INTERLINK NETWORKS

Technical Document

RAD-Series AAA Server Administrator's Guide

Version 8.0 for Solaris

© 2005-2011 Interlink Networks, LLC. All Rights Reserved.

This document is copyrighted by Interlink Networks, LLC. (Interlink Networks). The information contained within this document is subject to change without notice. Interlink Networks does not guarantee the accuracy of the information.

Trademark Information

Brand or product names may be registered trademarks of their respective owners.

Revision History

Date	Version	Type
10/2006	7.2.0	New Product Release
12/2007	7.3.0	New Product Release
07/2008	7.4.0	New Product Release
03/2011	8.0.0	New product release

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

Main - (734) 821-1200
Sales - (734) 821-1228
Fax - (734) 821-1235
Website - www.interlinknetworks.com

Table of Contents

About this Document ix
Audience ix
Notational Conventions ix
Introduction 1
Overview 1
Software Components 2
AAA Server 2
Server Manager 3
Remote Control 3
How the AAA Server Works 4
RADIUS Message Exchange 4
Authentication 7
Authorization 8
Accounting 10
WLAN Security 12
EAP Authentication 12
Tunneled EAP Authentication 14
Preparing the WLAN 16
Choosing an EAP Method 17
Supported EAP Methods 18
Process for Securing WLANs 19
Understanding Server Manager 20
Navigating in Server Manager 20
Configuring Servers through Server Manager 22
Managing Multiple Servers 23
Configuring Server Manager 24
Changing the UDP Port Number 24
Changing the Shared Secret 25
Changing the User Name and Password 25
Configuring Server Manager for SSL 26
Starting Server Manager 28
Stopping Server Manager 28

Starting Remote Control 28
Accessing Server Manager 29

Managed Servers 30

Adding Servers 30
Modifying Server Profiles 31
Deleting Servers 31

Server Administration 32

Starting the Server 32
Setting Server Start Options 33
UDP Ports 33
Debug Level 33
Reset Logs 33
To Change Server Start Options 34
Stopping the Server 34
Restarting the Server 34
Reporting Server Status 35
Extended Server Status Output 35
Changing Status Command Options 36
Reporting Server Time 37

Load Configuration 38

Loading Configurations into Server Manager 38

Save Configuration 39

Saving Configurations 39

Edit Configuration 40

Defining Access Devices 40
Using Wild Cards for IP Addresses 40
Adding an Access Device 41
Modifying a Device Definition 43
Deleting a Device 43
Defining Proxies 44
Self-referring Client Entry (localhost) 44
Using Wild Cards For Realms 44
Receiving Requests from a Proxy Server 45
Proxying Requests to a Remote Server 46

Modifying a Proxy Configuration 47
Deleting a Proxy Configuration 47
Proxying Accounting Requests to a Central Server 48
Defining Realms 49
About Realms 49
Configuring TTLS Authentication 50
Configuring PEAP Authentication 51
Configuring TLS Authentication 52
Defining Authentication Realms 53
Identifying LDAP Directories 54
Identifying Oracle Servers 58
Modifying a Realm Definition 59
Deleting a Realm 59
Administering Digital Certificates 60
Using the "Self-Signed" Digital Certificates 61
Using Your Own Digital Certificates and Keys 62
Defining Digital Certificate Locations 64
Defining Users 65
User A-V Pairs 65
User Name Formats 65
Adding User Profiles 66
Controlling and Provisioning Sessions 67
Modifying a User Profile 71
Deleting Users from Realm Files 71
Defining Server Properties 72
Modifying Server Properties 72
Certificate Properties 72
DHCPv4 Relay Properties 73
DNS Update Properties 73
Message Handling Properties 74
Miscellaneous Properties 75
ProLDAP Properties 76
Session Table Properties 77
SNMP Properties 78
Tunneling Properties 78

Using SNMP 79
Enabling and Disabling SNMP 79
MIB Objects 79
Using DHCP 80
Required DHCP Server Features 80
Process for Using DHCP 80
Configuring the RAD-Series Server for DHCP 81
Setting Up Realms for DHCP 82
Defining Address Pools for Specific Users 83
Configuring the DHCPv4 Server 83
Tunneling 84
Tunneling Hints 84
Establishing a Tunnel for a User 85
Tunneling Attributes 86
Tagged Tunneling Attributes 87
Reversing Configuration Changes 88
Maintenance 89
Reporting Server Log File 89
Reporting User Accounting Records 91
Viewing the Accounting Log 91
Accounting Record Format 92
Modifying the Accounting Log 94
Modifying Accounting Logging Behavior 96
Reporting Server Statistics 97
Reporting Active Server Sessions 98
Viewing Active Sessions 98
Stopping Active Sessions 98
Server Programs 99
radiusd 99
radiusd.sh 101
radcheck 102
radpwtst 103
radsignal 106
radrecord 108
Troubleshooting 111

Debugging 111
Error Messages 111

Using External Data Stores 114
Using an LDAP Server 114
About ProLDAP™ 114
Securing LDAP Communications 115
Password Encryption 116
Extending the ProLDAP Schema 117
LDIF User Entry Syntax 118
Check and Reply Items 119
Known Issues with ProLDAP™ 120
Using Oracle® Database 121
Requirements 121
Oracle Data Store Processing 121
Storing User Profiles in Oracle 122
Configuring db_srv Daemon 125

Using Advanced Policy 127
Decision Files 127
Implementing Advanced Policies 128
Policy Syntax 134
Attribute Specifications 137
Value Specifications 141
Attribute Value Handling 144
Operators 145
Comparison operators 145
Boolean Expressions 153
Action Commands 157
Attribute Functions 167
Internal Attributes 173
Calling Decision Files 174
Dynamic Access Control 175
Modifying the FSM for DAC 176
DAC Decision File 176
DNIS Routing 177

Modifying the FSM for DNIS . . . 177
DNIS Decision File . . . 178

Configuration Files . . . 179
About Configuration Files . . . 179
File Format . . . 179
File Location . . . 180
Commonly Used Files . . . 181
Attribute-Value Pairs . . . 182
aaa.config . . . 185
Including Files . . . 185
Server Variable Syntax . . . 185
Tunneling Properties . . . 203
authfile and EAP.authfile . . . 204
Authfile Entry Syntax . . . 204
Protocol Flags . . . 205
Case-sensitivity Flags . . . 206
Authentication Types . . . 207
Prefixed users and authfiles . . . 210
clients . . . 211
Clients Entry Syntax . . . 211
Realm files (.users) and default users file . . . 215
User Entry Syntax . . . 215
General Configuration Attributes Used as Check Items . . . 217
LAS Configuration Attributes Used as Check Items . . . 218
Check and Reply Items . . . 219
las.conf . . . 220
LAS Session Configuration . . . 220
LAS Realm Configuration . . . 222
Logging LAS Realms . . . 223
vendors . . . 224
Version Tracking . . . 224
Vendor Entry Syntax . . . 224
dictionary . . . 226
Version Tracking . . . 226
Attribute Entry Syntax . . . 226

Pruning Expressions . . . 227
Value Entry Syntax . . . 228
log.config . . . 229
Default Entry . . . 229
Default Path Entry . . . 229
Default-dateformat Entry . . . 229
Default-dateformat-timezone Entry . . . 229
Nodefault Entry . . . 230
Stream Entry . . . 230
Logging Multiple Streams . . . 233
iaaaAgent.conf . . . 234
Finite State Machine (FSM) . . . 234
Version Tracking . . . 234
States . . . 234
Events . . . 236
Actions . . . 239
Predefined .fsm Files . . . 241
Managing RADIUS Sockets . . . 242
Managing RADIUS Listen Sockets . . . 242
Managing RADIUS Proxying Sockets . . . 244
Closing and Reopening Listen Sockets and Proxy Sockets upon a HUP . . . 244
IPv6 Operation . . . 245
The IPv6 Readiness Tool . . . 246
The "ipv6_enabled" configuration parameter . . . 246
Configuring EAP-AKA . . . 248
EAP-AKA Features . . . 248
EAP-AKA User Credential Lookup Configuration . . . 249
EAP-AKA Realm-Based Configurations . . . 250
Global EAP-AKA Configuration in aaa.config . . . 260
EAP-AKA and EAP-SIM Common Global Configurations . . . 265
Configuring EAP-SIM . . . 268
EAP-SIM Features . . . 268
EAP-SIM User Credential Lookup Configuration . . . 269
EAP-SIM Realm-Based Configurations . . . 271
Global EAP-SIM Configuration in aaa.config . . . 281

A3, A8 and AKA Algorithms 286
Pseudonym and Fast Re-Authentication Data Base AATVs 291

Index 295

About this Document

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

- Server components and operation
- Server Manager configuration
- Advanced server configuration, including:
  - EAP authentication
  - External data stores
  - Advanced Policy
- Server and accounting logging and monitoring

See the *RAD-Series Getting Started Guide* for installation and basic configuration instructions.

Audience

This *Administrator's Guide* is for Network and Systems Administrators who must configure and maintain the AAA Server. It's assumed 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 or Oracle 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 to the account or to open the document in your browser.

Introduction

Overview

The Interlink Networks RAD-Series AAA 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 AAA 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 server also supports:

- Proxying authentication and accounting messages to another RADIUS server
- Redundant LDAP or Oracle® user data stores, including load balancing and failover functions
- SNMP integration
- DHCP relay from configured IP address pools
- Advanced Policy to implement:
  - DNIS routing
  - Dynamic Access Control
  - Authorization based on logical groups

Software Components

The RAD-Series software delivered to you includes:

- AAA Server RADIUS server application
- Server Manager administration console
- Remote Control

You may have additional components, depending on your license option.

AAA Server

AAA Server application is comprised of:

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

When the server is started, it loads and initializes the software modules and reads the configuration files.

Finite State Machine

At the core of the server is the Finite State Machine (FSM) and associated state tables that define all processes for handling RADIUS requests. At server 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 server comes with a default state table capable of handling standard RADIUS and EAP requests with no modification. 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

AAA 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.

To customize the AAA 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. Most configuration parameters are modified through the Server Manager, although some AAA Server features can only be configured by using a text editor. The files you'll most commonly work with are:

- **`aaa.config`** - defines all 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`** - defines all attributes and values that may be used to build A-V pairs recognized by the 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.
- **`EAP.authfile -`** defines realm authentication actions.
- **`las.conf`** - enables session tracking and specifies some session timing values.
- **`log.config -`** defines accounting message logging behavior.
- **`radius.fsm -`** is the Finite State Machine table. You can edit this to reorder 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.

Server Manager

Server Manager is a browser-based application for configuring and managing AAA Servers on local or remote machines. It can also be used to view or print server and accounting logs. Server Manager has two main components:

- Server Manager program
- Java graphical user interface

The Java interface can be customized for your application by using the UI API, which is part of the Software Developers Kit.

Server Manager requires the Java Run-Time Environment (JRE) installed on each administrator workstation from which you will access it. See the *Getting Started Guide* for more information.

Remote Control

The Remote Control program contains the Java RMI objects that facilitate communication between remote servers and the Server Manager program. You only need to install this component

on server machines that do not have Server Manager installed.

How the AAA Server Works

RADIUS Message Exchange

The 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.

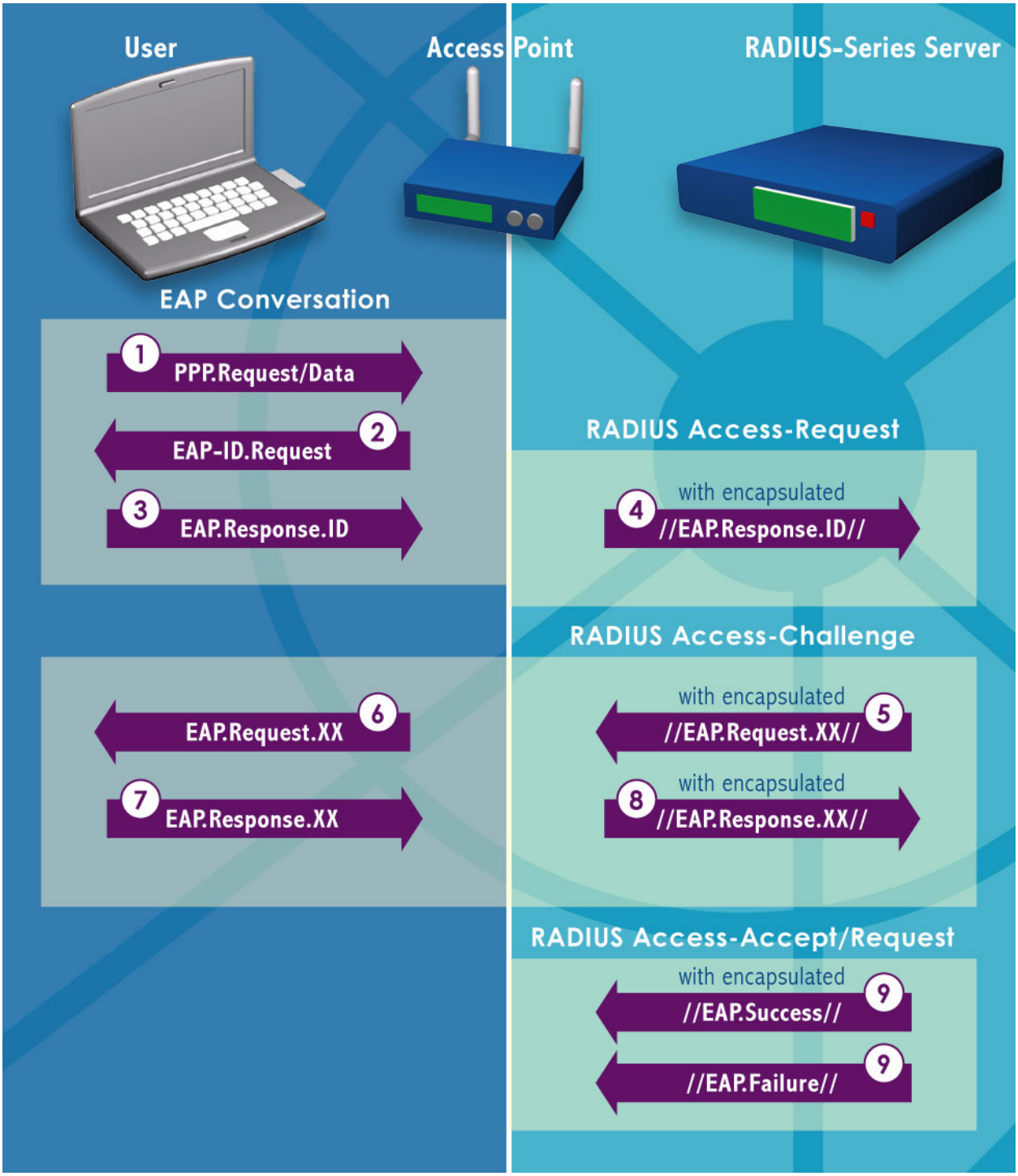
When the server receives a request, it first validates the access device that sent the request, then the user. If the access device is permitted to send requests to the server, the AAA software takes information from the **Access-Request** message and attempts to match the request to a realm user profile 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 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 AAA configuration file, the A-V pairs usually follow the syntax:

```
AttributeName=Value
```

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

```
AttributeName:Type=Value
```

Request Processing

When the AAA 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 server an Access-Request containing the User-Name from which the Server get the user ID and realm.

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

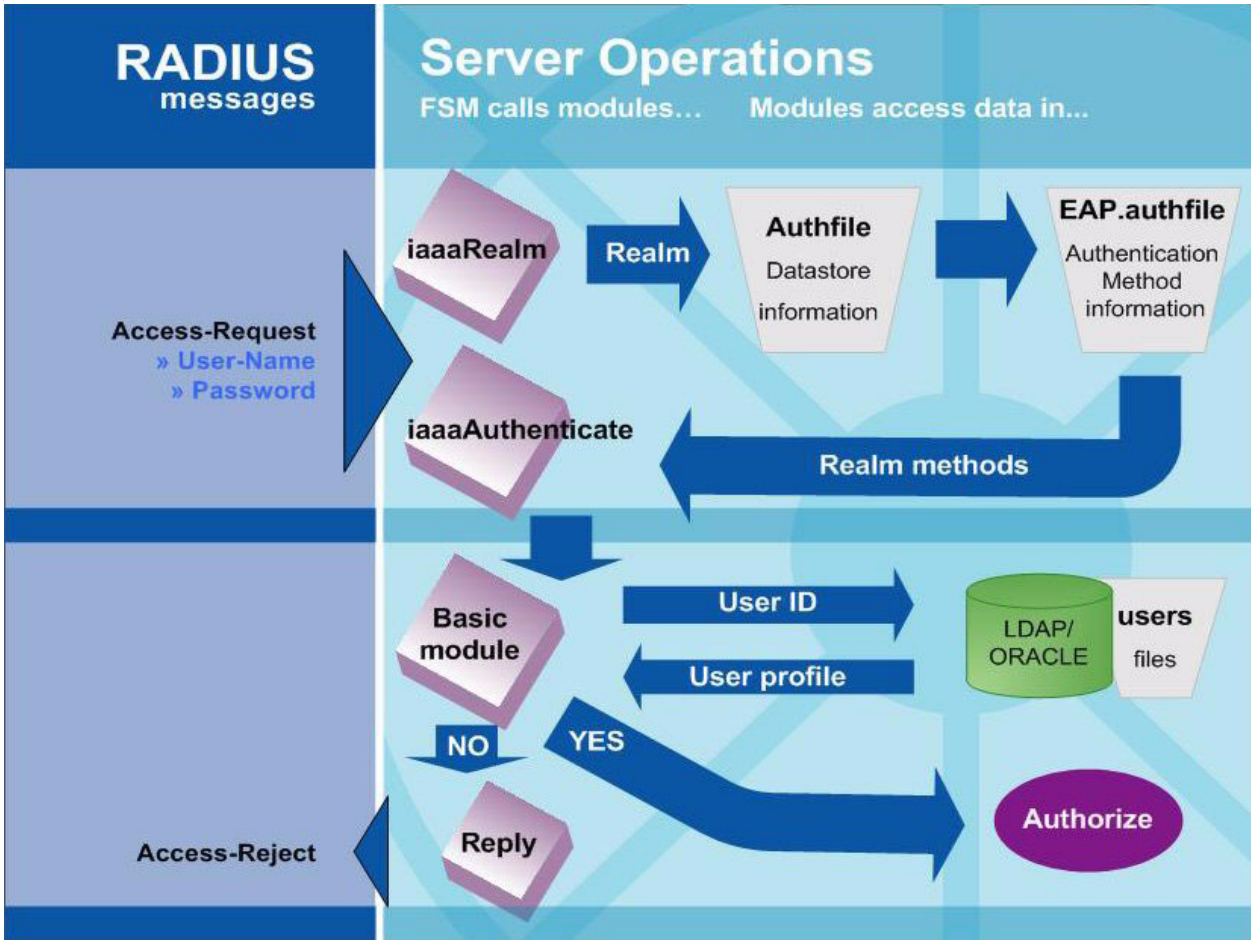
3 If the username is not found in the default users file then the Server finds the realm's user profile storage entry in the `authfile`.

4 The 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 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 AAA 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.

Note: You need the Advanced Policy Engine license to implement advanced policy.

---

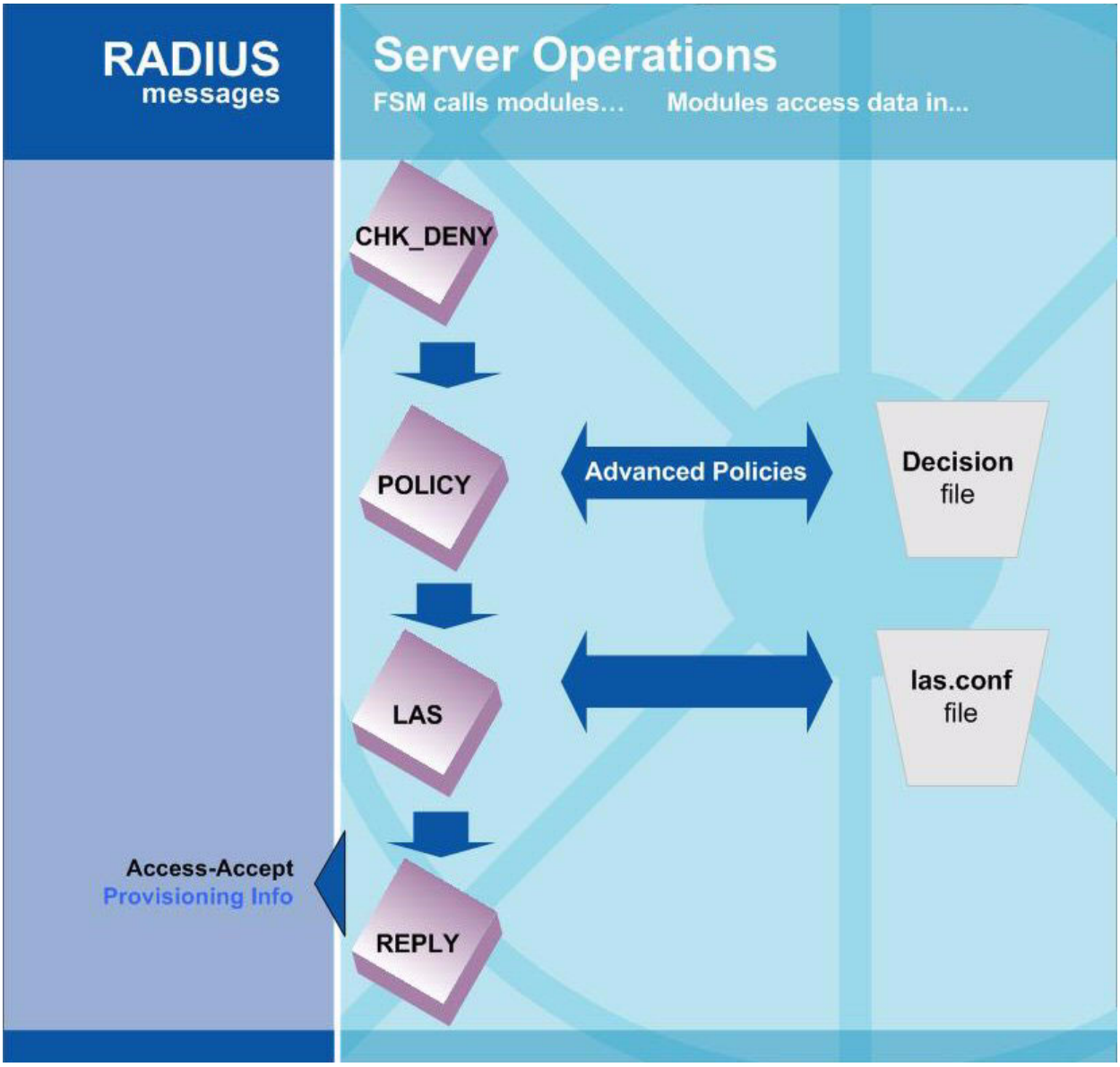
1 The server evaluates the request against any check and deny items stored with the user profile. For example: a check item indicating that the request must be from port 1 on the access device must match a corresponding `NAS-Port=1` A-V pair in the request for the request to be accepted.

2 If the Advanced Policy Engine is licensed and 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 The server retrieves realm authorization policies from the `las.conf` file (if LAS is enabled).

