one or more `Pseudonym-Algorithm-Key-`*n* parameters. The key number specified by "`Pseudonym-Algorithm-Current-Key`" is used to encrypt new pseudonyms. The other keys are used for decryption of pseudonyms previously generated with these other keys, but are not used for generation of new pseudonyms. With the algorithm based pseudonyms there is no lifetime applied to the pseudonym. A lifetime can be approximated by defining a new key and making the new key the current key, then after the desired lifetime the old key can be removed and the pseudonyms generated with it will be disabled.

When generating a pseudonym based on a permanent identity which is an IMSI, the RAD-Series Server uses a minor modification of an algorithm developed by Ericsson[2] and submitted to the 3GPP TSG SA WG3 working group. In this case, the pseudonym user name is 24 characters long.

When generating a pseudonym based on a permanent identity which is a generic username, e.g. "fred", the RAD-Series Server uses an algorithm derived from the same Ericsson algorithm. In this case, the length of the pseudonym varies, depending on the length of the permanent identity:

- `24 characters if the permanent user name is     1-8 characters.`
- `45 characters if the permanent user name is    9-24 characters.`

[1] WLAN – Pseudonym Generation for EAP-SIM/AKA, 3GPP, Ericsson, S3-020654.pdf

[2] WLAN – Pseudonym Generation for EAP-SIM/AKA, 3GPP, Ericsson, S3-020654.pdf

- `66 characters if the permanent user name is 25-40 characters.`
- `88 characters if the permanent user name is 41-56 characters.`
- `109 characters if the permanent user name is 57-72 characters.`
- `130 characters if the permanent user name is 73-88 characters.`
- `152 characters if the permanent user name is 89-104 characters.`
- `173 characters if the permanent user name is 104-120 characters.`
- `194 characters if the permanent user name is 121-136 characters.`
- `216 characters if the permanent user name is 137-152 characters.`
- `237 characters if the permanent user name is 153-168 characters.`
- `The pseudonym is not generated if the permanent user name is > 168 characters, as the pseudonym identity would exceed 253 characters.`

The RAD-Series Server will generate a pseudonym identity only if the length of the "pseudonym@realrealm" string will not exceed 253 characters.

For a given IMSI permanent identity, there are 56 random user bits involved in the pseudonym generation, resulting in over 7 million trillion (7 *10^{18}) different pseudonyms for a given IMSI. The probability that a random forgery decrypts back into anything that looks like an IMSI is less than 1 in 4 million.

For a given non-IMSI permanent identity, there are 32 random user bits involved in the pseudonym generation, resulting in over 4 billion different pseudonyms for a given user. The probability that a random forgery decrypts back into anything that looks like generic username is less than 1 in 50 million.

Fast Re-Authentication

Fast re-authentication is a mechanism for a SIM user to freshen his keys periodically. A fast re-authentication, if it is going to take place, will happen in a "short" time after a full authentication or a previous fast re-authentication. If the fast reauthentication identity is assigned and isn't used within this period of time, the fast reauthentication identity and the associated full authentication context is expired.

In order for fast re-authentications to be used, the realm configuration in `EAP-Type AKA{}` and `EAP-Type SIM{}` must specify:

- The Fast-Reauth-Update parameter with its value being the name of an AATV which saves the fast reauthentication identity to permanent identity mapping as well as the other fast re-authentication context (the Master Key from the user's last full authentication, the fast re-authentication counter, and the fast reauthentication identity expiration time).
- The Fast-Reauth-Lookup parameter with its value being the name of an AATV which maps the fast reauthentication identity to the permanent identity and returns the saved context.

The configured AATVs may be the Interlink provided set, `SIMAKA-ReauthCacheUpdate and SIMAKA-ReauthCacheLookup,` which do local caching of the fast reauthentication identity to the permanent identity mapping and the necessary attributes. The configured AATVs may also be a customer provided set of AATVs which access an external database to map the fast reauthentication identity to the permanent identity and returns the saved attributes. The requirements of a customer implementation are defined in the Software Developers Kit (SDK).

The Interlink EAP-AKA/SIM RAD-Series Server generates a fast reauthentication identity which is 10 characters long, consisting of the fast reauthentication identity prefix '3' or '5', followed by 9 random characters from the 31 character set consisting of the upper-case characters minus the vowels, plus the 10 digits 0-9, i.e. {BCDFGHJKLMNPQRSTVWXYZ0123456789}.

Since there are 31 choices for each of the 9 random characters, there are then 31^9 different, i.e. more than 26 trillion, fast reauth identities over the universe of all permanent identities.

Selecting only uppercase characters for the server-generated reauth identities allows for case-insensitive databases lookups.

The RAD-Series Server sends a fast reauthentication identity to the client, which includes a realm. Before generating a fast reauthentication identity, the server first checks that the total length of the “name@realm” string does not exceed 253 characters, the maximum length of a User-Name attribute value. If it is too long, the server does not generate a reauth identity. Since the name portion of the fast reauthentication identity is 10 characters, this problem only occurs if the realm is greater than 242 characters. The realm is either the configured fast reauth realm or the realm from the permanent identity. Recall that a fast reauth realm can be configured for the purpose of targeting a fast reauth authentication request to the specific server which generated the fast reauthentication identity.

Index