# GNU Radius Manual

# GNU Radius Manual

for Radius version 0.96



**Sergey Poznyakoff**

# Distribution

GNU Radius is *free software*; this means that everyone is free to use it and free to redistribute it on certain conditions. GNU Radius is not in the public domain; it is copyrighted and there are restrictions on its distribution, but these restrictions are designed to permit everything that a good cooperating citizen would want to do. What is not allowed is to try to prevent others from further sharing any version of GNU Radius that they might get from you. The precise conditions are found in the GNU General Public License that comes with Radius and also appears following this section.

One way to get a copy of GNU Radius is from someone else who has it. You need not ask for our permission to do so, or tell any one else; just copy it. If you have access to the Internet, you can get the latest distribution version of GNU Radius by anonymous FTP. It is available at `ftp://ftp.gnu.org/pub/gnu/gnu-radius`

# GNU GENERAL PUBLIC LICENSE

Version 2, June 1991



## Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

## TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you

indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.
8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

## NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

## END OF TERMS AND CONDITIONS

## How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and an idea of what it does.
Copyright (C) 19yy  name of author

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
type `show w'.  This is free software, and you are welcome
to redistribute it under certain conditions; type `show c'
for details.
```

The hypothetical commands '`show w`' and '`show c`' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than '`show w`' and '`show c`'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright
interest in the program `Gnomovision'
(which makes passes at compilers) written
by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

# Introduction to Radius

Radius is a system serving for authentication and accounting. The acronym RADIUS stands for Remote Authentication in Dial-In User Service and usually denotes the underlying protocol name.

Historically, radius servers were used as a means to authenticate the user coming from a dial-in connection, but GNU Radius is much more than an authentication system: it is an advanced, customizable and extensible system for controlling access to the network.

GNU Radius has several built-in authentication and accounting methods. When these methods are not enough, it allows administrator to implement any new method he deems convenient.

GNU Radius includes radius server program capable of serving authentication and accounting requests, and a set of accompanying programs designed to monitor the activity of the server and analyze the information it provides.

# Radius Glossary

Throughout this document the following terms are used:

RADIUS
: (All capitals) The Remote Authentication in Dial-In User Service protocol as described in RFC 2138, 2865 and 2866.

NAS
: NAS stands for Network Access Server. It is a computer or a special device designed to provide access to the network. For example, it can be a computer connected to the network and equipped with several modems. Such NAS would allow the user connecting to one of its modems to access the network.

Service
: A service, such as PPP, SLIP, telnet, etc., provided to a user by the NAS.

Session
: Every single instance of a service. Session starts when the service was first provided and ends when the service is ended. A user may have multiple sessions active simultaneously if he is allowed to.

Session ID
: Session Identifier. A string of characters uniquely identifying the session.

A/V pair
: Stands for Attribute-Value pair Section 2.1 [Attributes], page 17.

Dial-In or Dial-Up user
: A user connecting to a service through the modem line.

User Database
: A database in which Radius server keeps information about users, their authentication information, etc.

User's Profile
: A record in the User Database describing a particular user. User's Profile keeps the authentication and authorization information for that user, i.e. it contains data describing how this user should be authenticated as well as which services he is allowed to be provided and parameters of these services.

# 1 Naming Conventions

This chapter describes file naming conventions used throughout this document.

Programs from the GNU Radius package use the following directories to store various configuration and log files:

*Configuration or database directory*
: A directory where all configuration files are stored.

*Log directory*
: A directory where `radiusd` stores its log files.

*Accounting directory*
: A directory where `radiusd` stores accounting detail files Section 6.2 [Detailed Request Accounting], page 67.

The default locations of these directories are determined at compile time. By default these are:

| Directory | Short Name | Default location |
|---|---|---|
| Configuration directory | raddb | /usr/local/etc/raddb |
| Log directory | radlog | /var/log |
| Accounting directory | radacct | /var/log/radacct |

These location may differ depending on your local site configuration.

Throughout this document we will refer to these directories by their short names, e.g. saying