4 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 AAA 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 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 server to start recording the session.

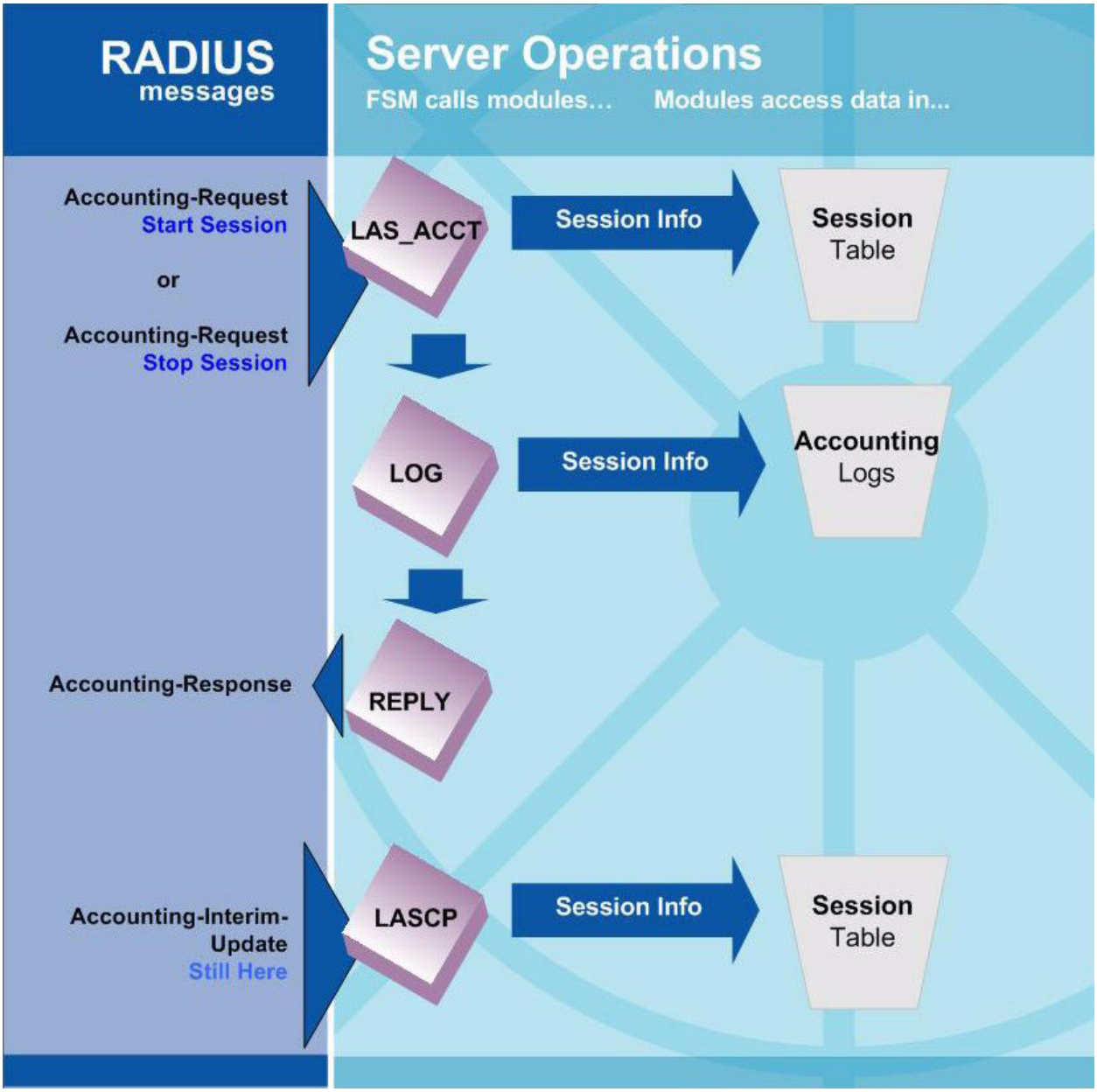
2 The server checks the `log.config` file for the realm's logging format.

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 AAA Server provides security framework to support EAP authentication mechanisms for Wireless Local Area Network (WLAN) users. The AAA 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 AAA 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 server an Access-Request containing the encapsulated EAP Identity Response message.

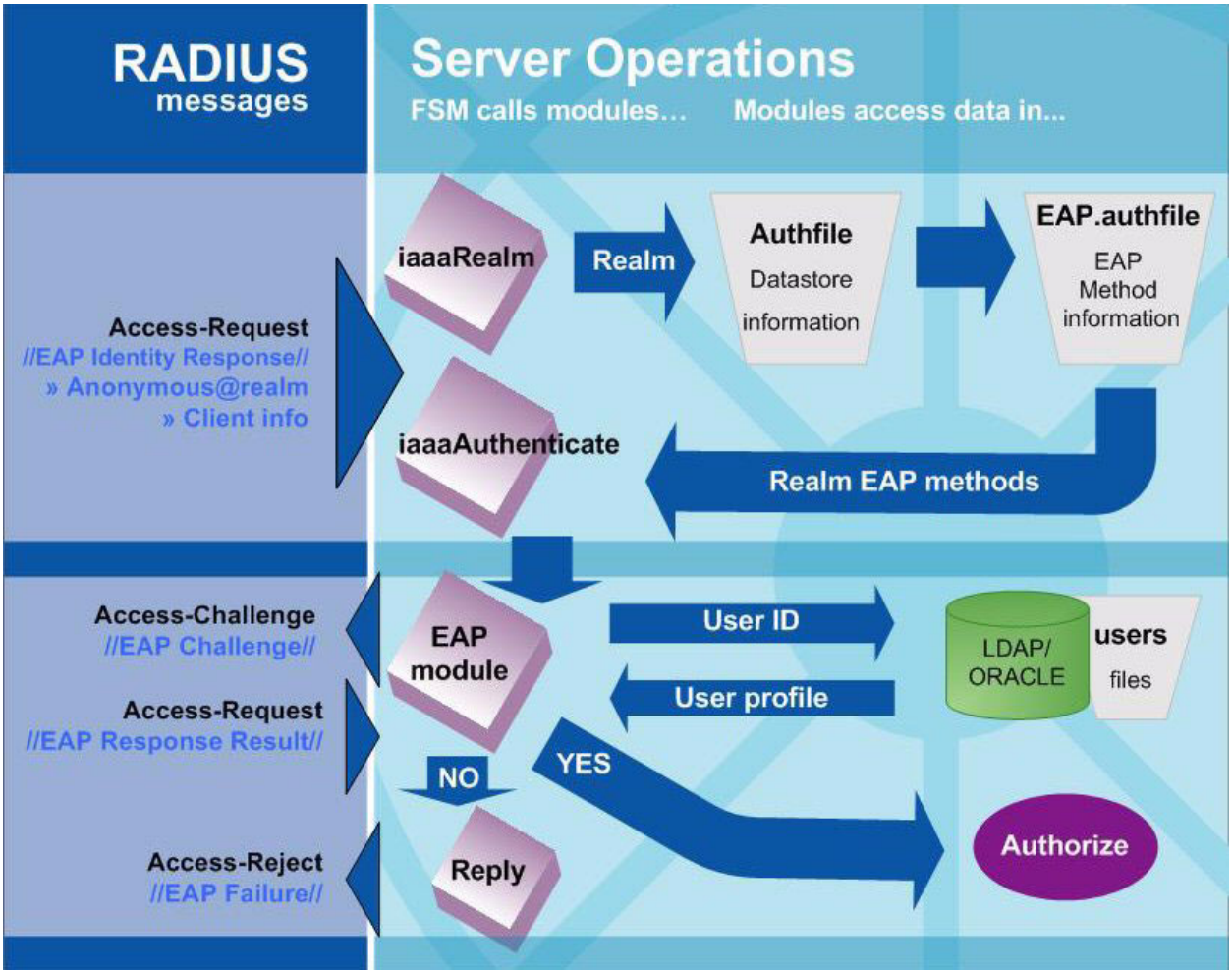
2 The Server checks for the user profile in the default users file and if found uses it for step 4.

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 AAA 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.

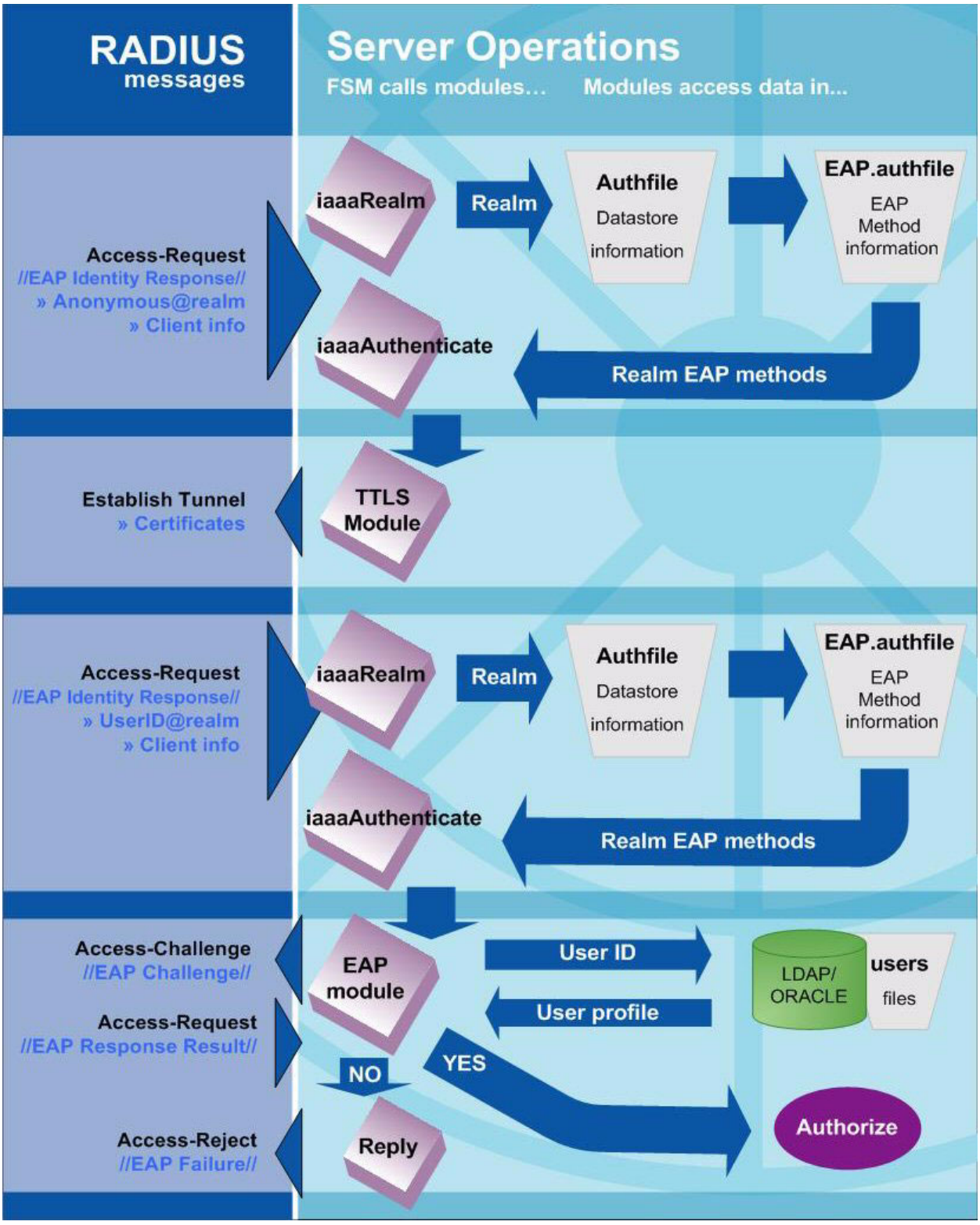
5 The user sends an Access-Request message through the tunnel containing the actual user information.

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 AAA 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 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 the supplicants support the EAP methods the 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 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 server must have access to a repository with information for each user. The server supports several different methods for retrieving user information. Server Manager can administer user information stored locally in realm flat files. Use the Users link to administer the default set of users. Use the Local Realms link and select the users icon to administer a specific set of Users associated with a realm.
Client Certificate	Client Supplicant	For TLS only. The digital certificate identifying the client
Client CA Certificate	Server	For TLS only. Used by AAA 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 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 AAA 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 AAA 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 AAA server to secure your WLAN. See the following sections of this guide for procedures explaining each of the steps.

1 Access Server Manager. See "Accessing Server Manager" on page 29.

2 If the server is remote, add the server to the Server Manager's Managed Servers list. See "Adding Servers" on page 30.

3 Load the AAA Server configuration into Server Manger. See "Loading Configurations into Server Manager" on page 38.

4 Identify the access devices that will send access requests to the AAA Server. See "Defining Access Devices" on page 40.

5 Configure tunnel realms if you are using TTLS. See "Defining Realms" on page 49.

6 Configure user authentication realms and user profile data stores. See "Defining Realms" on page 49.

7 Configure user profiles to identify each user accessing services through the AAA Server.

If you are using local realm files or the default users file, see "Defining Users" on page 65.

If you are using an LDAP directory, see "Using an LDAP Server" on page 114.

If you are using an Oracle database, see "Using Oracle® Database" on page 121.

8 Configure digital certificates if you are using TLS, TTLS, or PEAP. See "Administering Digital Certificates" on page 60.

9 Deploy the configuration by:

- Saving the configuration to one or more AAA Servers. See "Saving Configurations" on page 39.
- Stopping and starting the AAA Server program. See "Server Administration" on page 32.

Understanding Server Manager

Server Manager is the browser-based application for remotely managing your AAA Servers. It provides administrator access to configuration, administration, and monitoring functions from any networked workstation with an Internet browser.

This section describes the Server Manager graphical user interface. It shows how to:

- Use the Server Manager navigational elements
- Select servers for configuration and administration

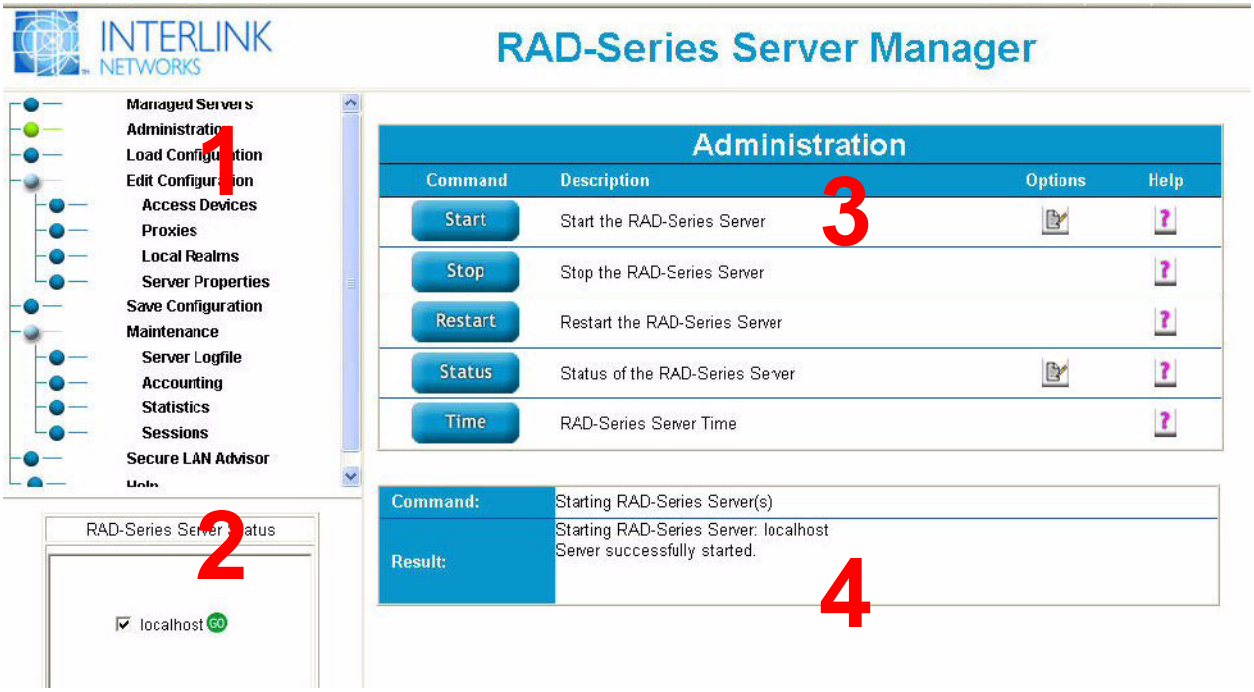
There are also instructions for reconfiguring the Server Manager:

- Administrative User Name and Password
- UDP Port Number
- Java Connector type (may use an SSL connection)

Navigating in Server Manager

The Server Manager screen contains four principal work areas

:



1. Navigation frame

The Navigation frame on the left of the screen lists the tasks you can perform on the selected AAA Servers. Clicking one of the links opens the corresponding page in the Workspace frame.

2. Server Status frame

The Server Status frame, located beneath the Navigation frame, allows you to select the AAA Servers to be managed before executing commands in the Workspace.

The icons next to each server indicate its current status.

- — Server Manager cannot connect to the administrative agent (RMI object) on the AAA Server.
- — the AAA Server is running.
- — the AAA Server is not running.

3. Workspace frame

The Workspace frame is the main work area for the Server Manager. The content of this frame changes depending on the task you've selected from the Navigation frame.

For some tasks, pop-up dialogs appear above the Workspace frame for entering additional configuration parameters, for example, when configuring realm entries and setting LDAP directory attributes.

4. Message frame

The area below the Workspace frame displays the results of administrative operations, such as starting and stopping the server.

The Secure LAN Advisor

The Secure LAN Advisor is an HTML tutorial/help system in the Server Manager GUI that walks you through the tasks for securing WLANs with the AAA Server. The Secure LAN Advisor provides information only—it does not edit configuration files. Follow the Secure LAN Advisor while using Server Manager to create and deploy basic AAA configurations for securing WLANs.

Configuring Servers through Server Manager

Server Manager provides a temporary workspace for editing configurations. As you work, changes are stored in local temporary files before being saved to the actual server configuration files. There is only one set of temporary files in Server Manager, reflecting the server configuration that was last downloaded.

To configure your AAA Servers in Server Manager, you'll:

1 Select managed AAA Server(s).

2 Load the configuration files from a single AAA Server.

3 Edit the configuration as necessary.

4 Save the new configuration to the selected AAA Server(s).

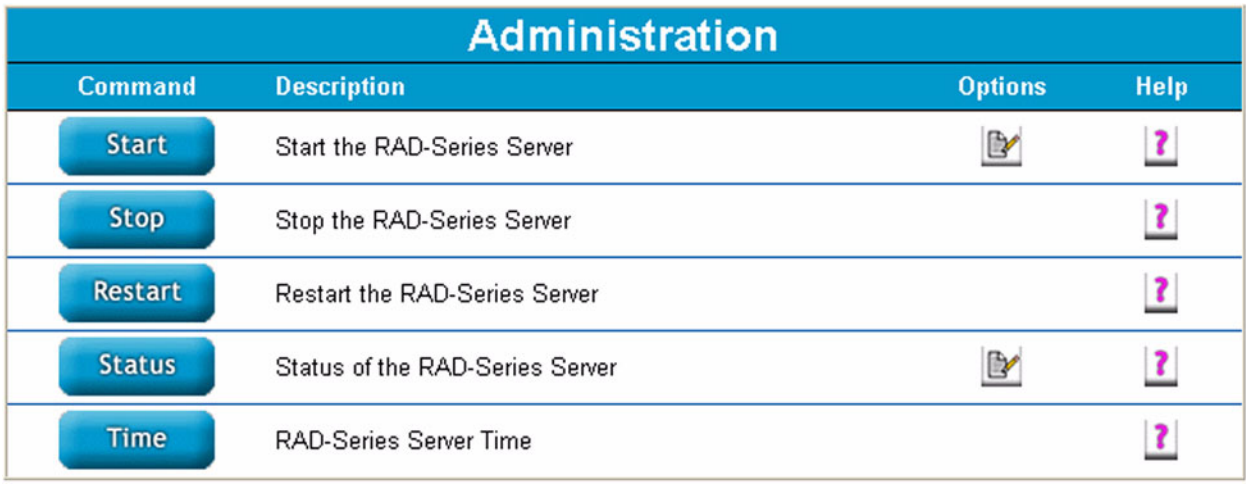
5 Stop and start the AAA Server(s) to activate the new configuration.

Managing Multiple Servers

The AAA Server Status frame in the lower left corner of the Server Manager screen shows a list of managed servers. This list enables you to see the current status of the servers, as well as to select them for administration.

Check the server in the Status frame to select it before executing commands in the Workspace.

Use the Server Manager Administration page to execute basic server commands. You can also use this page to change the server startup settings or reset the log files.



Server Manager can be used to view active session data, read server and accounting logs, and view server statistics. See "Maintenance" on page 89 for more information.

Configuring Server Manager

Certain features of the Server Manager program can be reconfigured in a text editor.

Changing the UDP Port Number

Server Manager uses two UDP port numbers to listen for information from the workstation browsers:

- Port 8080 for non-SSL (http) connections
- Port 8443 for SSL (https) connections

Port 8080 is enabled by default. To switch ports or to change either port number:

1 In a text editor, open the file `server.xml` (in `/opt/tomcat/conf` by default).

2 Locate the lines:

```
<Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
      port="8080" minProcessors="5" maxProcessors="75"
      enableLookups="true" redirectPort="8443"
      acceptCount="100" debug="0" connectionTimeout="20000"
      useURIValidationHack="false" disableUploadTimeout="true" />
```

and:

```
<Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
      port="8443" minProcessors="5" maxProcessors="75"
      enableLookups="true"
      acceptCount="100" debug="0" scheme="https" secure="true"
      useURIValidationHack="false" disableUploadTimeout="true" >
```

3 To switch from port 8080 to port 8443 (for SSL), remove the **<!--** and **-->** brackets surrounding the 8443 Connector tag and comment out the 8080 Connector tag by placing the brackets around it.

4 To change port numbers, change each instance of "8080" or "8443" to the port number you wish to use.

5 Save and close `server.xml`.

6 Stop and restart Server Manager.

Changing the Shared Secret

Information exchanged between AAA Servers and the Server Manager program is validated by a shared secret. This secret **must** be the same for all AAA Servers connected to the Server Manager.

1 In a text editor, open the file `gui.properties` (in `/opt/tomcat/webapps/aaa/WEB-INF` by default).

2 Find the line:

`rmi.config.secret = !!config.secret!!`

Replace `config.secret` with the shared secret you wish to use.

Note: In gui.properties, the slash character must be used as an escape for any special characters in your secret. rmiserver.properties does not require an escape character.

---

3 Save and close `gui.properties`.

4 For each server you have installed, repeat steps 2 and 3 for the file `rmiserver.properties` (in `/opt/aaa/remotecontrol` by default).

5 Stop Server Manager and Remote Control.

6 Start Server Manager and Remote Control.

Changing the User Name and Password

To change the Server Manager administrative user name or password:

1 In a text editor, open the file `aaa-users.xml` (in `/opt/tomcat/conf/` by default).

2 Find the line:

`<user name="`*`New-UserName`*`" password="`*`New-Password`*`" roles="tomcat" />`

Replace *`New-UserName`* or *`New-Password`* with the values you wish to use.

3 Save and close `aaa-users.xml`.

4 Stop and restart Server Manager.

Configuring Server Manager for SSL

To secure data passed between the AAA Server Manager program and your administrator workstations, we strongly recommend that you implement a Secure Socket Layer connection (https). Using SSL requires that, on the Server Manager host machine, you:

- Install Java Secure Socket Extension (JSSE) 1.0.2 or higher
- Create a keystore containing a new certificate
- Configure Server Manager for SSL

Install JSSE

1 Download the JSSE and unpack the JSSE distribution package.

2 Copy `jcert.jar`, `jnet.jar` and `jsse.jar` into the `/lib/ext` directory found in your Java home directory (`/opt/aaa/java` by default).

If this directory doesn't already exist in your Java home directory, create it.

3 If it does not exist, create the file `/lib/security/java.security` in your Java home directory.

4 Add the following to the `java.security` file:

```
security.provider.2=com.sun.net.ssl.internal.ssl.Provider
```

Create a Keystore and Certificate

The keystore is a repository file that contains keys and certificates. If you do not have an existing keystore, you'll need to create one and add a certificate for SSL to it.

You can create a certificate with the `keytool` utility included with the JDK. Refer to the JDK documentation for more information.

1 Change directory to the Java `bin` directory (`/opt/aaa/java/bin` by default).

2 Run `keytool`.

```
keytool -genkey -alias tomcat -keyalg RSA -keystore Keystore-path
```

For *`Keystore-path`* enter the full path of an existing keystore or specify a new one.

3 Follow the prompts to provide information about the keystore and certificate.

The final prompt asks for the certificate password. Make up a password and note it for use in `server.xml`. You can press Enter at this prompt.

The keystore will be created in the location you specified.

Edit Server.XML

1 In a text editor, open the file `server.xml` (in `/opt/tomcat/conf/` by default).

2 Locate and uncomment the following lines by deleting <!-- and -->:

```
<!--
<Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
      port="8443" minProcessors="5" maxProcessors="75" enableLookups="true"
      acceptCount="100" debug="0" scheme="https" secure="true"
      useURIValidationHack="false" disableUploadTimeout="true" >
<Factory className="org.apache.coyote.tomcat4.CoyoteServerSocketFactory"
      keystoreFile="<<the keystore path specified when you ran keytool>>"
      keystorePass="<<the password specified when you ran keytool>>"
      clientAuth="false" protocol="TLS" />
</Connector>
-->
```

3 Edit the `keystoreFile` and `keystorePass` parameters. For example:

```
<Factory className="org.apache.coyote.tomcat4.CoyoteServerSocketFactory"
      keystoreFile="/opt/aaa/java/lib/security/mykeystore"
      keystorePass="topsecret"
      clientAuth="false" protocol="TLS" />
```

You can optionally change the UDP port number by replacing 8443 with the number you wish to use.

4 Comment out the Connector block for port 8080 by inserting <!-- and --> before and after it.

5 Add the JSSE `.jar` files to your `CLASSPATH` by adding these lines to the Server Manager `startup.sh` file (in `/opt/tomcat/bin` by default):

```
CLASSPATH=${CLASSPATH}:${JAVA_HOME}/lib/ext/jsse.jar
CLASSPATH=${CLASSPATH}:${JAVA_HOME}/lib/ext/jnet.jar
CLASSPATH=${CLASSPATH}:${JAVA_HOME}/lib/ext/jcert.jar
export CLASSPATH
```

6 Run `shutdown.sh` (in `/opt/tomcat/bin` by default) to stop Server Manager.

7 Run `startup.sh` (in `/opt/tomcat/bin` by default) to restart Server Manager.

Starting Server Manager

On the machine where you've installed Server Manager:

1 Open your terminal window.

2 Change directory to `/Server Manager directory/bin` (`/opt/tomcat/bin` by default).

3 Run `startup.sh`.

If the process fails to start:

4 Check the log file `catalina.out` (in `/opt/tomcat/logs` by default).

If you see the message "Root cause - Address already in use," the default port 8080 is already being used by another process.

5 At your shell prompt, enter `lsof -i :8080` to identify the process currently using port 8080.

6 Either change the Server Manager port or kill the process using the port.

Stopping Server Manager

To stop the Server Manager program:

1 Open your terminal window.

2 Change directory to `/Server Manager directory/bin`.

3 Run `shutdown.sh`.

Starting Remote Control

Remote Control is installed on remote servers to connect them to the Server Manager interface. This program must be started on the host machine before you can use Server Manager to connect to the server. On each remote machine where you've installed Remote Control:

1 Open a terminal window.

2 Change directory to the `/remotecontrol` directory (default is `/opt/aaa/remotecontrol`).

3 Run `rmistart.sh`.

Accessing Server Manager

Once the Server Manager program is started, to access the graphical user interface:

1 Open your browser and enter the URL `http://IP-address:Port/aaa` where:
   - *IP-address* is the IP for the machine that hosts the Server Manager program.
   - *Port* is the port number assigned to Server Manager. Default is 8080.

2 Enter the Server Manager username and password you choose at installation.

   You'll see the Server Manager Administration page.

3 If this is the first time you're accessing Server Manager, enter the user name and password again to authenticate to the Java Run-Time Environment.

   You'll see the Server Manager Administration page, with localhost listed in the Status frame.

Tip: Bookmark this page in your browser for ease of access later on.

If you see a Java run-time error in the Status frame, the workstation may not have the Java Run-Time Environment installed or the second Java logon box may be hiding behind some window, awaiting your entry.

Managed Servers

Server Manager utilizes the Remote Control program installed on the AAA Servers in your network to form a Java RMI connection to them. Once Server Manager connects to an AAA Server, it is listed among the Managed Servers. You can download and upload configuration and log file data or issue server commands.

This section shows how to:

- Add AAA Servers to the list to be managed through Server Manager
- Change managed AAA Server profiles
- Delete AAA Servers from the Managed Servers list

Adding Servers

All AAA Servers managed by the Server Manager are shown in the Server Status frame in the lower left corner of the screen. Follow these steps to add a server to the list.

Note: The server installed on the same machine as the Server Manager is automatically added as **localhost**. It uses the loopback (local) address 127.0.0.1.

1 In the Navigation frame, click Managed Servers.

2 Click the Manage New Server link.

3 On the Add Server page, enter the server's:
   - Name — label to identify the server in Server Manager
   - Fully-qualified Domain Name or IP address

4 Click the Create button.

   The AAA Server's name appears in the Status frame, along with its status.

Modifying Server Profiles

You can change the name, domain name or IP address for any AAA Server currently listed in the Server Manager Server Status frame. You can also modify AAA Server start options for those servers already in the connected state.

After a server has been connected, the Managed Server:Modify Server page shows a list of installation directories under Server Properties. These paths must match the server's actual file directories.

1 In the Navigation frame, click the Managed Servers link.

2 On the Managed Servers page, click the server name link.

3 Modify the fields as necessary.

4 Click the Modify button.

Deleting Servers

To remove an AAA Server from the Managed Servers list:

1 In the Navigation frame, click the Managed Servers link.

2 Click the button next to the server name.

3 Click Delete to confirm the deletion.

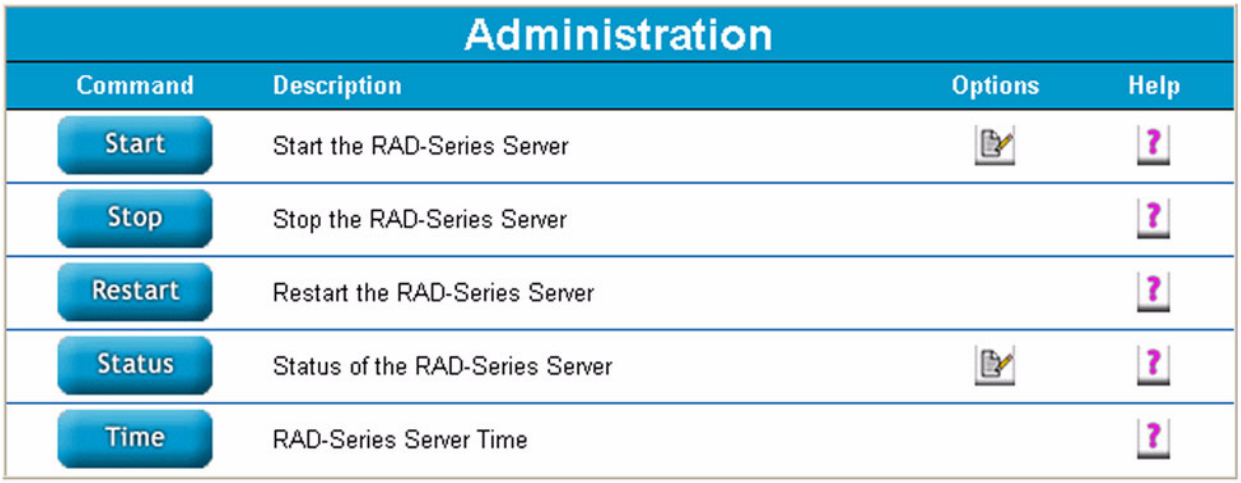
The server name should disappear from the Managed Servers page and the Server Status frame.

Server Administration

This section shows how to issue commands to:

- Start and stop the server
- Change the server's startup options
- Restart the server after changing configurations, without stopping the service
- Report on server status
- Change the options for the status check program
- Report on server time

The Server Manager Administration page provides controls for executing server commands.



The Message frame below the Administration page displays the most recently executed command and its result (server messages). Initially, this frame is empty.

Starting the Server

To start one or more AAA Servers:

1 In the Server Status frame, select the server(s) to start. Be sure only these servers are selected.

2 In the Navigation frame, click Administration.

3 Click Start.

You'll see the result of each server start in the Message frame. There should be a GO next to the server(s) in the Server Status frame.

Setting Server Start Options

Use the Server Start Options page to assign non-default arguments for AAA Server startup.

Note: These startup arguments are not part of the AAA Server configuration files. They are stored locally on the Server Manager and apply to all AAA Servers started by the Server Manager.

UDP Ports

You can change the port the server uses to listen for authentication or accounting requests from the UDP standard 1812 and 1813.

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

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

Reset Logs

You can set the server's log file and session table to back up to a file with the extension `.old` and to restart each time the server is started. Clearing the log file also clears the debug file.

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

To Change Server Start Options

1 In the Navigation frame, click Administration.

2 Click the button next to the Start button to display the Start Options page.

3 Make the desired changes.

4 Click OK.

Stopping the Server

To stop one or more AAA Servers:

1 In the Server Status frame, select the server(s) to stop. Be sure only these servers are selected.

2 In the Navigation frame, click Administration.

3 Click Stop.

You'll see the results of each server stop in the Message frame. There should be a

next to the server(s) name.

Restarting the Server

The Restart function causes selected AAA Server(s) to reload a new configuration without first being stopped and restarted.

1 In the Server Status frame, select the server(s) to restart. Be sure only the desired servers are selected.

2 In the Navigation frame, click Administration.

3 Click Restart.

You'll see the results of each server restart in the Message frame.

You must stop and start the AAA 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

Reporting Server Status

Use the Status function to check the operational status of the selected AAA Server(s).

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

1 In the Navigation frame, click Administration.

2 In the Status frame, select the AAA Server(s) for which to report status.

3 Click Status.

You'll see the results of each status check in the Message frame.

Extended Server Status Output

The extended status message contains this additional information

Message Lines	Description
MF: vp=44/18 auth=1/0 waldo=0/0 redo=0/0 DNS-MF: client=4/0 addr=3/0 name=3/0 CLIENT-MF: vendor=31/0 vendor_list=6/0 (found=6) LAS-MF: sess=1/0 str=0/0 abs=0/0 db=0/0	Memory allocation information in the format of ***=allocated/freed*** for A-V pairs, authreqs, clients, IP addresses, DNS names, vendors, vendor lists and sessions.
Status: 0 authen, 0 unconfirmed, 0 connected, 0 suspended, 0 unknown Status: 0 disconn, 0 reject, 0 no-token, 0 cancel, 0 collision	Local Authorization Service statistics. Will report 0 for all statuses unless Session Tracking is enabled for some realm.
auth queue: Qmax(40000) hwm(1) Qcur(1) rReq(1) xAcc(0) xRej(0) rDup(0) rBad(0)	RADIUS authentication request statistics: queue maximum, high water mark, current number, received requests, transmitted accepts, transmitted rejects, duplicate requests, bad requests.
acct queue: Qmax(40000) hwm(1) Qcur(0) rReq(1) xResp(1) rDup(0) rBad(0)	RADIUS accounting request statistics: queue maximum, high water mark, current number, received requests, transmitted responses, duplicate requests, bad requests.
authfile (x) clients(x) users(x) Start time(x.x)	The number of: defined realms (in authfile), clients, users (in default users file) and the server's last start time.

Message Lines	Description
fsmid(default-7.4-6) dictid(8.0-0), vendid(8.0-0)	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.
Version 8.0.0 AAA_POLL BINARY_AATV CHK_SHELLS CHK_TOKEN EMAIL LAS = 8.0.0	Server version and build information.
"t25(2026)" is responding	A success message. If radcheck fails, one of the following messages will appear: • No reply from server *Name (UDP-port)* • Received non-matching id in server response • Received invalid reply digest from server • No such server: *Name*
Number of retries(x)	Only appears when greater than 0.
Exit Codes	0 - Successful completion -2 or 254 - Remote server had errors -1 or 255 - Local errors 1 - Timeout errors

Changing Status Command Options

To change the values used in the Status command:

1 In the Status frame, select the server.

2 In the Navigation frame, click Administration.

3 Click the button next to the Status button to display the Status Options page.

4 Change the:
- Timeout (Seconds)
- Number Retries

5 Click OK.

Reporting Server Time

To check the system time on servers:

1 In the Status frame, select the server(s).

2 In the Navigation frame, click Administration.

3 Click Time.
You'll see the results in the Message frame.

Load Configuration

This section shows how to load server configurations into Server Manager for editing.

Loading copies configuration files from a source AAA Server installation into the Server Manager workspace. You can download a configuration from any one server, edit it and save it to many servers.

Loading overwrites the temporary files in Server Manager's workspace with the selected server's active configuration files. If you do not save the previous set of configurations back to one of the connected servers, any work done in Server Manager is lost when you reload the configuration. Do not reload between sessions if you are still configuring the same server, unless you wish to revert to the active server configuration.

The Server Manager only shows configurations for the last server loaded. When you wish to work with a different server, save the current configuration (if you wish to preserve it), then load the new server.

Loading does not copy any server security certificates or other security files. Manually copy these files between servers if they are to be shared.

Loading Configurations into Server Manager

To copy an AAA Server configuration into the Server Manager temporary workspace:

1 In the Navigation frame, click Load Configurations.

2 In the Workspace frame, select the source server from which to copy.

3 Click the Load button. Wait until the transfer is confirmed.

Save Configuration

This section shows you how to:

- Save configurations after editing them in Server Manager
- Push configurations from one AAA Server to another

Server Manager preserves changes to the temporary files between sessions, but you must save to copy these changes to the server's actual configuration files. The changes will not take effect until you restart the server, so you can save changes any number of times without affecting the server operation.

You can save a configuration to any (or all) servers that have an active connection with the Server Manager program. This way, you can reconfigure several servers at once or push configurations from a test machine to a production machine. The Save page always shows from which server the configuration originated and lets you select to which it should be saved.

Note: Server Manager saves the entire workspace configuration (access devices, proxies, local realms, users and server properties) to the servers you select. It does not save server security certificates and other security files. You must manually copy these files between servers if they are to be shared.

Saving Configurations

1 In the Navigation frame, click Save Configurations.

2 In the Workspace frame, select the destination server(s). You can select as many as you wish.

3 Click the Save button. Wait until the transfer is confirmed.

Edit Configuration

This section contains instructions on using the Server Manager to configure 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 114 to configure the server for use with an LDAP directory or other external data store.

Defining Access Devices

In Server Manager, 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 proxy servers, which are configured separately under Proxies. The server configuration must include an entry for all the access devices that will communicate with the server.

These procedures modify the server's `clients` file.

See "Defining Proxies" on page 44 for instructions on defining proxy servers.

Using Wild Cards for IP Addresses

When specifying a client device name, you can use the CIDR subnet notation to specify a set of contiguous IPv4 or IPv6 addresses. You can optionally use the *-style wild card 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::/0` is entered, then any IPv6 client knowing the shared secret can access the RAD-Series Server or if a 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 wild card patterns are allowed, where h is a hexadecimal digit and w, x, y, or z are 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 represents the same wild card pattern, since they will match the same set of IP addresses.