```
... this information is contained in file 'raddb/sqlserver'
```

we actually mean '`/usr/local/etc/raddb/sqlserver`'.

If necessary, locations of these directories can be overridden by specifying appropriate command line options to a program. For example, any program from the GNU Radius package accepts command line option '`-d`' or '`--directory`', which introduces the configuration directory path.

# 2 How Radius Operates

The main purpose of Radius is to centralize authentication of users coming from various network stations. Its primary usage is for dial-in users, though it can be used for any kind of network connection.

Radius uses the Client/Server model. The main server keeps the centralized user database. Each user's profile in this database determines which services are allowed for this particular user.

The Network Access Server (NAS in short) is a machine that actually provides a service for the user. It can do so, e.g. by running a pool of modems the users can connect to. Otherwise, it can be a machine connected to the network and allowing some form of remote access, like telnet or ssh. It can even be a workstation allowing console logins to it. Whichever service it provides the NAS sends the request to the central Radius server in order to determine whether the user trying to access the service is allowed to do so. Such request carries information about user's login name, password, NAS identifier (such as its IP address), etc.

On receiving such request Radius server retrieves from its database a profile corresponding to the user's login name. The profile basically consists of two parts: a checklist used for authentication and a reply list used for authorization. The server checks the authenticity of the user using first part of the retrieved profile. If this check succeeds, it uses second part of the profile to authorize the user, i.e. to determine which service he should be provided. Then, the server responds with the parameters of the service, such as connection speed, framed IP address, etc. If any of the described checks had failed, the server sends the negative response.

If the server needs some additional information in order to process the request, it asks the NAS to supply such information. Thus, for example, if the user is allowed to use several services, he can be asked which one of them he wishes to use, etc.

When NAS receives the positive authentication response, it initiates the connection.

The NAS can be configured to notify Radius server about such events as session start, session stop or changing some parameters during the session. It can also notify the server about other events, such as NAS shutdown or startup. This is called *accounting* and the radius server responsible for processing this information is called an *accounting server*.

## 2.1 Attributes

The information the Radius requests carry is stored as a list of *Attribute-Value pairs*. Each pair consists of *Attribute number* and *Attribute value*. The *Attribute number* identifies the type of information the pair carries and the *Attribute Value* keeps the actual data.

The Value part of an attribute can contain the data of one of the following types:

Integer
: A 32-bit unsigned integer value.

IP-number
: An IPv4 IP-number.

String
: A character string up to 253 characters long.

For convenience, the attributes and the values of some frequently used integer attributes are given symbolic names. These names are assigned to attributes and values in the dictionary file Section 4.2 [dictionary file], page 41.

The attribute numbers range from 1 to 255. The attributes with the numbers greater than 255 are used internally by the sever and cannot be sent to the NAS.

The special attribute 26, *Vendor-Specific*, is available to allow vendors of the NAS hardware or software to support their own extended attributes. Section 12.1.19 [Vendor-Specific attribute], page 116.

Each attribute has a set of properties associated with it. The properties are:

*usage flags*
: These flags determine usage of the attribute in configuration files '`huntgroups`', '`hints`' and '`users`'.

*propagation*
: When a radius server functions in proxy mode, it uses the *propagation* bit to determine which attributes from the reply packet should be passed back to the requesting NAS (see Section 2.4.1.1 [Proxy Service], page 23).

*additivity*
: Some configuration rules may cause addition of new A/V pairs to the incoming request. Before addition of a new pair, `radiusd` scans the request to see if it already contains a pair with the same attribute. If it does, the value of *additivity* determines the further actions:

  None — The old pair is retained in the request, the new pair is not added to it.

  Replace — The old pair is retained in the request, but its value is replaced with that of the new pair.

  Append — The new pair is appended to the end of the pair list.

The attributes are declared in '`raddb/dictionary`' file. For a detailed description of it, See Section 4.2.4 [ATTRIBUTE], page 42. For information about particular attributes, See Chapter 12 [Attribute List], page 109.

## 2.2 Radius Requests

The term *request* means both the authentication/accounting request from NAS to a Radius server and the response that the server sends back to the NAS.

Each request contains the following fields

'`Code`'
: The code field identifies the type of the request.

'`Identifier`'
: The number in the range 0-255 used to match the request with the reply.

'`Length`'
: The length of the request packet.

'`Authenticator`'
: The 16-byte hash value used to authenticate the packet.

'`Attributes`'
: The list of attribute/value pairs carrying actual information about the request.