No non-zero bits may follow the wild card specification. The following are examples of invalid wild cards:

```
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 In the Navigation frame, click Access Devices.

2 Click the New Access Device link or .

3 In Name, enter the device's IP address or fully-qualified domain name.

4 Enter and confirm the Shared Secret to be used between this device and the server.

The secret must be less than 1023 characters and cannot contain spaces.

5 The Vendors field determines which vendor-specific attributes are returned to the client in RADIUS messages. Use CTRL + Click to select all that apply.

Usually, it is sufficient to select the hardware vendor. If you don't want to send any VSAs, choose Generic.

6 To define additional instructions for handling Access-Requests, check the Option boxes. These are advanced features that are not required normally.

Option	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.
No Encaps	Do not encapsulate vendor response.This option is useful if the client requires nonencapsulated A-V pairs.
Old CHAP	Use pre-RFC CHAP with this client.

7 Click Create.

Modifying a Device Definition

To change settings for an existing device:

1 In the Navigation frame, click Access Devices.

2 Click the device name link on the Access Devices page.

3 Change the settings.

4 Click Modify.

Deleting a Device

To completely remove a device from the server's list of clients:

1 In the Navigation frame, click Access Devices.

2 Click the next to the device name on the Access Devices page.

3 Click Delete.

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

Self-referring Client Entry (localhost)

If the server has a self-referring client entry named **localhost**, the server Status command 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 seriously alter server functioning. See "Reporting Server Status" on page 35 for a description of the extended output.

Note: The localhost entry shown in the Proxies list should not be confused with the loopback localhost that appears in the Server Status frame and corresponds to an RAD-Series Server installed on the same host machine as the Server Manager.

Using Wild Cards For Realms

When specifying a proxy realm name, you can use the wild card 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 wild card `*.company.com`, all three realms would be forwarded to the same server.

It does not matter in what order you enter regular or wild card realm names in the Server Manager. When the server reads the realms it sorts the entries so that:

- Non-wild card entries come first
- Wild card entries of length *n* come before wild card entries of < *n*.

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

Receiving Requests from a Proxy Server

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

1 In the Navigation frame, click Proxies.

2 Click the New Proxy link.

3 In Name, enter the IP address or fully-qualified domain name of the proxy server.

4 In Shared Secret, enter the string used to establish trust between the proxy server and the RAD-Series Server.

The shared secret cannot be more than 1023 characters long or contain spaces.

5 The Vendors field determines which vendor-specific attributes are forwarded to the server in RADIUS messages. Use CTRL + Click to select all that apply.

Usually, it is sufficient to select the hardware vendor. If you don't want to send any VSAs, choose Generic.

6 Check the Response options to define any additional instructions for handling Access-Request messages. Unless you have special requirements, you probably do not need any options.

Options are:

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.
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.
Append attributes	This is useful when a remote server does not return all of the A-V pairs that it received followed by all the additional A-V pairs it wants to add. If it is not set, the server will append all the A-V pairs received from a remote server to the new A-V pairs sent in the response message.

7 Click Create.

Proxying Requests to a Remote Server

Most all the information needed for this can be configured using the Server Manager. If you have a rare requirement where you have a "dual homed" host, you will need to manually edit the clients file to add the `srcip=<ipaddress>` parameter to the entry created with the Server Manager (see "Clients Entry Syntax" on page 211 for information on this). To configure the RAD-Series Server as a proxy to a remote server:

1 In the Navigation frame, click Proxies.

2 Click the New Proxy link.

3 In Name, enter the IP address or fully-qualified domain name of the remote server.

4 In Shared Secret, enter the string used to establish trust between the remote server and the RAD-Series Server.

The shared secret cannot be more than 1023 characters long or contain spaces.

5 The Vendors field determines which vendor-specific attributes are to be forwarded to the remote server in request messages. Generally you would select Generic, but you can choose a set that applies if there are special requirements.

Use CTRL + Click to select more than one.

6 Enter all the Realms to forward to the remote server:

- Choose Add New Realm from the drop-down menu.
- On the Proxy Realm dialog, enter the realm Name.
- To Forward Accounting as well as authentication requests, click the Yes option button.
- Click Save.

Repeat this step for each realm to be proxied.

7 To send authentication requests to a port other than the server's default, enter the port number in Authentication Relay Port.

This number overrides any relay port you set up under the server's start options, content of /etc/services, or its 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.

---

8 To send accounting requests to a port other than the servers default, enter the port number in Accounting Relay Port.

This number overrides any relay port you set up under the server's start options, content of /etc/services, or its default of 1813.

9 To send all A-V pairs when returning messages to the authentication server, turn on (Yes) Append Attributes.

10 Click Create.

Modifying a Proxy Configuration

To change the settings for any proxy relationship between the RAD-Series Server and another server:

1 In the Navigation frame, click Proxies.

2 Select the other server from the Proxies list.

3 Change the settings as necessary.

To stop proxying a realm:

- Select the realm from the Realms to Forward drop-down menu.
- Click Delete.

To change ports: enter the new Authentication Relay Port or Accounting Relay Port number.

To stop forwarding accounting messages:

- Select the realm from the Realms to Forward drop-down menu.
- Change the Forward Accounting selection to Yes or No.

4 Click Modify.

Deleting a Proxy Configuration

To completely stop receiving requests from or forwarding requests to a remote server:

1 In the Navigation frame, click Proxies.

2 Click the button next to the server name the Proxies list.

3 Click Delete.

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 46.

To proxy all accounting:

1 Complete the steps to add the server to your Proxies list.

2 In a text editor, open `radius.fsm` (by default in `etc/opt/aaa/`) and locate the lines:

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

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.

3 To forward the accounting to a port other than the server's default, modify the `clients` file for that port, see "Proxying Requests to a Remote Server" on page 46.

4 Save and close `radius.fsm`.

5 Stop and start the server.

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 directories 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, use the Secure LAN Advisor from the Server Manager Navigation frame.

These procedures modify the server's `authfile` and `EAP.authfile`.

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 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.

Wild Cards

The wild card * can be used in authentication realm names, as well as for proxy realms. See "Using Wild Cards For Realms" on page 44 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 the outer realm user (e.g.: 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, so you know the realm of the anonymous user.

2 Add a tunnel realm.

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 46.

4 Generate and install server-side certificates and keys. See "Administering Digital Certificates" on page 60.

Add a Tunnel Realm

To add a TTLS tunnel realm:

1 In the Navigation frame, click Local Realms.

2 Click the New Local Realm link or .

3 In Name, enter the anonymous user's realm name as set up on the user machines.

4 In Realm Type, choose TTLS Tunnel.

5 Click Create.

Add Authentication Realms

Follow the steps in "Defining Authentication Realms" on page 53 to add the authentication realm.

- In Realm Type, choose Authentication.
- For TTLS with PAP or CHAP, select Password Authentication for the Security Method.
- For TTLS with EAP, select EAP Authentication for the Security Method, then choose all the methods set up on your clients from the drop-down list. Use CTRL + Click to select more than one.

Configuring PEAP Authentication

PEAP implementations use the same realm name for the outer and inner user, even though it is a two-phase authentication method. You need only one authentication realm for each domain name.

You cannot authenticate users on a legacy RADIUS server with PEAP.

The process for configuring PEAP realms is:

1 Configure your PEAP clients.

Note: PEAP client software is installed with Windows® XP Service Pack 1(or later), Windows® Vista, Windows® 7, and third-party supplicants.

---

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

3 Generate and install server-side certificates and keys. See "Administering Digital Certificates" on page 60.

Add Authentication Realm

Follow the steps in "Defining Authentication Realms" on page 53 to add the authentication realm.

- In Realm Type, choose Authentication.
- For the Security Method, choose EAP Authentication, then select the all the PEAP methods set up on your clients from the drop-down list. If you are using any third party supplicants that do not support PEAP version 1, you will need to select "PEAP Version 1 Disabled".

Configuring TLS Authentication

TLS does not access a data store for user profiles. Instead, this method requires certificates to be installed on both the RAD-Series Server and on each user workstation.

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 60.
   - 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 field that was selected in Server Properties > Certificate Properties > Client User Name Attribute. See "Client User Name Attribute" on page 73.

Add Authentication Realm(s)

Follow the steps in "Defining Authentication Realms" on page 53 to add the authentication realm.

- In Realm Type, choose Authentication.
- In User Profile Storage, choose No Store - EAP TLS Certificates.

Defining Authentication Realms

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

See the Secure LAN Advisor (found in the Navigation frame) for more detailed information on configuring RAD-Series Servers for WLAN and 802.1X environments.

1 In the Navigation frame, click Local Realms.

2 Click the New Local Realm link or .

3 In Name, enter the domain name users enter at login.

4 In Realm Type, choose Authentication.

5 In User Profile Storage, choose the type of data store you're using for this realm.
- LDAP — LDAP directory
- Local Storage — flat file identified by name in User Storage Parameters
- No Store - TLS — authenticate using client and server security certificates, rather than user profiles
- No Store - Allow — no authentication, allow all users in this realm
- No Store - Deny — no authentication, deny all users in this realm
- Oracle — Oracle database. Requires special licensing option.
- OS Security Database — UNIX password storage
- SecurID/ACE server — RSA SecurID identification and authentication. Requires special licensing option.
- KTH Kerberos IV — authenticate against a KTH Kerberos version IV server. Requires special licensing option.
- CNS Kerberos IV — authenticate against a CNS Kerberos version IV server. Requires special licensing option.

6 Enter any additional User Storage Parameters that appear:
- LDAP — complete steps in "Identifying LDAP Directories" on page 54
- Local Storage — enter unique realm file name or select the default users file
- Oracle — complete steps in "Identifying Oracle Servers" on page 58

7 Select the Security Method:
- For PAP, CHAP, or MS-CHAP, choose Password Authentication.
- For all other 802.1X methods, select EAP Authentication, then choose all the methods set up on your clients from the drop-down list.

Use CTRL + Click to select more than one option.

You can use the `authfile` to manually define custom authentication types. See "Authfile Entry Syntax" on page 204 for instructions.

8 To define a packet filter for the realm, enter the filter name in Filter ID.

This will override any explicit filters defined in user profiles.

9 To limit concurrent sessions, click Yes to enable Session Tracking.

Note: Enabling Session Tracking sets the number of concurrent sessions to the global Simultaneous Use value set in Session Table 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.

10 Click Create.

Identifying LDAP Directories

When you choose LDAP from the User Profile Storage drop-down list on the Add a Realm page, a set of additional parameters appears under User Storage Parameters. These fields let you identify the LDAP directories from which the RAD-Series Server will retrieve user profiles.

1 To treat *userid* as binary, case-sensitive, choose `BIN` for the `Filter-Type`. To treat *userid* as case-insensitive, choose `CIS` for the `Filter-Type`.

2 To search the LDAP Directory based on an attribute from the request other than the default one, User-Id, change the `Request-Attribute-For-Search` field to the desired attribute name.

3 From the drop-down list that appears in User Profile Storage Parameters, choose New LDAP Directory.

4 On the LDAP Directory dialog, enter the following fields:

Field	Description
Directory Name	Name of the directory that appears in the Server Manager drop-down list. This does not have to be the actual directory name, just an identifier.
Host	IPv4 address or fully-qualified domain name of the host the LDAP directory runs on. NOTE: A fully-qualified domain name that maps to an IPv6 address can be configured here but entering a IPv6 address will not. To do that you must manually edit the authfile, see "ProLDAP Authentication Type" on page 207.
Port	TCP port number the LDAP directory runs on. If no value is given, defaults to either 389 for non-SSL or 636 for SSL ports.
Use SSL	Enable (Yes) or disable (No) SSL between the RAD-Series Server and this LDAP directory. If using SSL, also specify the server's CA certificate path and file in the "ProLDAP Properties" on page 76.

Field	Description
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.
Password / Confirm Password	Password used by the `Administrator` to bind to the LDAP directory server. Enter it twice to confirm it.
Search Base	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.
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`.
Access Mode	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.

5 Click Save.

6 To associate other LDAP directories with this realm, repeat Steps 3 through 5.

Each realm may be configured with up-to-four redundant LDAP directories, which are used by the server when it performs load balancing and failover.

7 Complete the procedure to add or modify the realm.

Modify a Directory Configuration

To change the information entered for any of the defined LDAP directories:

1 Click Local Realms, then click the button next to the realm name to display the Modify Realm page.

2 Select the directory from the User Storage Parameters drop-down list.

3 On the LDAP Directory dialog, make the necessary changes.

4 Click Save.

5 Click Modify.

Delete a Directory

1 Click Local Realms, then click the button next to the realm name to display the Modify Realm page.

2 Select the directory from the User Storage Parameters drop-down list.

3 On the LDAP Directory dialog, click Delete.

4 Click Modify.

Identifying Oracle Servers

When you choose Oracle from the User Profile Storage drop-down list on the Server Manager Add a Realm page, a set of additional parameters appears under User Storage Parameters. These fields let you identify the Oracle daemon (`db_srv`) that will retrieve user profiles for the AAA Server.

1 From the drop-down list that appears in User Profile Storage Parameters, choose New Oracle Server.

2 On the Oracle Server dialog, enter the following fields. All are required:

Field	Description
Database Name	Name for this Oracle database to appear in the User Storage list. Does not have to be the database name in Oracle.
Host	IP address or fully-qualified domain name of the db_srv daemon host machine.
Port	TCP port number the db_srv daemon uses for authentication messages.

Each listed server must have a unique host name and port (you cannot have two servers running on the same machine and listening to the same port number).

3 Click Save.

4 To associate other Oracle servers with this realm, repeat Steps 2 through 4.

Each realm may be configured with up-to-32 redundant Oracle servers, which are used by the server when it performs load balancing and failover.

5 Complete the procedure to add or modify the realm.

Modify an Oracle Server Configuration

To change the information entered for any of the defined Oracle servers:

1 Click Local Realms, then click the button next to the realm name to display the Modify Realm page.

2 Select the server from the User Storage Parameters drop-down list.

3 On the Oracle Server dialog, make the necessary changes.

4 Click Save.

5 Click Modify.

Delete an Oracle Server

1 Click Local Realms, then click the button next to the realm name to display the Modify Realm page.

2 Select the server from the User Storage Parameters drop-down list.

3 On the Oracle Server dialog, click Delete.

4 Click Modify.

Modifying a Realm Definition

To change settings for an existing authentication realm:

1 In the Navigation frame, click Local Realms.

2 Click the realm name link on the Local Realms page.

3 Change the settings.

4 Click Modify.

Deleting a Realm

To completely remove an authentication realm:

1 In the Navigation frame, click Local Realms.

2 Click the next to the device name on the Local Realms page.

3 Click Delete.

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 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.

Using the "Self-Signed" Digital Certificates

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

The self-signed server certificates are in `/etc/opt/aaa/security/`. 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
- `root.cer` — CA for server 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.

Copy /etc/opt/aaa/security/root.cer to the CA storage on supplicants that enable server certificate checking.

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 /etc/opt/aaa/security/root.cer to the CA storage on the supplicant.

2 Copy /etc/opt/aaa/security/sampleclientcert.p12 to user the certificate storage on the supplicant:
   - The pass phrase for sampleclientcert.p12 is: 1234
   - The user name for sampleclientcert.p12 is: sampleclient@samplecompany.com

3 Configure a TLS realm for eap.realm 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 Start the RAD-Series Server that will be authenticating access requests.

2 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.

3 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/sa_key.pem`.

Note: Do not protect the key with a passphrase.

4 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.

5 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.

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).

Defining Digital Certificate Locations

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

1 Click Server Properties in the Navigation Tree.

2 Click Certificate Path Properties in the Main Screen.

3 Click each of the following links in the Workspace area, then enter the full path and click Create:

- Certificate Path: For TLS, TTLS, and PEAP. Full path to the RAD-Series Server certificate in `.pem` or `.cer` format.
- Private Key Path: Full path to the file in `.pem` or `.cer` format that contains the private key used to generate the RAD-Series Server certificate. This file cannot be encrypted.
- Certificate Authority Path: For TLS, TTLS, and PEAP. Full path to the CA certificate for the client certificate. Used by the RAD-Series Server to authenticate client certificates. The CA certificate for the client certificate must be in `.pem` format.
- Random Seed Path: For TLS, TTLS, and PEAP. Full path to the random seed used to generate keys.

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.

The following procedures modify the realm file you specified when configuring the realm.

For information on using LDAP storage, see "Using External Data Stores" on page 114.

User A-V Pairs

User profile information is stored as a set of 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.

The Server Manager creates standard RADIUS A-V pairs based on your selections and writes them in the realm file user entry. Some information, such as vendor-specific attributes, are entered on the Add User page Free tab using A-V pair syntax. All user A-V pairs, whether entered through the Server Manager GUI or directly into the file, represent a **one-to-one match** between the attribute and a specified value. You cannot specify lists of values for any single user attribute.

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

See "Configuration Files" on page 179 for user A-V pair syntax and a description of Interlink-specific user attributes. Standard RADIUS attributes are defined in the server's dictionary file, 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.

Adding User Profiles

To add new users to a realm file:

1 In the Navigation frame, click Local Realms for users stored in a realm file or click Users for users stored in the default users file.

2 If you clicked Local Realms then click the button next to the realm name.

3 Enter the User Name and click Create.

Note: User Name must be less than 64 characters and cannot contain &, ", ~, \, /, %, $, ', or spaces. Include only the *userid* portion of the NAI format user name except in the default users file which needs the full NAI.

---

4 Complete the authentication information on the General tab:

- Enter a Password and Confirm Password. The password cannot contain the \ character.
- To store passwords in hashed form, choose a Password Hashing Mechanism compatible with the user's authentication method.

Note: For PEAP, and TTLS, choose a password hashing mechanism compatible with the inner realm authentication method.

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

5 If you do not need to specify any other information for this user, click Create again.

Otherwise, follow the procedures in "Controlling and Provisioning Sessions" on page 67 to complete the user profile form, then click Create.

Controlling and Provisioning Sessions

Complete the Server Manager Users page tabs 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.

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 On the General tab, choose Login from Service Type drop-down menu.

3 Click the NAS/Login tab and complete the relevant Login fields.

Specify Framed Service Type

To configure dial-up users for framed service types:

1 Complete the steps to add a new user.

2 On the General tab, choose Framed from Service Type drop-down menu.

3 Click the Framed tab and select the Framed Protocol to use.

4 Optionally, complete any of the other Framed fields.

Set Timeout Values

1 Complete the steps to add a new user.

2 On the General tab, enter:
   - 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.

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 On the General tab, enter the 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 On the General tab, choose one of the Callback Service Type(s) from the drop-down menu.

2 Enter **either**:

Callback Number to dial exactly this string of numbers

or

Callback ID 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 Click the NAS/Login tab and enter the NAS IP address.

3 Optionally, enter:

- NAS Port — port number that must appear in an Access-Request for authorization to succeed
- NAS ID — NAS identifier that must appear in an Access-Request for authorization to succeed
- NAS Port Type — if NAS differentiates among its ports

Assign a Static IPv4 Address

1 Complete the steps to add a new user.

2 Click the Framed tab and enter the IPv4 address in Framed IP Address.

3 If the user is a router to a network, also enter 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 80.

1 Complete the steps to add a new user.

2 Click the Free tab and enter the A-V pair: `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 70.

1 Complete the steps to add a new user.

2 Click the Others tab.

3 To allow the user to only call **from** a specific phone number or machine, enter the number or MAC address in the Calling ID field.

To allow the user to only call **to** a specific phone number or access point, enter the number or MAC address in the Called ID field.

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 "Defining Authentication Realms" on page 53.

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

---

1 Complete the steps to add a new user.

2 Click the Free tab and in Reply text enter:

```
Terminate = 1
```

3 In Check text, enter:

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

Override Concurrent Session Limit

To do this procedure you must enable Session Tracking for the user's realm. See "Defining Authentication Realms" on page 53.

Note: Enabling Session Tracking sets the number of concurrent sessions to the global Simultaneous Use value set in Session Table 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 Click the Free tab and in Check text enter:

```
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 Click the Others tab and in Port Limit enter the maximum number of ports.

Deny Access

To deny a user access through a specific connection point:

1 Complete the steps to add a new user.

2 Click the Free tab and in Check text enter any or all of the following:

- `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 130 for information on using this attribute.

1 Complete the steps to add a new user.

2 Click the Free tab and in Check text enter the following:

- `Policy-Pointer = file.name`

NOTE: You need the Advanced Policy Engine license to implement an advanced policy.

Modifying a User Profile

1 In the Navigation frame, click Local Realms for users stored in a realm file or click Users for users stored in the default users file.

2 If you clicked Local Realms then click the button next to the realm name.

3 Click the button next to the User Name.

If you don't see the user listed, enter the User Name at the top of the page and click Search.

4 Change the entries on any of the five User tabs.

Note: To change the password hashing mechanism, first reenter and confirm the password.

---

5 Click Modify.

Deleting Users from Realm Files

To remove a user profile from the realm file:

1 In the Navigation frame, click Local Realms for users stored in a realm file or click Users for users stored in the default users file.

2 If you clicked Local Realms then click the button next to the realm name.

3 Click the button next to the user name on the User page.

4 Click Delete.

Defining Server Properties

Use the Server Properties page to change the default RAD-Series Server settings.

These procedures modify the server's `aaa.config` and/or `las.conf` files.

Some parameters/features cannot be handled via the Server Manager e.g. ipv6_enabled=Yes, radius_socket{}, default_ipv4_source_address=<ipaddress>, and default_ipv6_source_address=<ipaddress>, and if these parameters are needed then the aaa.config file must be manually edited.

Modifying Server Properties

1 In the Navigation frame, click Server Properties.

2 Click the link to the group of properties you wish to modify.

3 Change the values.

4 Click Modify.

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 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 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 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.

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 80.

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.

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. 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. If 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.

Max. 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.

Hold Accounting Requests

The number of seconds the RAD-Series Server holds an accounting request after replying to it. This hold time allows the server to capture duplicate requests. Default is 0. Specify a value that is twice the retransmission interval configured on the access devices. This value overrides the global value set with the Hold Replies parameter.

Max. 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.

Hold Authentication Requests

The number of seconds the RAD-Series Server holds an authentication request after replying to it. This hold time allows the server to capture duplicate requests. Default is 0. Specify a value that is twice the retransmission interval configured on the access devices. This value overrides the global value set with the Hold Replies parameter.

Max. 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.

Max. 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 server behavior.

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.

Maximum Log 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).

ProLDAP Properties

These properties specify how the server interacts with LDAP servers. They are global properties used for all LDAP servers.

TLS CA certificate directory

Used if SSL is enabled for the connection to the LDAP servers. 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.

TLS CA certificate file

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

TLS certificate file

The TLS RAD-Series Server certificate file specifies the file that contains the certificate that the RAD-Series 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 private key file

The TLS RAD-Series Server server private key file specifies the file that contains the private key that the RAD-Series 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.

OpenLDAP Debug

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 been enabled.

LDAP version

The version of the LDAP protocol to employ. The default value is 3. If your LDAP server requires version 2 of the protocol, enter that number here.

Timeout for TCP connect

Time in 1/10 seconds after which the RAD-Series Server stops waiting for a TCP connection to an LDAP server to complete. Default is 30 (3 seconds).

LDAP timeout

Time in seconds after which the RAD-Series Server stops sending a request to an LDAP server if it has not replied to the request. Default is 60.

Minimum wait before retry

Time in seconds the RAD-Series Server waits between requests to an LDAP server when there is no current connection for the data store. Default is 1.

Retry interval

In the event multiple LDAP servers are configured for a data store and at least one is connected, the interval in seconds at which the RAD-Series Server tries to connect to the remaining LDAP servers. Default is 60 seconds.

Session Table Properties

These properties determine how the RAD-Series Server records active session table information.

Session Hold Time

Time in seconds the RAD-Series Server waits before removing a session in the Stop state. Default is 45 seconds.

NOTE: This is **NOT** the time the RAD-Series Server waits for an Accounting-Start message. That time is 15 seconds plus the value of Token-Hold-Adjustment. After that time elapses without an Accounting-Start message, a session is moved into the Noconfirm state and does not count against a Simultaneous Use limit.

Session Kill Time

Time in seconds RAD-Series Server waits before completely removing a session in the Noconfirm, Disconnected, Rejected, or Collision state. Default is 300 (5 minutes).

Session Clear Time

Time in seconds the RAD-Series Server waits before removing a session in the Suspended state. Default is 1815 (30 minutes 15 seconds).

Session Check Time

Interval in seconds at which the RAD-Series Server checks for changes to the session table and writes them to the `session.las` file. Default is 300 (5 minutes).

Session Idle Time

Time in seconds the RAD-Series Server waits for a check point message before suspending a session. Default is 915 (15 minutes 15 seconds).

Session Update Time

Time in seconds between updates to the status of sessions. Default is 5.

Token Hold Adjustment

The time in seconds a token may be held after a session is accepted yet no confirmation is received after the request is released by the engine. A token may be held up to 15 seconds plus Token-Hold-Adjustment. Default is 5.

Auto Save Interval

Time in seconds between saving of the sessions to the session.las file by the RAD-Series Server. Default is 300 (5 minutes).

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 sets no limit. The value 0 prevents access to any user that does not have a specific value configured.

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. Default is 2147483647 (maximum allowed).

SNMP Properties

This property, Enable SNMP Support, turns on (Yes) or off (No) SNMP monitoring. The default is No.

For more information, see "Using SNMP" on page 79.

Tunneling Properties

This property, Tunneling Reply Items, specifies what the server should do with VPN tunneling attributes configured as reply items for a user entry if the Access-Request contains no tunneling hints. Make a selection from the drop-down list. For more information, see "Tunneling" on page 84.

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

1 In the Navigation frame, click Server Properties.

2 Click the SNMP Properties link.

3 Click Yes to Enable SNMP or No to Disable SNMP.

4 Click Modify.

MIB Objects

Information exchanged through SNMP is represented by objects in the Managed 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 objects that are sent to the management workstation by the server in response to 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 a 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 Stop and start the RAD-Series Server. See "Server Administration" on page 32 for instructions.

Configuring the RAD-Series Server for DHCP

1 In the Navigation frame, click Server Properties.

2 Click the DHCP Relay Properties link.

3 On the DHCP Relay screen, select Yes to Enable DHCP Support.

4 Enter either the:

DHCP Server Name — fully qualified domain name of the DHCP server

or

DHCP IP Address — IPv4 address of the DHCP server

5 If necessary, change any of the following defaulted values:

Property	Description
DHCP Server Port	The UDP port on the DHCP server to which DHCP requests are sent. Default is 67.
DHCP Relay Port	The UDP port on the RAD-Series Server at which DHCP responses are received. Default is 67.
Client Hardware Type	Value passed to the DHCP server to indicate hardware type. Options are: 0 (NONE) or 1 (ETHER). Default is 1.
Initial Retransmission Interval	Interval in seconds before the initial retransmission of a request to the DHCP server. Default is 4. The RAD-Series Server will double the retransmission interval for each subsequent retransmission.
Maximum Retransmission Interval	The maximum interval in seconds at which the RAD-Series Server retransmits DHCP requests. Default is 60.
Maximum Discover Retransmissions	The maximum number of retransmissions RAD-Series Server makes when acknowledging IP address assignments from the DHCP Server. No default is set.
Maximum Request Retransmissions	The maximum number of retransmissions RAD-Series Server makes when requesting IP address assignments from the DHCP Server. No default is set.

Property	Description
Maximum DHCP Message Length	The maximum size in bytes of messages that can be received from the DHCP server. No default is set.
Send Maximum DHCP Message Size Option	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. Default is No.
Send User Class Option	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. Default is No.

6 Click Modify.

Setting Up Realms for DHCP

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

1 In the Navigation frame, click Local Realms.

2 Click the realm name link.

3 Click Yes to turn on Session Tracking.

Note: Enabling Session Tracking sets the number of concurrent sessions to the global Simultaneous Use value set in Session Table 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.

4 Click Modify.

Defining Address Pools for Specific Users

If the user profile is stored in local storage:

1 In the Navigation frame, click Local Realms for users stored in a realm file or click Users for users stored in the default users file.

2 If you clicked Local Realms then click the button next to the realm name.

3 Click the button next to the User Name.

4 Click the Free tab on the Modify User page.

5 Enter the A-V pair: `Address-Pool=`*`name-of-pool`*

6 Click Modify.

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 tunneling attributes to the client device.

Tunneling Hints

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, the hints are returned to the client.

To specify how the server should handle requests that do **not** contain any tunneling hints:

1 In the Navigation frame, click Server Properties.

2 In the Workspace frame, click the link for Tunneling Properties.

3 Make a selection from the Tunneling Reply Items drop-down list:
   - Return-Configured-Tunnel-Attributes: Allow the return of tunnel attributes in the authentication reply.
   - Return-No-Tunnel-Attributes: Do not return any tunnel attributes in the authentication reply.
   - Reject-Access-Request: Fail the authentication by silently discarding the Access-Request.

4 Click Modify.

Or, you could manually configure the `NO_HINT` attribute in the aatv.Tunneling{} block of the `aaa.config` file.

Establishing a Tunnel for a User

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

1 In the Navigation frame, click Local Realms for users stored in a realm file or click Users for users stored in the default users file.

2 If you clicked Local Realms then click the button next to the realm name.

3 Click the button next to the User Name to display the User Profile page.

4 Click the Free tab and enter the tunneling attributes one per line in the Reply list box using standard A-V pair syntax:

attribute = value

5 Click Create or Modify:

You can also configure user tunnel attributes by adding them as reply items to the realm file, default users file or LDAP user profile. 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,
   . . .
```

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

aaaReply: *Tunneling-Attribute = Value*

For example:

```
aaaReply: Tunnel-Password = :0:Michigan
```

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	Description
Tunnel-Medium-Type tag-int	Indicates the transport medium to use when establishing the tunnel. Values are: • IPv4 • IPv6 • NSAP • HDLC (8-bit multidrop) • BBN-1822 (1822) • IEEE-802 (all 802 media plus Ethernet, "canonical format") • E-163 (POTS) • E-164 (SMDS, Frame Relay, ATM) • F-69 (Telex) • X-121 (X.25, Frame Relay) • IPX • Appletalk • DecnetIV • Banyan-Vines • E-164-NSAP
Tunnel-Type tag-int	Indicates the tunneling protocol to use when establishing the tunnel. Values are: • PPTP • L2F • L2TP • ATMP • VTP • AH • IP-IP-Encap • MIN-IP-IP • ESP • GRE • DVS • IP-IP • VLAN
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.

Attribute	Description
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 = :2:L2TP,
Tunnel-Medium-Type = :2:IPv4,
Tunnel-Client-Endpoint = :2:192.168.127.1,
Tunnel-Server-Endpoint = :2:192.170.130.1,
Tunnel-Password = :2:California,
Tunnel-Private-Group-ID = :2:engineering,
Tunnel-Assignment-ID = :2:management,
Tunnel-Preference = :2:2,
Tunnel-Client-Auth-ID = :2:NET,
Tunnel-Server-Auth-ID = :2:California
```

Reversing Configuration Changes

If you do not wish to keep changes you made to the temporary configuration files in Server Manager, you can revert to a previous copy by reloading the original configuration from the server, provided you haven't yet saved since loading.

Maintenance

This sections explains the AAA Server functions for:

- Reading the server log file
- Reading accounting logs
- Reporting on server authentication and accounting request statistics
- Reporting on active sessions
- Debugging the server

Reporting Server Log File

The AAA 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 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 through the Server Manager:

1 In the Server Status frame, select the server.

2 In the Navigation frame, click Server Logfile.

3 Under Search Parameters, enter any filters to limit the messages displayed.
   - Begin/End — date and time range during which event occurred
   - Number of Records — the maximum number of records to show; may be any number from 1 to 999
   - User — user name in request; may be just userid (e.g.: fred) or full NAI format name (e.g.: fred@abc.com)

4 Under Message Type, select (Yes) all the types you want displayed and deselect (No) any you do not want.

Message Type	Description
Failure	Messages indicating a server internal error or a serious problem with the configuration files. 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	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.
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. Notification messages, flagged as "(N)" in the logfile, are indications of noteworthy events, such as the receipt of a HUP signal by the server. By default, these types are not displayed.
Server start	Messages generated during each server startup or restart.
Server stop	Messages generated when the administrator shuts down the server.
Authentication request (grey)	Access-Request message received.
Authentication failure (red)	Access-Reject message sent.
Authentication success (green)	Access-Accept message sent.
Accounting response (blue)	Accounting-Request response message sent.

5 Click Search to display the log file results.

You'll see the Type, Time, and Description for each message displayed in the Message frame.

Reporting User Accounting Records

The AAA 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 92 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 96.

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 in Server Manager:

1 In the Server Status frame, select the server.

2 In the Navigation frame, click Accounting.

3 Under Search Parameters, enter any filters to limit the records displayed:

- Begin/End — date and time range during which session occurred
- Number of Records — the maximum number of records to show; may be any number from 1 to 999
- User — user name; may be just userid (e.g.: fred) or full NAI format name (e.g.: fred@abc.com)
- Realm — realm name; if no User specified, pulls all records from this realm that match the other criteria

---

Note: Enter NULL to display NULL realm records. Don't leave the field blank.

- NAS — network access server from which session originated; may use NAS-Identifier, NAS-IP-Address or NAS-IPv6-Address

---

Note: This parameter will find a leading substring or an exact match. For example, a value of t14 would match records with NAS/Port field t14.interlinknetworks.com/1234)

---

4 Click Search. The selected accounting records appear in the Workspace frame.

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 LAS-Code 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.
LAS-Code (keyword)	Reason why record was logged. Codes are: • LAS-Normal 0 • LAS-Reject 1 • LAS-Cancel 2 • LAS-Noconfirm 3 • LAS-Overtime 4 • LAS-Unknown 5 • LAS-Notlocal 7 • LAS-Suspend 8 • LAS-Failed 9 • LAS-Authorized 10 • LAS-NASreboot 11 • LAS-Remote 12 • LAS-Duplicate 13 • LAS-Collision 14 • LAS-Stop 15
LCL-Time (r-epoch)	Time when record was logged, relative to LAS-Start.
LAS-Duration (integer)	Duration of session (in seconds) according to the LAS.

Attribute	Description
LCL-Duration (integer)	Duration of session (in seconds) according to RADIUS huntgroup server for NAS.
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 set the number of concurrent sessions for the realm to whatever value you have defined for the Simultaneous-Use parameter of las.conf.
Session-ID (string)	Session ID (is unique to each session).
Token (string)	Token pool name. Currently always NA.
Session-Timeout (integer)	Session Time limit.
NAS/Port (special)	NAS-Identifier "/" NAS-Port-Number.
Service-Class (string)	Merit charging/access-ctl service class.
Filter-ID (string)	Filter name associated with session.
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 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 Accounting Extensions section of 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` file. You may extend the VSA list by adding the vendor to the `vendors` file and the attributes and their values to the `dictionary` file.

Example Accounting Record

```
1079387320 LAS-Stop 7 7 NA t20011@iln.com t20011@iln.com '4054b8.1.t15'
NA NA t25.iln.com/2001 NA NA NA
##      User-Name:0='t20011@iln.com'
##      NAS-IP-Address:2=192.168.3.25
##      Acct-Authentic:1=RADIUS
##      Acct-Session-Id:0='t04-1-4d8133d5'
##      Acct-Status-Type:1=Stop
##      Interlink-Proxy-Action:0='ACCT'
##      Microsoft-MS-CHAP-Error:6='Invalid entry'
##      Microsoft-MS-Primary-DNS-Server:2=192.168.1.2
##      Acct-Delay-Time:1=0
```

Modifying the Accounting Log

Certain features of the standard RAD-Series 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 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 Stop and start the server.

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 Stop and start the server.

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 are automatically compressed when they rollover.

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 Stop and start the server.

Modifying Accounting Logging Behavior

You can change server 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 Stop and start the server.

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 48 to change the .fsm settings.

Reporting Server Statistics

Use the Server Manager Statistics page to retrieve a count of selected events that occurred on the AAA Server within a given time period. Statistics are displayed using a bar graph. You'll see the total number of:

- Server starts
- Server stops
- Accounting-Requests
- Authentication-Requests
- Authentication-Successes
- Authentication-Failures

1 In the Server Status frame, select the server.

2 Navigation frame, click Statistics.

3 Under Search Parameters, enter the Begin and End date and time of the period for which to retrieve statistics.

4 Click Search.

Reporting Active Server Sessions

Once the server receives an Accounting-Request-Start message, it begins to store session information in an active session table record. Only after the server receives the Accounting-Request-Stop message from the access device, indicating the user's session is terminated, does it clear its active session table record and write the session information to the accounting log file.

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

Viewing Active Sessions

1 In the Server Status frame, select the server.

2 In Navigation frame, click Sessions.

3 Enter the Session Filtering parameters.

You can enter either or both values. Only sessions matching these values will be retrieved.

4 Click Display.

The Workspace display changes to show a list of active sessions.

5 Click a user session link to display the session attributes.

6 Click OK to return to the active session list.

Stopping Active Sessions

To clear sessions that were terminated on the access device but are maintained as active by the AAA Server:

1 Follow the procedure Viewing Active Sessions.

2 Click Stop.

The AAA Server will clear the record from its active session table, but no action is taken by the access device.

Note: You can only stop a session on a server that contains a self-referring client entry. When you install an AAA Server, it automatically adds this entry (localhost) to the server's Proxies configuration.

Server Programs

radiusd

`radiusd` is the main 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 the server is first started. You can configure which finite state machine the 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 from the Server Manager console, the command line or through an inetd service.

Synopsis

```
radiusd  -c <Working-directory> -p <Authentication-port>
-pp <Authentication-relay-port> -q <Accounting-port>
-qq <Accounting-relay-port> -da <AATV-directory>
-d <Config-directory> -dl Logfile-directory> -l <Log-format>
-di <IPC-directory> -dm <Accounting-directory> -a <Livingston-
directory> -dd <Data-directory> -dr <Run-directory> -f <FSM>
-g {syslog|logfile|stderr} -h -n -s -x -v -z
```

Options

You can start radiusd with any of the following options to override built-in defaults. Server startup options can also be changed in the Server Manager.

Option	Description
-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.
-p <*Authentication-port*>	Default port number to listen for authentication requests on. See "Managing RADIUS Listen Sockets" on page 242 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 242 for a complete description.

Option	Description
-qq <*Accounting-relay-port*>	Default port number to relay accounting requests to.
-da <*AATV-directory*>	Location of binary AATVs, if other than installation default.
-d <*Config-directory*>	Location of configuration files.
-dl <*Logfile-directory*>	Server log file directory. Default is /var/opt/aaa/logs.
-l <*Logfile-name*>	Server log file name. Default is logfile.%Y%m%d.
-di <*IPC-directory*>	Location of shared memory operation files.
-dm <*Accounting-directory*>	Location of accounting session records in MERIT format.
-a <*Livingston-directory*>	Location of Livingston Call Detail Records (when used as an alternative to the default MERIT format).
-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.
-dr <*Run-directory*>	Location of the file where the server's process ID is stored.
-f <*FSM*>	Alternate FSM file to load upon startup, instead of the default radius.fsm file.
-g {logfile|syslog|stderr}	Where server events are logged. Enter option plus logfile (server logfile), syslog (system logfile), or stderr.
-h	Display the help syntax.
-n	Start new session table for LAS upon server startup.
-s	Run in single process (non-spawning) mode.
-v	Display radiusd version information.
-x	Add to debug flag value [0 to 4 levels of debugging are allowed]. These switches must follow all other switches.
-z	Empty (zap) the log file & debug file.

radiusd.sh

`radiusd.sh` is a script that can be used to start the AAA Server at boot time.

Synopsis

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

Arguments

The 'start' argument will start the AAA Server if it is not already running.

The 'stop' argument will stop the AAA Server if it is running.

The 'restart' argument will stop the AAA Server if it is running and then start it back up.

The 'reload' argument will HUP the AAA Server if it is running so that it can re-read its configuration files.

The 'status' argument will report the current status of the AAA Server.

radcheck

`radcheck` determines whether a given AAA Server is operational. `radcheck` may be invoked from any host, not just those with a self-referring clients entry, although more information is returned for those so registered.

Synopsis

radcheck -d <*Directory*> -h -ipv6 <on/off> -p <*UDP-port*> -r <*Retries*> -t <*Timeout*> -v -x -0 ***<ServerName>***

Arguments

<*ServerName*> IP address or fully-qualified domain name of a machine running a AAA Server.

Options

Option	Description
-d <*Directory*>	An alternate directory containing clients and dictionary 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 help syntax.
-ipv6 <*on/off*>	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.
-p <*UDP-port*>	UDP port on which the 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.
-x	Add to debug flag value.
-0	Suppress copyright message in the output.

See "Reporting Server Status" on page 35 for a description of radcheck output.

radpwtst

radpwtst is a test client program. It generates RADIUS packets to perform a password authentication on a specified user using the current 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."

Synopsis

```
radpwtst -a <ACKs> -c <Code> -d <Directory> -f <File> -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> -0
<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 NULL.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*>	An alternate 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.
-f <*File*>	Input file of A-V pairs to send with radpwtst.
-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. 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 instead of the default of 1.
-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.
-secret <*secret*>	Shared secret (no default). If not specified, radpwtst will look in the clients file for the shared secret.

Option	Description
-t <*Timeout*>	An alternate timeout value (in seconds) instead of the default of 3.
-u <*Service-Type*>	One of several *Service-Type* values instead of 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-p*assword*>	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.
-0	Suppress copyright message in the output.

Stopping Sessions

You can clear sessions with radpwtst. To clear a session, you must know its `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 <*Async-Port*> **-:Acct-Status-Type=Stop -:Class="<*class-value*>"**

radsignal

The `radsignal` utility performs 3 functions:

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

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 AAA 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.
-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 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.20110304`' then it will be renamed to '`logfile_part01.20110304`' and the current logfile will be called '`logfile_part02.20110304`'. 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.2011-03-04.log`' then it will be renamed to '`session.2011-03-04_part01.log`' and the current accounting logfile will be called '`session.2011-03-04_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.

radrecord

The `radrecord` utility reads Interlink (MERIT-style) accounting log files and prints out specific session information to standard output. This utility will not read Livingston Call Detail Records. Using various command line options, it is possible to create flexible and powerful search criteria to obtain information from the logs.

Note: If any session log file is compressed with either gzip (1) or compress (1), radrecord will try to use the corresponding decompression program to read the file.

You can also read the accounting logs through the Server Manager Accounting page.

Synopsis

```
radrecord -a <After> -b <Before> -c -d <Directory> -e <NA-equ>
-f <Filter> -h -i <Session-ID> -l <Log-config> -n <NAS/port> -o <Time>
-p <Protocol> -r <Reason>:<Reason> -s <Service-Class> -t <tokenpool>
-u <User-ID> -v <version> -x -y <type> -:<attribute=value> -0
-1 <filename> <Accounting-logfile> <...>
```

Arguments

<*Accounting-logfile*> One or more accounting log files to search.

Options

-a <*After*>	The date and time of the session to begin retrieving data from. Takes a string in the format: %Y-%m-%d/%H:%M:%S. For example, 2011-12-01/15:01:30 means 3:01:30pm, December 1, 2001. Hour, minute, and second may be omitted and each default to 0.
-b <*Before*>	The date and time of the last session to retrieve data from. Takes a string in the format: %Y-%m-%d/%H:%M:%S. For example, 2011-02-01/15:01:30 means 3:01:30pm, February 1, 2011. Hour, minute, and second may be omitted and each default to 0.
-c	Match chargeable records only.
-d <*Directory*>	Alternate directory name containing the AAA Server authfile, clients, and .users files, instead of the default /etc/opt/aaa directory. If no -d option is given, radrecord will look for the default location. An error will be displayed if the configuration files cannot be found.
-e <*NA-equ*>	Assign a value for output to replace N/A values in record.
-f <*Filter*>	Only searches for sessions with a Filter-ID matching the given regular expression.
-h	Displays the radrecord syntax.
-i <*Session-ID*>	Only searches for sessions with the given session ID.
-l <*Log-config*>	Outputs any session log in the format and to the file configured in the specified configuration file for the LOG AATV (defaults to log.config).
-n <*NAS/port*>	Only searches for sessions with the given NAS ID and (optionally) port.
-o <*Time*>	Specify a time in '%Y-%m-%d/%H:%M:%S' format, i.e., 2003-01-17/12:49:03.
-p <*Protocol*>	Only searches for sessions that used the specified protocol.
-r <*Reason*>:<*Reason*>	Only searches for sessions with a given reason code or within a specified range. A reason code may either be a integer or a text name as listed in the LAS-Code value section of the dictionary file. e.g. '-r 15' or '-r 0:15' or '-r Stop'
-s <*Service-Class*>	Only searches for sessions with a service class name matching the specified service class regular expression.

-t <*Token-pool*>	Only searches for sessions that match the token pool name regular expression.
-u <*User-ID*>	Only searches for sessions that match the user ID regular expression.
-x	Add to debug flag value. May occur 0 to 4 times. Each additional -x will increase the amount of debug output.
-y <*Service-Type*>	Only searches for sessions with a service type name matching the specified service type regular expression.
-:<*attribute=value*>	Specify an A-V pairs to match with each session record. This may be used multiple times if needed.
-0	Suppress copyright message in the output.
-1 <*filename*>	Find only first session of each user listed in the specified file. You must create and save the file with a list of users.

Troubleshooting

Debugging

The 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 33 and "radsignal" on page 106 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
Request from unknown client.	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.
LAS_ACCT: user realm '*xxx.xxx*' not configured	Realm is not configured in the las.conf file and you're using the LAS feature for authorization policies.	Add the realm to the las.conf file.
read_clients: Missing required client '*xxx*' : line *y*	Entry on line *y* of the clients file is missing the required parameter '*xxx*'.	Add the omitted parameter to the clients file entry.
Couldn't get our own IP address(es)	No entry for the AAA Server in your DNS database.	Add both backward and forward entries for AAA Server in DNS database.

Message	Probable Cause	Possible Solution
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.
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.
Vendor *x* attribute *y* unknown	Server received an attribute that is not defined in the dictionary.	Add vendor-specific attributes to the dictionary file.

radiusd Error Messages

These messages are returned by the server if `radiusd` fails to start. They appear in the Message frame at the bottom of the Server Manager Administration page or on your Terminal screen.

Message	Probable Cause	Solution
Invalid port number	Port number is a non-numerical value	Specify a numerical value in Start Options.
Invalid timeout value	Port number is a non-numerical value	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.
setup_logfile: 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.
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.
gethostname() returned no or incorrect DNS	NAS server's DNS entry is configured incorrectly.	Check DNS configurations for both forward and reverse entries.
Received invalid reply digest from server	Server and client do not agree on shared secret.	Verify that the shared secret in the clients file agrees with the secret configured on the client.

Server Reply Messages

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

Message	Probable Cause	Possible Solution
Access controlled	Host name user wants to access requires authentication.	
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 AAA 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
- Oracle databases

Using an LDAP Server

If you only need to retrieve user IDs and passwords from the LDAP server, just follow the procedure "Identifying LDAP Directories" on page 54.

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 iPlanet 5.0.

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 Follow the steps in Modifying a Directory Configuration to display the LDAP Directory dialog.

2 Click Yes to Use SSL, then click Save.

3 Click Modify.

4 Click Server Properties > ProLDAP Properties.

5 Enter the path to the AAA Server TLS CA certificate directory.

or

Enter the name of the AAA Server TLS CA certificate file.

6 Enter the TLS certificate file if required by the LDAP server.

7 Enter the TLS private key file if required by the LDAP server.

8 Click Modify.

9 If the AAA Server is already running:
- Click Save Configuration and select the server.
- Stop and restart the server.

The AAA 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`

If you're using Solaris 8, you can add `/dev/urandom` and `/dev/random` devices by installing patch 112438 (Sparc), which are available via the Patchfinder at http://sunsolve.sun.com. Starting with Solaris 9 these devices are included by default.

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 AAA Server can receive passwords from the LDAP server in clear text format or encrypted by SSHAMD5, SHA1, x-NT hash, x-lm hash and UNIX-crypt. If passwords are encrypted by other methods, configure the AAA 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 server includes the `55iaaa-radius.ldif` file as an example of a file that has been modified for the iPlanet 5.0 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 Attribute Types

The following Interlink 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 AAA Server's dictionary file 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 AAA 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
```

Known Issues with ProLDAP™

These are a few issues to be aware of when configuring the server to use ProLDAP.

Failover not operating properly

If the LDAP server process terminates in such a way that the TCP connection remains active, the AAA Server will not recognize that the LDAP server is not accessible. Instead of rotating to the next LDAP server, the AAA Server will retransmit the request until it times out.

Authentications time out on LDAP restart

If the LDAP server is started (or restarted) while the AAA Server is running, one or two authentication requests may timeout during the LDAP restart. Starting the LDAP server before starting the AAA Server will prevent this issue from occurring.

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 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.

Lost LDAP-Response Messages

If the LDAP server is installed on the same host as the AAA Server or if it is an especially fast LDAP server, some of the LDAP-response messages may be lost by the AAA Server and authentication requests will timeout.

Using Oracle® Database

The AAA Oracle module allows you to retrieve user profiles from an Oracle database. When dealing with a large number of users, an Oracle data store delivers much higher server performance than File or Unix-PW storage. It also supports the server's load balancing feature.

Oracle can be configured to store:

- User name and password
- A limited set of reply item attributes

Its current limitations are:

- No check or deny item attributes
- No logging of Accounting-Requests to database, only to text file

You can store users in an Oracle database on one platform and authenticate those users with an AAA Server installed on another platform, such as the Red Hat Linux operating system.

To set up realms to use an Oracle data store, follow the procedure "Identifying Oracle Servers" on page 58.

Requirements

To implement an Oracle data store, you must first purchase and install:

- Red Hat Enterprise Linux or Sun Solaris operating systems. Only these platforms support the `db_srv` daemon required to communicate with Oracle.
- Oracle development package. This includes the libraries required by the `db_srv` daemon.

Set up your environment to enable the Oracle database and `db_srv` daemon.

Oracle Data Store Processing

When the AAA Server receives a request from a user in a realm that has been configured for Oracle storage, the Finite State Machine (FSM) calls the iaaaRealm action, which in turn calls the Oracle action to retrieve user profiles from the Oracle database.

When the Oracle action receives a request from iaaaRealm. it uses the `AUTH_NET_REQ` data structure to send the request to the `db_srv` daemon. The daemon acts as an Oracle listener, waiting to receive requests from the Oracle action. `db_srv` daemon sends a SQL query to the Oracle database and uses the `AUTH_NET_USERSAUTH_NET_USER` data structure to return the appropriate replies to the Oracle action. The Oracle action returns the results to the FSM.

Multiple `db_srv` daemons can communicate with a single database or a set of replicated databases. Each daemon must be on a different machine or listening to a different port.

The Oracle module also provides round-robin load balancing and failover of Oracle `db_srv` daemons.

Storing User Profiles in Oracle

The AAA Server authenticates users against the values stored in the `network_auth_name` and `network_auth_password` columns of the `AUTH_NET_USERS` table. The rest of the table stores RADIUS reply items returned to the server through the `db_srv` daemon.

Users Table Structure

Define a users table with the following SQL statements:

```
create table AUTH_NET_USERS
(
network_auth_name VARCHAR2(63),
network_auth_password VARCHAR2(128),
session_timeout number (10),
idle_timeout number(10),
port_limit number(10),
tunnel_type number(10),
tunnel_medium_type number(10),
tunnel_client_end VARCHAR2(64),
tunnel_server_end VARCHAR2(64),
acct_tunnel_connection VARCHAR2(64),
service_type number(10),
framed_protocol number(10),
framed_ip_addr number(10),
framed_ip_netmask number(10),
framed_routing number(10),
filter_id VARCHAR(128),
framed_compression number(10)
);

COMMIT;
```

The columns in the AUTH_NET_USERS table correspond to the following RADIUS attributes. Store the A-V pair value in the corresponding column.

Column	RADIUS Attribute
network_auth_name	User-Name
network_auth_password	Password
network_timeout	Session-Timeout
idle_timeout	Idle-Timeout
port_limit	Port-Limit
tunnel_type	Tunnel-Type
tunnel_medium_type	Tunnel-Medium-Type
tunnel_client_end	Tunnel-Client-End
tunnel_server_end	Tunnel-Server-End
connection_id	Connection-Id
service_type	Service-Type
framed_protocol	Framed-Protocol
framed_ip_address	Framed-IP-Address
framed_ip_netmask	Framed-IP-Netmask
framed_routing	Framed-Routing
filter_id	Filter-Id
framed_compression	Framed-Compression

Create the Users Table

To execute the table statement:

1 Start an instance of the database where user profiles will be stored.

2 Start SQL*Plus and connect to the database.

3 At the SQL prompt, enter `START create.sql` to create the users table.

4 Add user records to the table with the following SQL command:

```
insert into AUTH_NET_USERS values ('User-Name', 'Password', Session-
Timeout, Idle-Timeout, Idle-Timeout, Port-Limit, Tunnel-Type,
'Tunnel-Client-End', 'Tunnel-Server-End', 'Acct-Tunnel-Connection',
Service-Type, Framed-Protocol, Framed-IP-Address, Framed-IP-Netmask,
Framed-Routing,'Filter-Id', Framed-Compression);

commit;
```

Replace the variables in the example above with the attribute values for this user. Enclose string type values in quotes. You can write a NULL value to any attribute, except for `User-Name` and `Password`.