### 2.2.1 Authentication Requests

A NAS sends authentication requests (packets with code Access-Request) to a RADIUS server when a user is trying to connect to that NAS. Such requests convey information used to determine whether a user is allowed access to the NAS, and any special services requested for that user.

An Access-Request must contain a `User-Name` attribute Section 12.1.1 [User-Name], page 110. It should contain either a `NAS-IP-Address` attribute or `NAS-Identifier` attribute, or both of them. It also must contain either a `Password` attribute or `CHAP-Password` attribute. These attributes are passed encoded using a method based on the RSA Message Digest Algorithm MD5.

The Access-Request should contain a `NAS-Port` or `NAS-Port-Type` attribute or both unless the type of access being requested does not involve a port or the NAS does not distinguish among its ports.

Upon receiving Access-Request packet for a particular user and authenticating that user, Radius server replies to the NAS that has sent the packet with either of the following packets:

- Access-Accept
- Access-Reject
- Access-Challenge

Radius replies with Access-Accept packet when it has successfully authenticated the user. Such reply packet provides specific configuration information necessary to begin delivery of service to the user.

Radius replies with Access-Reject packet when it was unable to authenticate the user. Such packet may contain a descriptive text encapsulated in one or more `Reply-Message` attributes. The NAS may display this text along with its response to the user.

Radius replies with Access-Challenge packet when it desires to obtain more information from the user in order to determine its authenticity or to determine the kind of service to be provided to the user.

An Access-Challenge packet may include one or more `Reply-Message` attributes, and may also include a single `State` attribute, or none. No other Attributes are permitted in an Access-Challenge.

On receipt of an Access-Challenge, the Identifier field is matched with a pending Access-Request. Additionally, the Response Authenticator field must contain the correct response for the pending Access-Request. Radius discards invalid packets and issues appropriate log message.

If the NAS does not support challenge/response, it treats an Access-Challenge as though it had received an Access-Reject instead. Otherwise, upon receipt of a valid Access-Challenge the NAS prompts the user for a response, possibly displaying the text message provided in `Reply-Message` attributes of the request. It then sends its original Access-Request with a new request ID and Request Authenticator, with the `Password` attribute replaced by the encrypted user's response, and including the `State` attribute from the Access-Challenge, if any.

### 2.2.2 Accounting Requests

Accounting-Request packets are sent from a NAS to a Radius server to allow for accounting of a service provided to a user.

Upon receipt of an Accounting-Request, the server attempts to record the accounting packet Chapter 6 [Accounting], page 67, and if it succeeds in doing so it replies with an Accounting-Response packet. Otherwise it sends no reply, which then causes the NAS to retransmit its request within a preconfigured interval of time. Such retransmits will continue until either the server responds with an Accounting-Response packet or a preconfigured number of retransmits is reached, whichever occurs first.

Any attribute valid in an Access-Request or Access-Accept packet is also valid in an Accounting-Request packet, except the following attributes, which are never present in an Accounting-Request packet:

- User-Password
- CHAP-Password
- Reply-Message
- State

Either `NAS-IP-Address` or `NAS-Identifier` must be present in an Accounting-Request. It should contain a `NAS-Port` or `NAS-Port-Type` attribute or both unless the service does not involve a port or the NAS does not distinguish among its ports.

If the Accounting-Request packet includes a Framed-IP-Address, that attribute *must* contain the actual IP address of the user.

There are five types of accounting packets, which differ by the value of `Acct-Status-Type` attribute. These are:

*Session Start Packet.*

The session start packet is sent after the user has successfully passed the authentication and has started to receive the requested service. It must contain at least following attributes:

- Acct-Status-Type = Start
- User-Name
- Acct-Session-Id
- NAS-IP-Address
- NAS-Port-Id

*Session Stop Packet.*

The Session Stop Packet is sent after the user has disconnected. It conveys the information about the duration of the session, number of octets transferred, etc. It must contain at least the following attributes:

- Acct-Status-Type = Stop
- User-Name
- NAS-IP-Address
- Acct-Session-Id

The last three of them are used to find the corresponding *Session Start Packet*.

*Keepalive Packet*

The keepalive packet is sent by the NAS when it obtains some new information about the user's session, e.g. it has determined its IP address or has changed the connection speed, etc. The packet must contain at least following attributes:

- Acct-Status-Type = Alive
- User-Name
- NAS-IP-Address
- Acct-Session-Id

*Accounting Off Packet*

By sending this packet NAS requests that radius mark all sessions registered from this NAS as finished. Receiving this packet usually means that the NAS is to be shut down, or is about to change its configuration in a way that requires all currently opened sessions to be shut down. The packet must contain at least the following attributes:

- Acct-Status-Type = Accounting-Off
- NAS-IP-Address

*Accounting On Packet*