Note: You can add all users at once by modifying the fill.sql command file included with db_srv, then running START fill.sql from the SQL prompt.

Always create the table:

- In the database's default SYSTEM table space
- In the default database or a database that has been assigned an SID

Modify the Users Table

To remove users from the table, at the SQL prompt, enter:

```
delete * from AUTH_NET_USERS where network_auth_name = 'User-Name';

commit;
```

Substitute the `network_auth_name` column value for the *'User-Name'* placeholder.

Note: You can use the clear.sql command file included with db_srv to clear all users from the table. Run START clear.sql from the SQL prompt.

Whenever modifying the SQL statement or `create.sql` script that creates the users table:

- Always name the table `AUTH_NET_USERS`
- Do not add, remove, modify, or change the order of the predefined columns

Configuring db_srv Daemon

The `db_srv` daemon is the client that interfaces with the Oracle database and AAA Server. Run a daemon on each Oracle database host machine you wish to access (one `db_srv` per AAA connection).

Install db_srv Daemon

The db_srv Daemon is installed with the AAA Server. If the Oracle database is also hosted on this machine, you can choose Oracle server as a component when you install the AAA Server software. If not, run the installer on the machine that hosts the Oracle database and choose to install only the Oracle server component.

To install the daemon:

1 Copy the AAA installer from the download directory to the Oracle host machine.

2 Log on as the user permitted to communicate with the Oracle database.

3 Run the installer:

```
# sh /local directory path/RAD-Series.8.0.0.solaris.sun4u.bin
```

4 Enter option number 4 for Oracle:

```
ENTER A COMMA-SEPARATED LIST OF NUMBERS REPRESENTING THE COMPONENTS
TO BE INSTALLED: 4
```

5 Follow the prompts to choose the directories where you want the files installed.

Edit db_srv.opt

Before executing the daemon, edit the configuration file, `db_srv.opt` (in `/etc/opt/aaa` by default):

1 Locate the lines:

```
DB_SRV_PORT=Port
DB_SRV_ORA_UID=User-Name
DB_SRV_ORA_PWD=Password
DB_SRV_ORA_SID=Oracle-SID
export DB_SRV_PORT DB_SRV_ORA_UID DB_SRV_ORA_PWD DB_SRV_ORA_SID
```

2 Enter values for:

Parameter	Description
DB_SRV_PORT	Port number used by db_srv for incoming authentication requests from the AAA Server. Should match port you configure on the AAA Server. Any available port number may be used. Typically, port numbers greater than 4000 are used, since port numbers for standard services are usually less than 4000. If multiple db_srv daemons are running on the same machine, each daemon must be listening to a different port.
DB_SRV_ORA_UID	Oracle user name used to access the database.
DB_SRV_ORA_PWD	Oracle password used to access the database.
DB_SRV_ORA_SID	Oracle ID for the database to connect to when more than one database exists on the machine. If the parameter is omitted, the daemon connects to the default database, which is defined during database installation.

Start db_srv Daemon

To execute the daemon, at the command line, run:

```
run_db_srv.sh
```

If `db_srv.opt` is not installed in the default location, use the `-f` *`Directory`* option where *`Directory`* is the full path to the configuration file.

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. There is a new additional format for Decision files now. The old format contains policy group entries which are still supported but it is not documented here. If you are using the group format you can to 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 234 for more information on FSMs.

Implementing Advanced Policies

The default FSM file distributed with the AAA 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

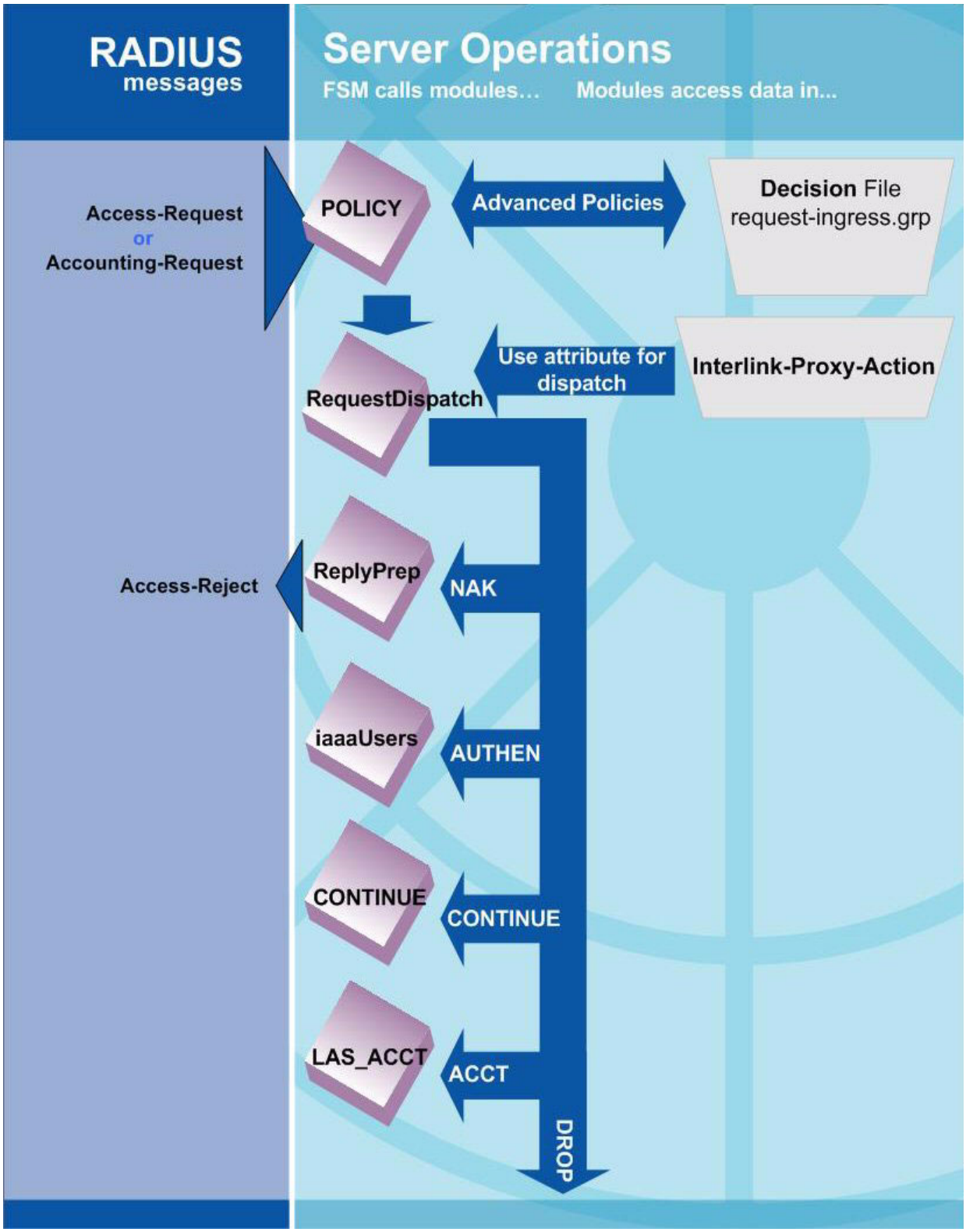
The `radius.fsm` file can also be modified to implement advanced policies in decision files at additional points in the FSM file. See "Calling Decision Files" on page 174 for further details on modifying the FSM file to implement advanced policies

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 70 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.

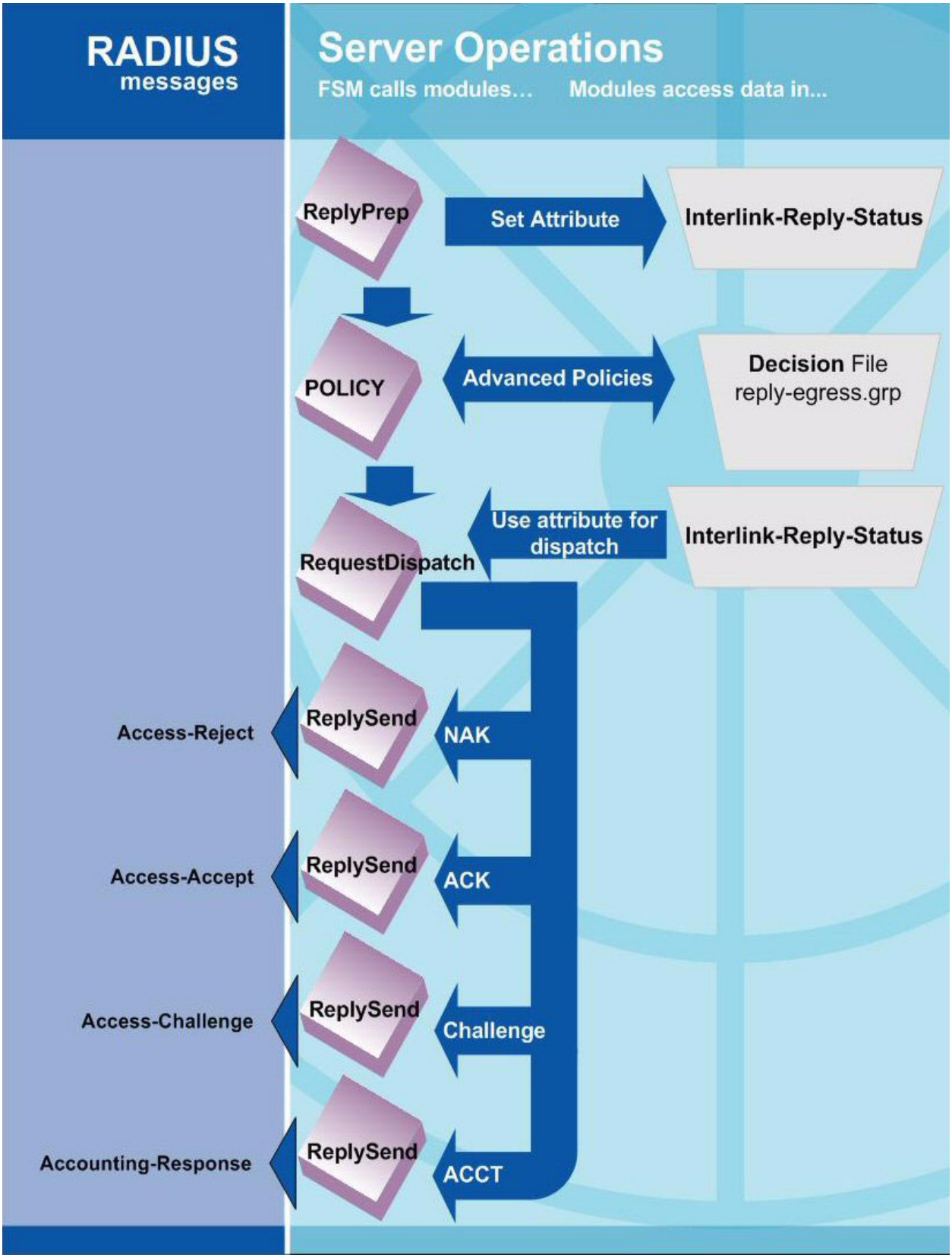
Reply post-processing policy

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 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

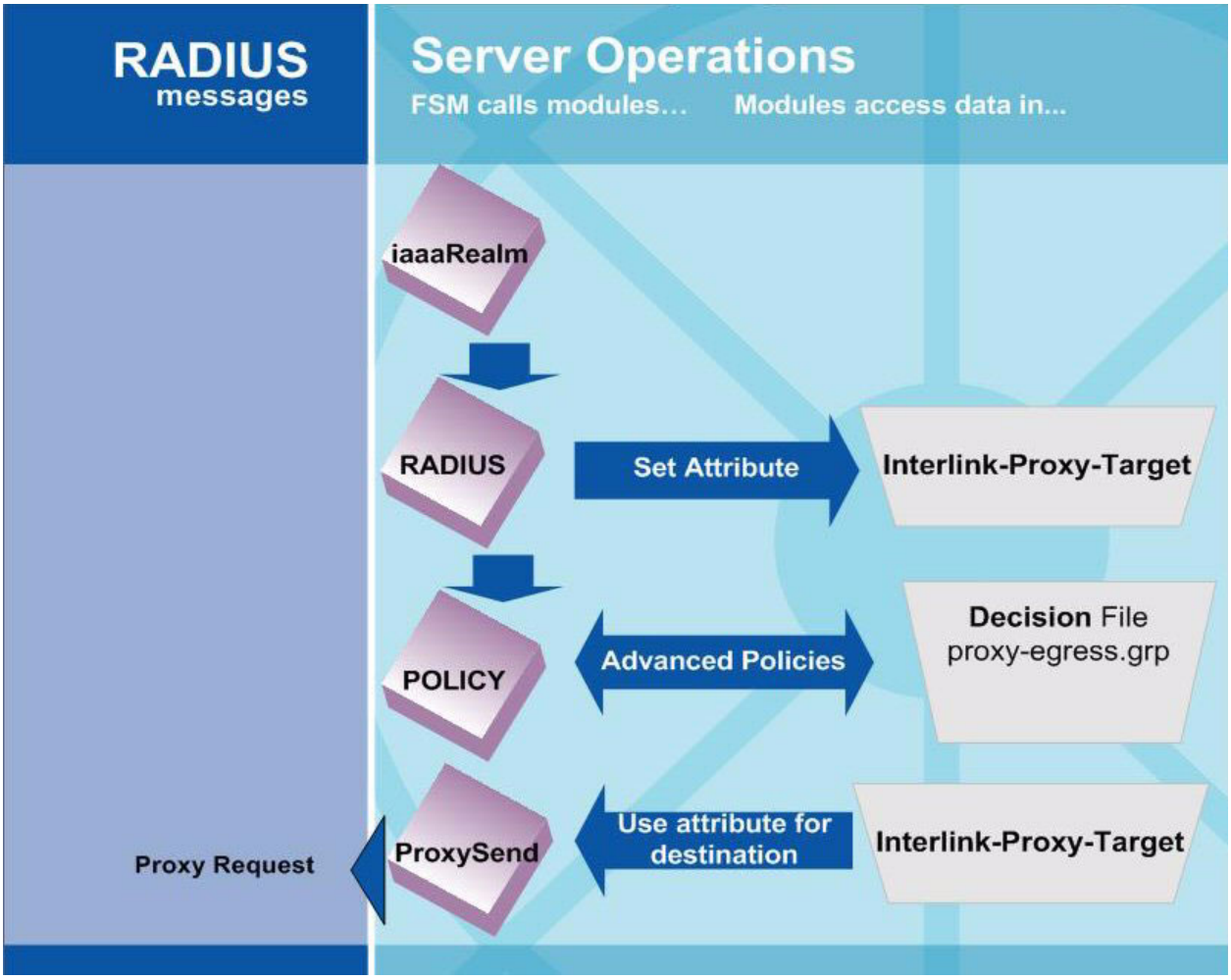
Proxy send request policy

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.



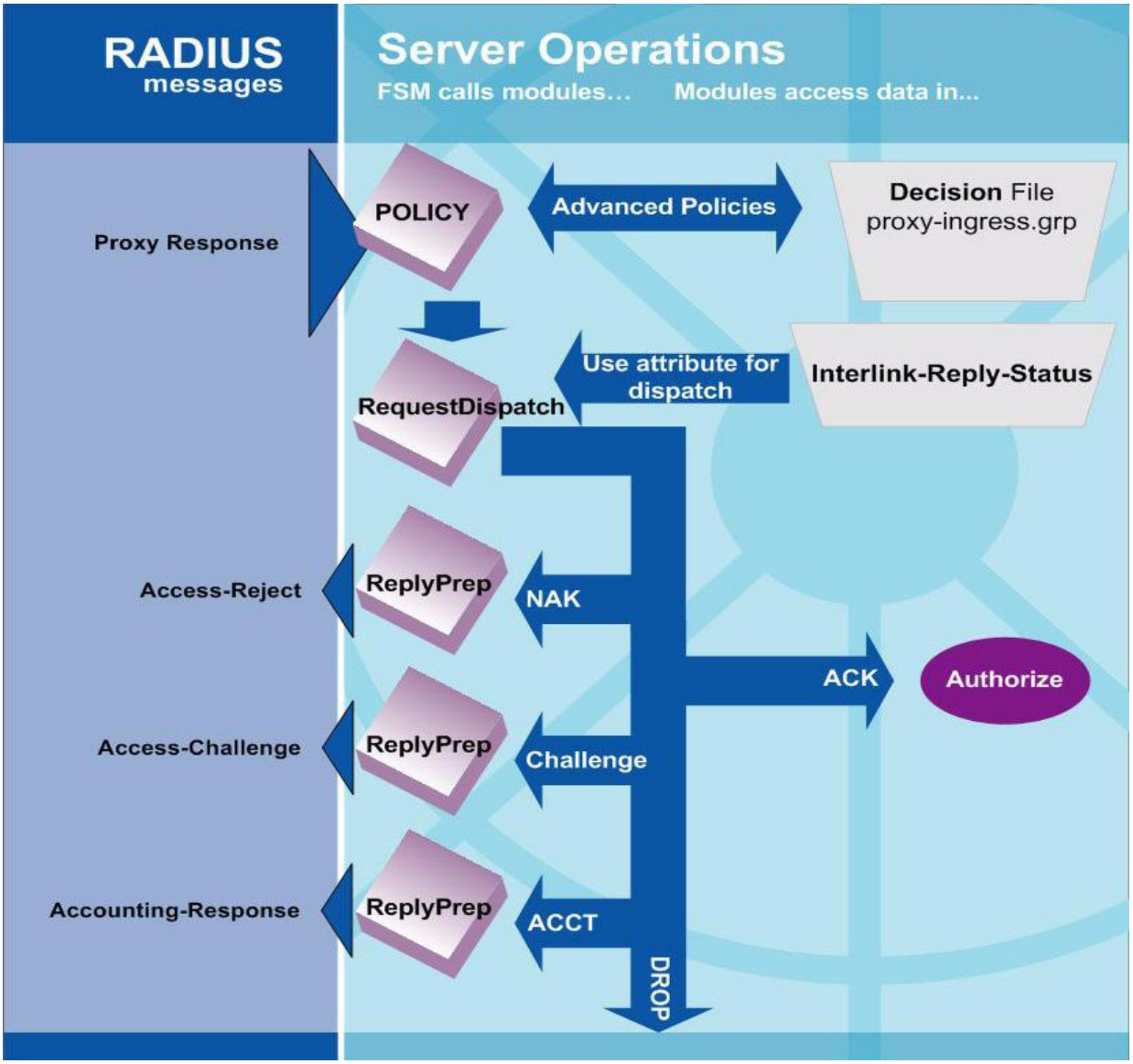
Flow of Proxy Send Request Policy

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 153 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 171 for details. The types of ***<attr-spec>*** and ***<value-spec>*** must be compatible. See "Type Compatibility" on page 156 for details.

Command	Parameter Description
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 171 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 171 for more details. The value types for ***<attr-spec>*** and ***<value-spec>*** must be compatible. See "Type Compatibility" on page 156 for details.
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 236 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: NOTICE, ERROR, CRITICAL, ALERT, WARNING, 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. 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 "last".
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 168 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 234 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 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:

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

Vendor Names

When more than one vendor has defined an attribute with the same name, it is necessary to distinguish between them. This is accomplished by prefixing the attribute name with the vendor name, offset by a ':' (colon) character.

For example:

```
RADIUS:Reply-Message
Cisco:Cisco-AvPair
MERIT:Xvalue
```

No white space is allowed around the ':' character.

For example:

```
RADIUS : Reply-Message
RADIUS: Reply-Message
RADIUS :Reply-Message
```

Are NOT valid. Using these forms will result in a syntax-error load-time error. See "Error Handling" on page 171 for more details.

Vendor names are defined in the server's vendors file.
Vendor names are case-insensitive (`MERIT` is the same as `Merit`).

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 server's dictionary file.
Attribute names are case-insensitive (`Reply-Message` is the same as `REPLY-MESSAGE`).

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 141 for more information.

Value

See "Value Specifications" on page 141 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]
```

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 139). 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 134 for details and examples.

  Using this keyword in other places results in an invalid-instance-specification load-time error. See "Error Handling" on page 171 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"
  ```

  Then:

  ```
  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 171 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 167 for details.

Value Specifications

Constant values may be specified in various ways depending on the value type. See "Type Compatibility" on page 156 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, short, octet 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 a short attribute can have a value range of 0 to 65535.
Integer values used with an octet 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 171 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 server's dictionary file.
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 171 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 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. 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 171 for more details.

5. 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 171 for more details.

6. InterfaceId values

The InterfaceId values are enclosed in double quotes '"'.
The InterfaceId values must be specified in "hhhh:hhhh:hhhh:hhhh" standard notation.

7. Date values

There is no provision for specifying date value constants at this time. The value of date type attributes may be compared and copied. It is only date value constants that are not supported.

The use of a constant value in conjunction with a date type attribute results in a syntax-error load-time error. See "Error Handling" on page 171 for more details.

8. Tag-Str values

Attributes of type Tag-Str use string values. See String values (item 3 above) for details.

9. 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.

10. Octets values

Attributes of type octets use string values. See String values (item 3 above) for details.

11. Octet values

Attributes of type octet use integer values. See Integer values (item 1 above) for details.

12. Short values

Attributes of type short 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 RADIUS 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 RADIUS 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 146
!=	Not equal to, see "!= Operator" on page 147
>	Greater than, see "> Operator" on page 150
<	Less than, see "< Operator" on page 148
>=	Greater than or equal to, see ">= Operator" on page 151
<=	Less than or equal to, see "<= Operator" on page 149
&&	Logical AND, see "&& Operator" on page 153
||	Logical OR, see "|| Operator" on page 153
!	Logical NOT, see "! Operator" on page 154
()	Parentheses, see "() Operator" on page 154

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 156 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 144.

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 171 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 145 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 145 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 145 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 145 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 145 for details.

3. Operation

Compares two values for less than:

false: ***<value1>*** not less than
true: ***<value1>*** less than ***<value2>***

See Operation in "General comparison operator" on page 145 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 145 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 145 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 145 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 145 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 145 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 145 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
short
octet

String-value attribute types:
string
tag-str
octets

Date-value attribute types:
date

IPv4-address-value attribute types:
ipaddr

IPv6-address-value attribute types:
ipv6addr

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 171 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 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 153.

<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 144 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 137. 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 137. 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 171 for details.

   The types of <attr-spec> and <value-spec> must be compatible. See "Type Compatibility" on page 156 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 144 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 137. 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 171 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 171 for more details.

   The value types for ***<attr-spec>*** and ***<value-spec>*** must be compatible. See "Type Compatibility" on page 156 for details.

3. Operation

   Modifies ***<attr-spec>*** to have the value of ***<value-spec>***. See "Tagged Attributes and the Advanced Policy Language" on page 144 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 236 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 172 for more details.

The use of an undefined event name results in an undefined-event load-time error. See "Error Handling" on page 171 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:
   FATAL, ERROR, CRITICAL, ALERT, WARNING, 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 137. 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 137.
     The default instance for ***<attr-spec>*** is '**last**'.

  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 137.
     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 144 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 171 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 137.
     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 144 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 171 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 137.
   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 144 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 171 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 137.
   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 144 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 171 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 137.
     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 144 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 171 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 137. 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 144 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 171 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 172 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 172 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 174 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-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 AAA Server receives a RADIUS message. You can preempt this value 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 AAA 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 server parses the NAI (userid@realm), it assigns the userid value to this attribute.
User-Realm	After the 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://*Filename

You must specify the decision file to use each time the `POLICY` action is called. If necessary, your group entries should also contain a `Decision` reply item to return the next event to the FSM.

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 AAA Server already call the corresponding decision file at the appropriate point in the server process. You need only make implementation-specific changes. See "Modifying the FSM for DNIS" on page 177 and "Modifying the FSM for DAC" on page 176 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. The `Policy-Pointer` string cannot be more than 63 characters long.

```
realm ProLDAP "comment"
{
   Policy-Pointer = "decisionfile://path-to-file"
   Directory . . .
}
```

The `POLICY` action will look for the first group entry in the decision file that matches the A-V pairs in a user request and reply accordingly.

From user profiles

In the default users file or realm files, place a `Policy-Pointer` as a check or reply item naming the full path to the decision file containing the group authorization policies. Enclose the pointer in double or single quotes. The `Policy-Pointer` string cannot be more than 63 characters long.

```
carl Password = carl, Policy-Pointer = "decisionfile://path-to-file"
or
```

```
fred Password = fred
   Policy-Pointer = "decisionfile://path-to-file"
```

The POLICY action will look for the first group entry in the decision file that matches the A-V pairs in a user request and reply accordingly. See "User/Realm policy" on page 130 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.

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 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 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 AAA 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] = "7341234567" ) )
    {
        exit "Forward"
    }
}
if ( ( count(Called-Station-Id[0]) > 0 ) &&
     ( Called-Station-Id[0] = "7341236543" ) )
    {
        exit "Forward"
    }
}
```

You can enter additional conditions and attributes 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 server's configuration directory, by default `/etc/opt/aaa`.

Configuration Files

About Configuration Files

Some advanced features of the RAD-Series Server cannot be configured through the Server Manager console. For example, if you want to define Advanced Policy, vendor-specific attributes, or logging behavior, you will need to manually edit the RAD-Series Server 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 an italic face. 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 server—a client, a user profile, a realm, and others.

A parameter (field) may contain one or more A-V pairs. See page 182 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 dictionary and vendors files have a maximum input line length of 255 characters. All other configuration files have a maximum input line length of 1023 characters. Anything exceeding the limit is processed as another line.

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 leading pound sign ('#') character. The server ignores all comment lines, as well as empty lines and entirely-blank lines. All lines are counted for the purpose of reporting errors, warnings, or changes. The Server Manager removes all the comment lines when making changes to a file.

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 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 any logical group that can be defined with attribute-value (A-V) pairs. See "Using Advanced Policy" on page 127 for information about creating decision files.
- **`dictionary`** defines all attributes and values that may be used to build A-V pairs recognized by the 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.
- **`EAP.authfile`** defines realm authentication actions.
- **`las.conf`** defines Local Authorization Server 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. You can edit this to reorder server 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, and radius.fsm files cannot be edited with the Server Manager; they must be manually-edited. Furthermore, these three files are read once at server startup time and are not re-read when the server receives a HUP signal.

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, string, vendor, tag string, tag integer, date, integer, string, octet, short, IPv6 prefix and interface-id values. The attribute may take any of the supported, legal values defined for it in the `dictionary` file.

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` file. See the RADIUS RFC documents (2865, 2866, 2867, 2868, 2869) 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. Put a comma after each line, except for the last in the list:

```
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 (2011).
- '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 2010"
Expiration = "Apr 18 2010 UTC"
Expiration = Apr-19-2010UTC
Expiration = "Apr-19-2010 10:32:59"
Expiration = "Apr-09-2010 10:32:59 utc"
Expiration = "2010-04-05 0:0:0"
Expiration = 2010-03-16
Expiration = 2010-03-17UTC
Expiration = 2010-03-18/UTC
Expiration = 2010-03-15/01:02:03
Expiration = 2010-03-15/04:05:06utc
Expiration = 2010-03-15/04:05:07/utc
Expiration = "2010-04-05 0:0:1 UTC"
Expiration = "2010-04-05 0:0:2utc"
Expiration = "2010-04-05 0:0:3 UTC  "
Expiration = " 2010-04-05 0:0:3 UTC "
```

The following example is a syntactically valid A-V pair list:

```
Password = "rock", Service-Type = "Framed", Reply-Message = "OK"
```

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"Reply-Message="This is not OK"
Password = rock Service-Type = Framed Reply-Message = This is not OK
```

Tagged Attributes

An AAA 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.

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 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.

aaa.config

The `aaa.config` file contains user-defined entries for server properties. These entries override the server's built-in defaults.

Note: aaa.config combines the tunneling.hint and engine.config files of previous server versions.

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

You can include configuration data in multiple text files and load them at 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 server will look for the file in the configuration directory.

Server Variable Syntax

Server variables override the server's built-in defaults. Those that can be configured on the Server Properties pages in the Server Manager will be noted

To add a server variable, enter a line in aaa.config for:

variable = value

variable value

Server variables are loaded into memory when the server starts. Restart 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
authenticate_as_computer (boolean)	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`. The default value is Enabled (Yes). Server Manager property: **Microsoft Host-Based Authentication**
avpair_checking (boolean)	Whether or not the 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.
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. Server Manager property: **Hold Replies**

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 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. Server Manager property: **Global Retry Limit**
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, fifteen, thirty, or forty-five minutes to the base value to determine when a DNS entry should be refreshed. The default value is 3600 (1 hour). Server Manager property: **DNS Refresh Interval**
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. Server Manager property: **DNS Refresh Time Frame**
dns_max_aliases (integer)	Maximum number of aliases per client. The default value is 3.
global_acct_q.hold (integer)	The number of additional seconds each accounting request should be held after it has been replied to. When this time expires, the request is released. This hold time allows the server to detect and respond to duplicate requests. You should specify a value that is twice the retransmission interval configured on your access device. The default value is 0 seconds. Server Manager property: **Hold Accounting Requests**

Variable	Description
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. Server Manager property: **Max. Accounting Requests**
global_auth_q.hold (integer)	The number of additional seconds each authentication request should be held after it has been replied to. When this time expires, the request is released. This hold time allows the server to detect and respond to duplicate requests. You should specify a value that is twice the retransmission interval configured on your access device. The default value is 0 seconds. Server Manager property: **Hold Authentication Requests**
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 you can 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 server stops responding to the radcheck command. The default value is 40000. Server Manager property: **Max. Authentication Requests**

Variable	Description
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 Operation" on page 245 for more information. The default value is off.
list_copy_limit (integer)	For customized server configurations that accumulate A-V pairs or generate large responses. The default value is 512.
maximum-output-file-size (integer)	The maximum size in bytes of the 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.2010120103, then it is renamed to logfile_part01.2010120103 and the new logfile is named logfile_part02.2010120103. See "DESCRIPTION `roll logfile`" on page 107 and "DESCRIPTION `roll stream`" on page 108 for more details of the file naming. The default value is 2147483647. Server Manager property: **Maximum Log File Size**
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. Server Manager property: **Max. Receive Message Size**

Variable	Description
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 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. Server Manager property: **Max. Send Message Size**
send_proxy_action (boolean)	Controls the sending of the MERIT Proxy-Action A-V pair in RADIUS replies or packets forwarded through the server to another RADIUS server. Valid values are on or off. The default value is off.
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 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 when they reach their end-state. It is useful for a customized 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.

Variable	Description
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 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 and +mf. • 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 and +mf.

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 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 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. Server Manager property: **Enable SNMP Support**

Certificate Path 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 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:

```
security.paths
{
    Certificate-Path          "<file_path>"
    Key-Path                  "<file_path>"
    CA-Path                   "<file_path>"
    Random-Path               "<file_path>"
    CRL-Path                  "<file_path>"
    ECDSA-Certificate-Path    "<file_path>"
    ECDSA-Key-Path            "<file_path>"
    ECDHE-Parameter-Curve     "<curve_name>"
    Cert-Name-Attr            "<attr_desc>"
    Tunneled-EAP-MTU-Reduction<value>
    SSL-Debug-Level           <value>
}
```

Variable	Description
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* Server Manager property: **Server Certificate Path**
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* Server Manager property: **Server Private Key Path**

Variable	Description
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* Server Manager property: **Client Certificate Authority Path**
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* Server Manager property: **Random Seed Path**
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* Server Manager property: **Certificate Revocation List Path**
ECDSA-Certificate-Path (string)	Full path to the RAD-Series Server ECDSA 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*
ECDSA-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 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*
ECDSA-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".

Variable	Description
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" Server Manager property: **Client User Name Attribute**
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
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

ProLDAP Connection Properties

The ProLDAP support uses the block format to define variables. The block format looks like this:

```
aatv.ProLDAP
{
    variable value
    . . .
}
```

Variable	Description
LDAP-Version (integer)	The version of the LDAP protocol to employ. If your LDAP server requires version 2 of the protocol, enter that number here. The default value is 3. Server Manager property: **LDAP version**
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. Server Manager property: **Retry interval (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. Server Manager property: **Minimum wait before retry (seconds)**
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. Server Manager property: **LDAP timeout (seconds)**

Variable	Description
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). Server Manager property: **Timeout for TCP connect (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. Server Manager property: **OpenLDAP Debug**
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. Server Manager property: **TLS CA certificate directory**
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. Server Manager property: **TLS CA certificate file**
TLS-CertFile (string)	The TLS RAD-Series Server certificate file specifies the file that contains the certificate that the RAD-Series 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. Server Manager property: **TLS certificate file**
TLS-KeyFile (string)	The TLS RAD-Series Server private key file specifies the file that contains the private key that the RAD-Series 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. Server Manager property: **TLS private key file**

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 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 1 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 not configured, the server will not open an accounting listen socket for the given <ipaddress>.
authport (integer)	Sets the UDP port number to receive authentication requests on. The minimum value is 1 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 not configured, the server will not open an authentication listen socket for the given <ipaddress>.

Parameter	Description
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
    send-user-class-option                      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
}
```

Variable	Description
dhcp-enabled (string)	If Yes, it enables the DHCP Relay function. The default is No. Server Manager property: **Enable DHCP Support**
dhcp-server-name (string)	Specifies the fully qualified domain name of the DHCPv4 server to use. You can configure this variable or the dhcp-server-ip-address but one is required. There is no default value. Server Manager property: **DHCP Server Name**
dhcp-server-ip-address (IP address)	Specifies the IPv4 address of the DHCPv4 server to use. You can 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. Server Manager property: **DHCP Server IP Address**
dhcp-server-port (integer)	The UDP port on the DHCP server to which DHCP requests are sent. The default value is 67. Server Manager property: **DHCP Server Port**
dhcp-relay-port (integer)	The UDP port on the RAD-Series Server at which DHCP responses are received. The default value is 67. Server Manager property: **DHCP Relay Port**
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. Server Manager property: **Client Hardware Type**

Variable	Description
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. Server Manager property: **Initial Retransmission Interval**
max-retransmission-interval (integer)	The maximum interval in seconds at which the RAD-Series Server retransmits DHCP requests. The default value is 60 seconds. Server Manager property: **Maximum Retransmission Interval**
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. Server Manager property: **Maximum Discover Retransmissions**
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. Server Manager property: **Maximum Request Retransmissions**
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. Server Manager property: **Maximum DHCP Message Length**
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. Server Manager property: **Send Maximum DHCP Message Size Option**
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. Server Manager property: **Send User Class Option**

Tunneling Properties

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 variable specifies what the 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. Server Manager property: **Tunneling Reply Items**

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 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"
```

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.

The entries in these files correspond to the Local Realms page in Server Manager.

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).

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.

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 "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 EAP-AKA" on page 248 for information on configuring EAP-AKA. See "Configuring EAP-SIM" on page 268 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

Flag	Matches Access-Request Attribute
-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 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

ProLDAP Authentication Type

The ProLDAP authentication type requires extended parameters to configure the connection to the LDAP server. The extended parameters look like this:

```
<realm_name> ProLDAP "<realm_comment>"
{
    Request-Attribute-For-Search    <attribute_name>
    Filter-Type                     <filter_value>
    retrieve-only                   <value>
    Directory "<LDAP_directory_name>"
    {
        URL                 "<type>://<server>[:<port>]"
        Administrator       "<admin>"
        Password            "<password>"
        SearchBase          "<search_base>"
        Authenticate        { Search | Bind | Auto }
        Filter              <filter>
    }
    Directory "<LDAP_directory_name2>"
    {
        . . .
    }
}
```

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). 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`.
Filter-Type	The 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.

Parameter	Description
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.
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 server's TLS CA certificate path and file in the "ProLDAP Connection Properties" on page 197. 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"

Parameter	Description
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	Password used by the `Administrator` to bind to the LDAP directory server. `Password` is not required if there is no `Administrator`. 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 ProLDAP authfile configuration

```
# This realm uses an LDAP database
realm3.com          PROLDAP     "LDAP lookup"
{
    Request-Attribute-For-Search    User-Id
    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
        Filter          uid
    }
}
```

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 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 server, 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.

In versions prior to 7.1.1 of the server there was only one `EAP.authfile` that was applied to all clients. Starting with version 7.1.1, the client prefix is applied to the `EAP.authfile`.

clients

The `clients` file defines the access devices RADIUS servers that this server communicates with. The entries in this file correspond to your entries on the Server Manager Access Devices page and Proxies page. `Prefix, reply_holdtime and srcip` must be added manually to the device entry in the `clients` file.

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:NASv1 west.
a1.flatlink.com     f25lt       type=Merit:Proxy
10.2.7.7:1666:1667 real-secrettype=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 wild card pattern (see "Using Wild Cards for IP Addresses" on page 40). To use IPv6 addresses you must enable IPv6 in the aaa.config file. See `ipv6_enabled`, "General Server Properties" on page 186.
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 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 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 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 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 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	Normally created by Server Manager, but it only needs to be added to a clients entry if you wish to specify a `Prefix` of '`V1`' or '`v1`'. In that case specify this option 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. See "Token-Hold-Adjustment" on page 222 for details. 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 Operation" on page 245. 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. This option can only be configured by manually adding it to a device entry in the clients file.

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`. If you create them through the Server Manager, they will be named whatever you specified when configuring the realm, followed by 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.

Parameter	Description
check-items	Any check or deny items to be matched by the Access-Requests before authorizing a user. See the dictionary 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 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 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.

Most of these attributes are configurable through the Server Manager Users:Add User page tabs. Any attribute that doesn't have a corresponding field can be entered as an A-V pair on the Free tab or manually added to the user entry in the realm file.

See the `dictionary` file 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 wild card: Deny-Message = "*" This wild card 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 183 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 • 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 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 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 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 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` to determine which instances to send based on the NAS type. See "clients" on page 211 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` file. They can be configured through the Server Manager User:Add User page tabs.

Date type attributes used for check/deny/reply attributes can be entered in many formats. See "Date Attribute Value Formats" on page 183 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 authentication 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 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.

Most parameters can be configured on the Server Manager's Server Properties:Session Table Properties page. The parameters described here can be manually edited in `las.conf`.

LAS Session Configuration

These parameters let you override the server's default values related to session timing and session limits. Timing parameters are specified in seconds.

Syntax

attribute value

Session Attributes

Attribute	Description
Auto-Save	The interval, in seconds, between saves of the session table if there are any changes. The default value is 300 seconds (5 minutes). Server Manager property: **Auto Save Interval**
Session-Check-Time	The interval, in seconds, at which the LAS checks for Session-Idle-Time timeouts. The default is 300 (5 minutes). Server Manager property: **Session Check Time**

Attribute	Description
Session-Clear-Time	Time in seconds the LAS waits before removing a session in the Suspended state. The default is 1815 (30 minutes and 15 seconds). Server Manager property: **Session Clear Time**
Session-Hold-Time	Time in seconds the LAS waits before removing a session in the Stop state. The default is 45 seconds. Server Manager property: **Session Hold Time**
Session-Idle-Time	Time in seconds the LAS waits for an Accounting Interim Update message (check point) before suspending a session. The default is 915 (15 minutes and 15 seconds). Server Manager property: **Session Idle Time**
Session-Kill-Time	Time in seconds the LAS waits before removing sessions from the session table that are in the Noconfirm, Normal, Cancel, Reject, NASreboot, or Collision state. The default is 300 (5 minutes). Server Manager property: **Session Kill Time**
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). Server Manager property: **Session Table Limit**
Session-Update-Time	Time in seconds between timer-based updates to the status of sessions. The default is 5 seconds. Server Manager property: **Session Update Time**
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 server's overall licensed limit. The value 0 prevents access to any user that does not have a specific value configured. Server Manager property: **Simultaneous Use**

Attribute	Description
Token-Hold-Adjustment	The additional time the LAS waits for an Accounting-Start message. The minimum base time is the client's `reply_holdtime`, which by default is 6 seconds, but is configurable in the `clients` file See "Reply_holdtime" on page 214 for details. After the combined time elapses without an Accounting-Start message, a session is moved into the Noconfirm state and does not count against a Simultaneous Use limit. The default is 5 seconds. Server Manager property: **Token Hold Adjustment**
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 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. Server Manager property: **Session Collision Checking**

LAS Realm Configuration

This section lists realms by name and, optionally, any custom AATV support for the realm. Turning on Session Tracking through the Server Manager automatically adds a realm entry to `las.conf`.

Note: If you make a realm entry by editing las.conf directly and then disable Session Tracking through the Server Manager, the entry will be deleted.

Syntax

```
Realm realm-name
    Authorization   LAS-Authorization-AATV
    Accounting      LAS-Accounting-AATV
    Service         number-of-services
        service-name
        service-name
        ...
    End-Service
    Tokenpool       number-of-tokenpools
        pool-name   max-number-of-tokens
        pool-name   max-number-of-tokens
        ...
```

```
    End-Tokenpool
End-Realm
```

Realm Attributes

Attribute	Description
Realm	Required, defines the realm name associated with this block.
Authorization	Optionally specifies the AATV used for realm authorization. The default is LASGEN.
Accounting	Optionally specifies the AATV used for realm accounting. The default is ACCTGEN.
Service	Optionally specifies the number of services supported by the realm and lists the names of the defined services to support. There is no default.
service-name	Names of a supported service. There is no default.
End-Service	Defines the end of the Service block
Tokenpool	Optionally specifies the number of token pools supported by the realm and lists the pool names There is no default.
pool-name	The name of a defined token pool followed by how many tokens a realm may use from that pool.
End-Tokenpool	Defines the end of the Tokenpool block
End-Realm	Required, defines the end of the Realm block

Logging LAS Realms

The default 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 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

You can 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-StringValue-StringVendor-Code  Vendor-Name
    {
        Standard-ValueVendor-Specific-Value
        . . .
    }
```

Parameter	Description
Attribute-String	An optional string that defaults to ATTRIBUTE when not specified. When adding vendor-specific attributes to the dictionary file, 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 file, 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 and .users files in vendor-specific attributes.

Parameter	Description
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 nonconflicting vendor-specific attributes. They address backward compatibility issues for very old NASes. You should not have to use these parameters for new VSA definitions.

Example

```
Merit.Attribute     Merit.Value    61   Merit
    (
              144         144
              145         145
                . . .
    )
```

dictionary

The `dictionary` file 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

You can 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.

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.
Integer-Encoding	Actual attribute number code.
Type	The attribute type. May be: • octet: 8-bit unsigned integer value • short: 16-bit unsigned integer value • integer: 32-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 • 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)

Parameter	Description
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

Pruning is a feature that allows the 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, ENCAPS|NOENCAPS, CONFIG, INTERNAL*)*

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 server log file or session logs.

ENCAPS: Encapsulate the value in the vendor-specific attribute, regardless of the vendor (default value).

NOENCAPS: Do not encapsulate the value within the vendor-specific attribute. ENCAPS and NOENCAPS keywords are mutually exclusive. If you specify both, only the last one will apply.

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.

Note: Always specify pruning expressions for vendor-specific attributes or they will not be returned to the client in any replies.

Example

```
ATTRIBUTE       Framed-Protocol  7    integer (1, 0, 0)
Merit.ATTRIBUTE User-Realm       223  string  (*, 0, 0, ENCAPS,INTERNAL)
Interlink.Attr  Address-Pool     1    string  (0, 0, 0, INTERNAL)
```

Value Entry Syntax

Valid values for integer-type attributes may be defined in the `dictionary` file. 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.

Example

```
VALUE         Framed-Protocol  PPP             1
VALUE         Framed-Protocol  SLIP            2
Merit.VALUE   LAS-Code         LAS-Normal      0
Merit.VALUE   LAS-Code         LAS-Reject      1
```

log.config

The `log.config` file specifies how accounting logs are generated in the 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, you can 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: `'2011-01-30 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. "2010-02-24" in local time zone.
LOG_ACCT (=Livingston accounting) displays a date value "mmm d yyyy",
e.g. "Feb 24 2010", 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, you can 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_V1_1 (previous version of Merit logging) • 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 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)
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.

Parameter	Description
debug	The value specifies the logging debug level for accounting messages. If the 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.
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. 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 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 old {
    aatv      log_v1_1
    buffer    1
    close     on
    filename  record.%Y%m%d.las
}
stream new {
    aatv      log_v2_0
    aatv-value 7
    buffer    1
    close     on
    filename  recordv2.%Y%m%d.las
}
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 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 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

You can 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. 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, ORACLE, 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 that may be 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 that may be passed to an Action as an argument. Value may be any string. Only one string argument may be specified for each event. With the POLICY module you can use the Xstring attribute to specify an URL that identifies where the policy definitions are stored. The pointer string must be less than 64 characters long. See "Using Advanced Policy" on page 127 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, but you can create your own names 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_CANCEL	The incoming Accounting-Request is a message to cancel the session.
ACCT_DUP	The incoming Accounting-Request is a duplicate.
ACCT_MSTART	The originating NAS has just rebooted.
ACCT_MSTOP	The originating NAS is about to reboot.
ACCT_OFF	Received accounting message has a Status-Type of Accounting-Off.

Event	Description
ACCT_ON	Received accounting message has a Status-Type of Accounting-On.
ACCT_START	Received accounting message has a Status-Type of Start.
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.
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.
CONTINUE	The incoming Access-Request is a continuation of an in-progress EAP conversation. Generally, you should allow the 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 server to handle these events without modification.
ERROR	The previous action generated an error. Generally, you should allow the server to handle these events without modification.

Event	Description
LASCP	The incoming Accounting-Request is an accounting-interim-update. Generally, you should allow the server to handle these events without modification. This event is not pre-defined, it must be defined in the FSM file.
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.
RETRIEVE_ERROR	This value may be returned by the user profile retrieval (iaaaRealm, iaaaUsers) when a user's profile cannot be found. Also returned when the "DEFAULT" entry is "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 server to handle these events without modification.

Creating New Event Names

To create custom event names, add to the .fsm file:

```
%event name value
```

- The `%` must be in column 1.