By sending this packet, the NAS informs radius that it is ready to accept the incoming connections. Usually this packet is sent after startup, or after a major reconfiguration of the NAS. It must contain at least the following attributes:

- Acct-Status-Type = Accounting-Off
- NAS-IP-Address

## 2.3 Matching Rule

A record in the radius database describing a particular rule for matching an incoming request is called a *Matching Rule*. Each such rule defines an action to be taken when the match occurs.

The *Matching Rule* consists of three distinct parts:

*Label*

This is used to identify the rule. The special usernames `DEFAULT` and `BEGIN` are reserved. These will be described in detail below.

*Left-Hand Side (LHS)*

The list of attribute-value pairs used for matching the profile against an incoming request.

*Right-Hand Side (RHS)*

The list of attribute-value pairs that define the action to be taken if the request matches LHS.

The following radius configuration files keep data in a *Matching Rule* format: '`hints`', '`huntgroups`' and '`users`'. Although they keep data in similar format, the rules that are used to match incoming requests against contents of these files differ from file to file. The following section describes these rules in detail.

## 2.4 Processing Requests

Upon receiving a request Radius applies to it a number of checks to determine whether the request comes from an authorized source. If these checks succeed, the request is processed and answered. Otherwise, the request is dropped and corresponding error message is issued (see Chapter 7 [Logging], page 71).

The following checks are performed:

Check if the username is supplied
: If the packet lacks `User-Name` attribute it is not processed

Check if the NAS is allowed to speak
: The source IP address of the machine that sent the packet is looked up in the '`clients`' file (see Section 4.3 [clients file], page 44). If no match is found, the request is rejected.

Compute the encryption key
: Using the data from the packet and the shared key value from the '`clients`' file, Radius computes the MD5 encryption key that will be used to decrypt the value of the `Password` attribute.

Process user-name hints.
: *User-name hints* are special rules that modify the request depending on user name and his credentials. These rules allow to divide users into distinct groups, each group having its own authentication and/or accounting methods. The user-name hints are stored in '`raddb/hints`' (see Section 4.6 [hints file], page 48).

Process huntgroup rules.
: *Huntgroup rules* allow to segregate incoming requests depending on the NAS and/or port number they came from. These rules are stored in '`raddb/huntgroups`' (see Section 4.7 [huntgroups file], page 48).

Determine whether the request must be proxied to another radius
: server The requests pertaining to another realm are immediately forwarded to the remote radius server for further processing. See Section 2.4.1 [Proxying], page 22, for the description of this process.

Process individual user profiles.
: This step applies only to authentication requests.

### 2.4.1 Proxying

*Proxying* is a mode of operation when a radius server forwards an incoming requests from a NAS to another radius server, waits for the latter to reply, and forwards the reply back to the requesting NAS. A common use for such operation mode is to provide *roaming* between several Internet Service Providers (ISPs). Roaming permits the ISPs to share their resources, allowing each other's users to connect to other party's equipment. Thus, users traveling outside the area of one ISP's coverage are still able to access their services through another ISP.

#### 2.4.1.1 Proxy Service

Suppose the ISP 'Local' has a roaming arrangement with the ISP 'Remote'. When the user of 'Remote' dials in to the NAS of 'Local', the NAS sends the authentication request to 'Local' radius server. The server then determines that this is a roaming user, stores a copy of the request in its internal queue, and forwards the request to 'Remote' radius server for processing. Thus, 'Local' radius acts as a client for 'Remote' radius. When 'Remote' radius responds, the 'Local' receives the response, and passes it back to the NAS. The copy of the request from the server's queue serves to determine which NAS originated the request. Before passing the request back to the NAS, radius removes from it the information, specific for 'Remote' site, such as `Framed-IP-Address`, `Framed-Netmask`, etc. Only the attributes marked with 'propagate' flag (see Section 2.1 [Attributes], page 17) are passed back to the NAS. After removing site-specific attributes, 'Local' radius passes the request through its user profiles (see Section 2.4.4 [User Profiles], page 25) to insert any local site-specific information that might be needed. Finally, it passes the reply back to the NAS.

The accounting requests are processed in the similar manner, except that no attribute filtering takes place, as the accounting responces do not carry any A/V pairs.

This example illustrates the simplest *proxy chain*, consisting of only two servers. The proxy chains may consist of several servers. In our example, the 'Remote' radius server may also act as a proxy and forward the request to still another radius server, etc.

*Note*, that when the accounting request passes through a chain of forwarding servers, the accounting records are stored on all servers in the chain.

#### 2.4.1.2 Realms

Radius server determines which server a request must be forwarded to by request's *authentication realm*. There are three kinds of realms:

1. A *Named Realm* is the part of a user name following the 'at sign' ('@'). For example, if the user name is 'jsmith@this.net', then 'this.net' is the realm.