- ***name*** can be any alphanumeric string and may include underscores (_).
- *value* is an optional integer value for the event which allows your custom plugin to use a specific value for an event.
- 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 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.
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

Action	Description
NULL	No action placeholder.
PASSWD	Verifies password against Unix password system
POLICY	Evaluates advanced policies
POSTLAS	Allocates tokens for L.A.S.
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
SECURID	Performs authentication with a SecurID server
SIM-TripletFile	Retrieves user profiles from realm files based on the Request-Attribute-For-Search option which defaults to Real-Username. If the reply items contain a set of GSM-Triplet attributes then starting randomly in the list only the required number of GSM triplets will be returned.
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 server startup:

.FSM	Description
radius.fsm	Basic functions. This is the default read by the server at startup, unless another .fsm is specified.
logall.fsm	Logs all accounting messages in the format specified in the log.config file.

The default `radius.fsm` file is created from the predefined `/etc/opt/aaa/default.fsm` during server installation. Unless you choose to use an alternate state table, 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 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 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 server builds the internal list of authentication listen sockets. Opening of accounting listen sockets is done in a similar manner:

- The 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 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 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.

  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 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 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 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 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 189), 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 187, 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 187). 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 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 211 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.

Note: Configuration of `radius_socket{}` and parameters are not currently supported by the Server Manager, and must be manually edited.

- There are some very preliminary steps not described here, such as updating one's local DNS with IPv6 mappings, and updating the 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 246 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 246 for details on what this means. Configure the server's aaa.config file with a "`ipv6_enabled yes`" parameter. This parameter must be manually configured.
- Configure any IPv6 sockets on which the RADIUS server needs to listen for requests. See "Managing RADIUS Listen Sockets" on page 242 and "RADIUS Listen Socket Properties" on page 199. This configuration must be manually edited.
- 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 244. This configuration must be manually edited.
- Configure any IPv6 clients and servers in the clients file. This can be done with the Server Manager as indicated in "Defining Access Devices" on page 40 and "Defining Proxies" on page 44. To manually edit them, see "clients" on page 211.
- Configure the server to communicate with an LDAP server using TCP/IPv6, if desired. See "ProLDAP Authentication Type" on page 207. This configuration must be manually edited for IPv6 addresses.
- 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 102 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 103 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 od 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 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 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 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 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 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

The configuration files must be edited manually as EAP-AKA cannot currently be configured using the Server Manager.

Configuration information for EAP-AKA is placed in several files, with some default values built into the 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 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 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 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 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 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 286 for a description of the algorithm. The 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 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 215 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\x33\x33\x33\x33\x
33\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\x
55\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 wild carded.

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 wild carded 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 "ProLDAP Authentication Type" on page 252 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 291 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 286 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 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 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 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 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.
Protected-Identity-Exchanges	The use of the AT_CHECKCODE attribute is an optional feature in EAP-AKA. This parameter determines if the 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.

Parameter	Description
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.
Fast-Reauth-Lookup	Fast re-authentication is an optional EAP-AKA feature which the Interlink EAP-AKA 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 293 for details. There is no default.

Parameter	Description
Fast-Reauth-Update	Fast re-authentication is an optional EAP-AKA feature which the Interlink EAP-AKA 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 293 for details. There is no default.
Pseudonym-Lookup	Pseudonyms are an optional EAP-AKA feature which the Interlink EAP-AKA 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 291, 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 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 291. There is no default.

Parameter	Description
Max-Number-Of-Reauths-Before-Full-Auth-Is-Required	Fast re-authentication is an optional EAP-AKA feature which the Interlink EAP-AKA 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.
Fast-Reauth-Realm	When providing a fast reauthentication identity, the 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.

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 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.
Generate-Random-Character-Pseudonyms	The Interlink 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 265.

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 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
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 286 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 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 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 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 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 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 Yes
Max-Number-Of-Reauths-Before-Full-Auth-Is-Required	Fast re-authentication is an optional EAP-AKA feature which the Interlink EAP-AKA 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 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),

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 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-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 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 server shuts down. The valid values are `"Enabled"` and `"Disabled"`. 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

The configuration files must be edited manually as EAP-SIM cannot currently be configured using the Server Manager.

Configuration information for EAP-SIM is placed in several files, with some default values built into the 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 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 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 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 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 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 286 for a description. The 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` 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 215 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\x1
a\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\x2
b\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\x3
c\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\x4
c\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\x5
c\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\x6
c\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\x7
c\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\x8
c\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\x9
c\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 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 272.

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 wild carded.

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 wild carded 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 252 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 291 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 286 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 286 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 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 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 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 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 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 293 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 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 293 for details. There is no default.
Pseudonym-Lookup	Pseudonyms are an optional EAP-SIM feature which the Interlink EAP-SIM 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 291, 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 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 291. 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 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 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 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 265.

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 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 286 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 286 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 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 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 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 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"`.
Max-Number-Of-Reauths-Before-Full-Auth-Is-Required	Fast re-authentication is an optional EAP-SIM feature which the Interlink EAP-SIM 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 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 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 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 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 plugin, 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.

Parameter	Description
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. 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 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 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 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 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 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.`
- `  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.`

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

- 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 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 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 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

Numerics

3GPP Milenage 287
3GPP-Milenage 287
55iaaa-radius.ldif 114, 117
802.1X 1, 53, 65

A

A3/A8 algorithm 269, 276, 282
A3/A8 algorithm plug-in 287
aaa.config 72, 84, 119, 181, 185, 260, 265, 281, 287, 288
aaaCheck 117
aaaDeny 117
aaaPerson 118
aaaReply 117
aaasample.ldif 117
aaa-users.xml 25
AATV 235, 239
access
  controlling 68, 127, 175
  denying 119
access device 40, 211
  adding 41
  deleting 43
  modifying 43
access policy pointer 174
Access-Accept 4, 6, 86
Access-Challenge 4, 12, 14
Access-Reject 4
Access-Request 4, 6, 42, 84
Accounting
  log file rollover 106, 108
accounting 10
  attributes 92
  extensions 93
  LAS realms 223
  log file 6, 10, 91, 94, 96, 108, 229
  messages 69, 74
  port 33, 46, 47, 48
  proxying 44, 48, 96
  record format 92
Accounting Interim-Update 4
Accounting-Alive 10
Accounting-Request 4
Accounting-Response 4, 10
acct_udp_recv_buffer_size 200
ACCTGEN 223
acctport 199
actions 2, 6, 235, 239
active sessions 77
administration 20
Advanced Policy 8, 127
AgentX 79
AKA algorithm 249, 254, 261
AKA algorithm plug-in 287
algorithmic pseudonyms 259, 264, 266, 280, 285, 291
anonymous user 50
attributes
  accounting 92
  appending 46, 119
  custom 226, 228
  Interlink 114, 117, 173, 175, 182, 216, 220
  Oracle 123
  RADIUS 182
  tagged 87, 144, 184
  tunneling 84, 123
  types 182
  vendor-specific 41, 45, 46, 65, 94, 137, 144, 212, 224, 225
AUTH_NET_REQ 121
AUTH_NET_USERS 123
auth_udp_recv_buffer_size 200
authentication 7, 12, 14
  port 33, 46, 47
  requests 74
authentication realm 49, 50, 51
  adding 51, 52
  deleting 59
  modifying 59
authfile 7, 12, 14, 49, 53, 136, 174, 181, 251, 272
  prefixed 210
authorization 8
  policies 114, 127
authport 199
A-V pairs 6, 65, 179, 186, 217
  LDAP 118
  returning 46
  syntax 182
avpair_checking 186

C

CA certificate 52, 72, 76
callback service 68
CA-Path 195
Case-sensitivity 206
central accounting 48
Certificate Authority 61, 62
Certificate Revocation List 73
Certificate-Path 194
certificates 12, 26, 38, 52, 60, 72, 76
  EAP-TLS 61
  generating 62
  installing 63
  locations 64
  PEAP 61
  self-signed 61
  TTLS 61
Cert-Name-Attr 196
Challenge-Handshake Authentication Protocol
  See CHAP and MS-CHAP
CHAP 1, 7, 53
check items 8, 65, 67, 114, 184, 215, 219
  LDAP 117
clear hash 218
clients file 40, 181, 210, 211, 227
compression 89, 95
concurrent sessions 54, 77, 78, 119, 218, 221
configuration attributes 215, 217
configuration files 2, 3, 6, 38, 179
  format 179
  input length 180
configurations 34
  editing 22
  loading 38
  reverting 88
  saving 39
connecting servers 31
control access 119, 175, 220
  deny 70
  device IP 68
  dial-up number 69
  MAC address 69
  NAS 127
  policy 70
CRL-Path 195
cwd 186

D

DAC 127, 174, 175, 176, 177
  DAC.fsm 176
  DAC.grp 176
DataStore 7, 14
db_srv 121, 125
db_srv.opt 125, 126
Debug 198
debugging 33, 76, 106, 111, 196, 198
decision files 127, 176, 177, 181
  calling 174
  DAC 174, 176
  DNIS 174
  DNIS routing 178
  reply-egress.grp 130
  request-ingress.grp 128
default users file 136, 239
default.fsm 96, 241
default_reply_holdtime 186
default_retry_limit 187
default-source-ipv4-address 187
default-source-ipv6-address 187
deny access 70, 119
deny items 219
device
  See access device
DHCP 73, 80, 81
DHCP properties
  dhcp-client-hardware-type 201
  dhcp-enabled 201
  dhcp-relay-port 201
  dhcp-server-ip-address 201
  dhcp-server-name 201
  dhcp-server-port 201
  initial-retransmission-interval 202
  max-dhcp-msg-len 202
  max-number-of-dhcpdiscover-retransmissions 202
  max-number-of-dhcprequest-retransmissions 202
  max-retransmission-interval 202
  send-maximum-dhcp-message-size-option 202
  send-user-class-option 202
dhcp-client-hardware-type 201
dhcp-enabled 201
dhcp-relay-port 201
dhcp-server-ip-address 201
dhcp-server-name 201

dhcp-server-port 201
dial-up number 69
dictionary 93, 181, 182, 217, 219, 227, 228
Distinguished Name 120
DNIS 174
DNIS routing 127, 177, 178
  DNIS.fsm 177
  DNIS.grp 177
DNS 73, 80, 187, 190
dns_address_aging 187
dns_address_window 187
dns_max_aliases 187
domain name 30, 31, 41, 49, 50
Dynamic Access Control
  See DAC
Dynamic Host Configuration Protocol
  See DHCP

E

EAP 1, 6, 7, 12, 14, 49, 53
EAP.authfile 7, 14, 49, 181, 253, 275
EAP-AKA 248
EAP-GTC 18, 66
EAP-MD5 66
EAPOL 6
EAP-SIM 268
  RFC 248, 268
EAP-TLS 52, 60, 66, 72, 76
  certificates 61, 75
ECDSA-Certificate-Path 195
ECDSA-Key-Path 195
ECDSA-Parameter-Curve 195
Enable-Default-Conf 197
encapsulation 6, 12, 14, 227
encryption 6, 45, 66, 116
error messages 111, 112
events 2, 6, 235, 236
  custom 238
  predefined 236
Extensible Authentication Protocol
  See EAP
extensions
  accounting 93, 94
  LDAP 114, 117
  MIB 79
  server plug-ins 2

F

failover 56, 58, 122
fast reauthentication 253, 256, 258, 262, 263, 266, 275, 277, 278, 279, 283, 284, 293
fill.sql 124
filters 53, 67, 119, 120
Finite State Machine
  See FSM
framed service 67
FSM 2, 6, 10, 91, 121, 127, 128, 130, 132, 177, 181, 229, 234, 238
  DNIS.fsm 177
  modifying 48, 174, 176, 177, 236
  radius.fsm 176, 177
  states 234

G

global_acct_q.hold 187
global_acct_q.limit 188
global_auth_q.hold 188
global_auth_q.limit 188
GSM triplet 269, 272, 273, 277, 283
gui.properties 25

H

hashing algorithms 6, 66, 71, 116
hints 84

I

iaaaAgent.conf 234
iaaaFile 251, 272
iaaa-radius.schema 117
IANA 224
including files 185
initial-retransmission-interval 202
installation directories 72
interim accounting logging 96
Interlink
  attributes 114, 117, 173, 175, 182, 216, 220
  object classes 117
IP address 30, 31, 41, 73, 80, 183
  static 68, 119
ipaddr 199
iPlanet 5.0 114, 117
IPv6 245
IPv6 Operation 245

IPv6 Readiness 246
ipv6_enabled 189

J

Java RMI 3, 30
java.security 26
JDK 26
JSSE 26, 27

K

Key-Path 194
keys 26, 60, 72
  generating 62
  installing 63
keystore 26
keytool 26

L

LAS 8
  configuration 220
  configuration attributes 218
  logging 223
  realms 222
  sessions 220
las.conf 8, 72, 181, 220
LASGEN 223
LDAP 54, 76, 114, 207, 252, 274
  check/reply items 117, 119
  Distinguished Name 120
  extending 117
  ProLDAP 114
  tunneling attributes 85
  using SSL 115
LDAP-Version 197
LDIF file
  tunneling attributes 85
legacy server 50, 51
limiting sessions 119
list_copy_limit 189
Livingston Call Detail Records 92, 94, 108
LM hash 116, 218
Local Authorization Server
  See LAS
localFile 251, 272
localhost 30, 35, 44, 98
log files 23, 30
  accounting 10, 91, 96, 108, 223, 229
  centralizing 96
  compression 89
  modifying 94, 96
  naming 95
  rollover 95, 106, 107
  server 33, 89, 111
  size 75
log.config 10, 92, 94, 181, 229
log_forwarding 190
log_generated_request 190
logall.fsm 241
logical groups 127
login service 67
loopback 30, 44

M

MAC address 69
master agent 79
max-dhcp-msg-len 202
maximum sessions 78, 221
maximum-output-file-size 189
max-number-of-dhcpdiscover-retransmissions 202
max-number-of-dhcprequest-retransmissions 202
max-retransmission-interval 202
MD5 hash 6, 116, 218
MERIT 91, 92, 94, 100, 108
message handling 74
messages
  error 111
  log file 111
  logging 96
  proxying 45, 46, 48
  radiusd 112
  reply 113
  server status 35
  statistics 97
MIB objects 79
Microsoft
  See also MS-CHAP
Milenage 287
monitoring 20, 98, 105
MS-CHAP 1, 7, 53, 66
Multilink PPP 70

N

NAI format 49, 65
NAS port 119

no_append 213
NO_HINT 203
NO_HINTS attribute 84
NT hash 116, 218
NULL realm 49, 50

O

OpenLDAP 114
OpenSSL 62
Oracle 58, 114, 121
  failover 56, 58
  port 58, 126
  users table 122, 124
ourhostname 189

P

packet filter 53, 67
PAP 1, 7, 53, 66
password 25, 66
  encryption 6, 66, 71, 116
Password Authentication Protocol
  See PAP
PEAP 14, 51, 60, 72, 75
  certificates 61
  MD5 18
  MS-CHAP 18
performance tuning 75
plug-ins 2, 6
pointer
  LDAP 55
POLICY 8, 127, 174, 176, 177, 236
Policy Attribute Functions
  count 136, 167
  length 136, 168
  substr 136, 168
  tolower 136, 170
  toupper 136, 171
Policy Commands
  Delete 134, 158
  Exit 135, 166
  if 134, 157
  Insert 134, 160
  Log 135, 166
  Modify 135, 163
ports
  accounting 46, 47, 48
  authentication 46, 47
  limiting 70, 119
  Oracle 58, 126
  TCP 54, 58, 208
  UDP 6, 24, 27, 33
PPP 70
prefixed authfile 210, 213
prefixed EAP.authfile 213
prefixed users file 213
ProLDAP 76, 114, 197, 207, 252, 274
  extending 117
  known issues 120
ProLDAP authentication type 207, 252, 274
ProLDAP connection properties 197
proxy server 40, 44, 45, 46, 47, 50, 211, 212
  relay port 33
  removing 47
proxy_udp_recv_buffer_size 189
pruning 46, 219, 227
pseudonyms 253, 257, 275, 278
  algorithmic 259, 264, 266, 280, 285, 291
  random 259, 264, 280, 285, 291

R

RAD2RAD 48
radcheck 44, 101, 102, 226
raddbginc 106
RADIUS 6
  attributes 123, 182
  encryption 6
  messages 4, 6, 41, 45, 69, 74, 96
  protocol 1
  RFC 182, 219, 225, 248, 268
radius.debug 33, 106, 111
radius.fsm 2, 48, 128, 136, 174, 176, 177, 181, 234, 241
radius_log_fmt 189
radiusd 99, 112, 190, 193, 194, 199, 201, 234
radpwtst 103
radrecord 108
random number generator 115
random pseudonyms 259, 264, 280, 285, 291
random seed file 73
Random-Path 195
realm 49
  accounting 48
  authentication 50, 51
  deleting 59
  modifying 59

NULL 49
proxying 44, 46, 47
TTLS tunnel 50
realm file 49, 53, 65, 66, 67, 136, 174, 181, 215
check/reply items 219
configuration attributes 217
LAS attributes 218
reauthentication 69, 119
recv_buffer_size 189
relay ports 33
Remote Control 2, 3, 30
remote server 46, 47
reply hold time 74, 186
reply items 8, 65, 67, 78, 114, 119, 122, 174, 184, 215, 219
LDAP 117
tunneling 85, 86
reply messages 113
reply_check 193
reply_holdtime 214
reply-egress.grp 130
request-ingress.grp 128
reset
log file 33
session table 33
restarting 34, 39
retransmissions 74, 186, 187
retry limit 74, 187
Retry-Interval 197
Retry-Wait 197
rmiserver.properties 25
rollover 95, 106, 107, 108

S

SDK 2, 3, 218
Secure Socket Layer
See SSL
security files 38, 72, 76
self-referring client 35, 44, 98
self-signed certificates 61
send_buffer_size 190
send_proxy_action 190
send-maximum-dhcp-message-size-option 202
send-user-class-option 202
server
administering 23, 32
commands 21, 23, 32, 34, 35, 44
components 2
connecting 30, 31
console 20
customizing 2
DHCP relay 81
localhost 30
log file 33, 89, 111
messages 97
monitoring 98
operations 4, 6, 7, 8, 10, 12, 74
performance 75
ports 6, 33
private key 72
proxying 33, 44, 45, 46, 47, 48, 211, 212
restarting 39
start options 33, 46, 185
starting 23
statistics 97
status 21, 35, 44
stopping 34
time 37
using LDAP 54, 207
using Oracle 58
Server Manager 2, 3, 54
components 3
customizing 3
loading 38
navigation 21
password 25
shared secret 25
status 21
temporary files 22, 38, 88
user name 25
workspace 21
server programs
radcheck 101, 102
raddbinc 106
radiusd 99
radpwtst 103
radrecord 108
server properties 31, 33, 72, 185 191, 192
avpair_checking 186
CA-Path 195
Certificate-Path 194
Cert-Name-Attr 196
CRL-Path 195
cwd 186
default_reply_holdtime 186

default_retry_limit 187
default-source-ipv4-address 187
default-source-ipv6-address 187
DHCP 73, 80, 201
DNS 190
dns_address_aging 187
dns_address_window 187
dns_max_aliases 187
ECDSA-Certificate-Path 195
ECDSA-Key-Path 195
ECDSA-Parameter-Curve 195
global_acct_q.hold 187
global_acct_q.limit 188
global_auth_q.hold 188
global_auth_q.limit 188
ipv6_enabled 189
Key-Path 194
list_copy_limit 189
log_forwarding 190
log_generated_request 190
maximum-output-file-size 189
ourhostname 189
process tracking 190
proxy_udp_recv_buffer_size 189
RADIUS socket 199
radius_log_fmt 189
Random-Path 195
recv_buffer_size 189
reply_check 193
send_buffer_size 190
send_proxy_action 190
SNMP 193
SSL-Debug-Level 196
Tunneled-EAP-MTU-Reduction 196
users_file_maxlen 190
server properties - DHCP
dhcp-client-hardware-type 201
dhcp-enabled 201
dhcp-relay-port 201
dhcp-server-ip-address 201
dhcp-server-name 201
dhcp-server-port 201
initial-retransmission-interval 202
max-dhcp-msg-len 202
max-number-of-dhcpdiscover-retransmissions 202
max-number-of-dhcprequest-retransmissions 202
max-retransmission-interval 202
send-maximum-dhcp-message-size-option 202
send-user-class-option 202
server properties - LDAP
Debug 198
Enable-Default-Conf 197
LDAP-Version 197
Retry-Interval 197
Retry-Wait 197
TCP-Timeout 198
Timeout 197
TLS-CACertDir 198
TLS-CACertFile 198
TLS-CertFile 198
TLS-KeyFile 198
server properties - Socket
acct_udp_recv_buffer_size 200
acctport 199
auth_udp_recv_buffer_size 200
authport 199
ipaddr 199
server properties - Tunneling
NO_HINT 203
server.xml 24, 26, 27
session table 10, 33, 77, 91, 98
sessions 67
active 77
concurrent 78, 119, 218, 221
controlling 220
limiting 54, 67
logging 96, 229
maximum 78, 221
port limit 70
reauthenticating 69, 119
stopping 98, 105
timeout 69, 119
tracking 3, 35, 54, 69, 82, 181, 218, 222
viewing 98
sesstab 105
SHA-1 hash 218
shared secret 6, 25, 41, 45, 212
Simple Network Management Protocol
See SNMP
Simple Network Management Protocol.See SNMP
Simple Policy 114, 119
SIM-TripletFile 251, 272, 273
simultaneous sessions
See concurrent sessions
slapd.conf 117

SNMP 79, 234
Software Developers Kit 2, 3, 218
srcip 214
SSHA hash 116, 218
SSL 24, 26, 76, 115
SSL-Debug-Level 196
start options 33
states 234, 235
static IP address 119
statistics 97
status check 35, 36, 44
stopping
  server 34
  sessions 98
stopping sessions 105
system time 37

T

tagged attributes 87, 144, 184
TCP 54, 58, 120, 208
TCP-Timeout 198
temporary files 22, 38, 88
testing 33, 39, 49, 103, 106, 190
Timeout 197
timeout 67, 69, 119, 120
TLS
  See EAP-TLS
TLS-CACertDir 198
TLS-CACertFile 198
TLS-CertFile 198
TLS-KeyFile 198
tracking
  sessions 3, 35, 54, 69, 82, 181, 218, 222
  versions 224, 226, 234
troubleshooting 111
TTLS 14, 60, 72
  certificates 61
  tunnel realm 50
Tunneled-EAP-MTU-Reduction 196
tunneling 14, 78, 84
  attributes 123
  hints 78, 84
  tagged attributes 87
tunneling attributes
  LDAP 85
Tunneling Properties 203
Tunneling properties
  NO_HINT 203

U

UDP ports 6, 24, 27, 33
UNIX-crypt hash 218
user name 25, 49, 52, 65, 66
user profiles 7, 8, 12, 14, 49, 52, 53, 67, 114, 117, 174, 215
  adding 66
  attributes 65, 67, 117
  check/reply items 219
  configuration attributes 217
  LAS attributes 218
  LDAP 54, 207
  LDAPcheck items 119
  modifying 71
  Oracle 58, 121, 122, 123
user sessions 78, 221
users file 174, 181, 215
users table 122, 124
users_file_maxlen 190

V

vendors file 181, 224
vendor-specific attributes 41, 45, 46, 65, 94, 137, 144, 212, 224, 225
VeriSign 63
version tracking 224, 226, 234
VPN tunnel 78, 84

W

wild cards 40, 217
wildcards 44
Windows domains 49

X

Xstring 48, 174, 217, 236
Xvalue 217