   The named realms can be cascaded, e.g. a request with user name 'jsmith@this.net@remote.net' will first be forwarded to the radius server of realm 'remote.net', which in turn will forward it to 'this.net'.
2. A *Default Realm* defines the server to which the requests for realms, not mentioned explicitly in the configuration, are forwarded.
3. An *Empty Realm* defines the server to which the requests *without* explicit named realm are forwarded. If the configuration does not define the empty realm, such requests are processed by the server itself.

GNU Radius keeps the information about the realms it serves in configuration file 'raddb/realms' (see Section 4.8 [realms file], page 49).

### 2.4.2 Hints

*User-name hints* are special rules that modify the incoming request depending on user name and his credentials. Hints are stored as a list of *Matching Rules* (see Section 2.3 [Matching Rule], page 21). Upon receiving a request, `radiusd` scans the hints entries sequentially, comparing each rule's *Label* with the value of `User-Name` attribute from the request. If they coincide, then `radiusd` appends the contents of the rule's RHS to the request pair-list.

Both user names must match exactly in order for a hint to take effect, unless the hint's checklist contains either `Prefix` or `Suffix` attribute. The special name '`DEFAULT`' or '`DEFAULT%d`' (where %d denotes any decimal number), used as hint's *Label* match any username.

The two special attributes, `Prefix` and `Suffix`, may be used in LHS to restrict the match to a specified part of a user-name. Both are string attributes. The `Prefix` instructs radius to accept the hint only if the username begins with the given prefix. Similarly, `Suffix` instructs radius to accept the hint only if the username ends with the given suffix. A hint may contain both `Prefix` and `Suffix` attributes.

In addition to these two attributes, a hint's LHS may contain `User-ID` and `Group` attributes.

The following attributes, when used in a hint's RHS have special meaning. They are not appended to the request pair-list. Instead, they are removed after completing their function:

`Fall-Through`
: If this attribute is present, and is set to `Yes`, `radiusd` continues scanning the hints, after processing current entry. This allows for applying several hints to a single packet.

`Rewrite-Function`
: If this attribute is present, the specified rewrite function is invoked.

`Replace-User-Name`
: The value of this attribute is expanded (see Section 4.14 [Macro Substitution], page 60), and replaces the value of `User-Name` attribute from the request.

The hints rules are defined in '`raddb/hints`' file (see Section 4.6 [hints file], page 48).

### 2.4.3 Huntgroups

Huntgroups are special rules, that allow to alter processing of incoming requests, depending on NAS IP address and port number they come from. The rules are stored as a list of *Matching Rules* (see Section 2.3 [Matching Rule], page 21). Upon receiving a request, `radiusd` scans this list sequentially until it finds an entry, such that the conditions set forth in its LHS are matched by the request. If such an entry is found, `radiusd` verifies that the request meet the conditions described by RHS. If it does not, the request is rejected. In short, a huntgroup requires that any request matching its LHS must match also its RHS. The *Label* part of the rule is not used in comparisons, instead it is used to label huntgroups. All entries with the same label form a single huntgroup. The special attribute `Huntgroup-Name` can be used to request a match against a particular huntgroup (see Section 12.3.9 [Huntgroup-Name], page 125).

The huntgroups rules are defined in '`raddb/huntgroups`' file (see Section 4.7 [huntgroups file], page 48).

### 2.4.4 User Profiles

*User Profiles* are the per-user matching rules (see Section 2.3 [Matching Rule], page 21). Any incoming authentication request is compared with the *User Profiles* after it has passed both *Hints* and *Huntgroups*. `Radiusd` selects from the *User Profiles* those rules, whose *Label* matches the value of `User-Name` attribute from the incoming request.

The selected profiles form the list of authentication rules for the request. In order for a profile to be selected, its label must either coincide literally with the `User-Name` value, or it must be one of special labels `DEFAULT` and `BEGIN`. The rules in the authentication list are ordered as follows: first go all the profiles with `BEGIN` label, they are followed by the profiles, whose labels match the `User-Name` literally, and, finally, these are followed by rules labeled with `DEFAULT`.[1] Within each of the three sub-lists the rules preserve the order in which they appear in '`raddb/users`' file. Once the list is constructed, it is scanned sequentially until the rule is found, whose LHS matches the incoming request. If no such rule is found, then the authentication fails. Otherwise, the contents of its RHS is appended to *Reply List* being constructed. If the RHS of the matched rule contains attribute `Fall-Through` with the value `Yes`, the matching continues. When the list is exhausted, the authentication result is sent back to the NAS along with the A/V pairs collected in the *Reply List*.

The user profiles are defined in '`raddb/users`' file (see Section 4.9 [users file], page 50).

[1] For compatibility with other implementations of radius, GNU Radius treats profile labels in the form `DEFAULT%d`, where `%d` represents a decimal number, in the same way it treats `DEFAULT` labels. The same applies to `BEGIN` labels.

# 3 How to Start the Daemon.

When started `radiusd` uses the configuration values from the following sources (in order of increasing precedence):

- Compiled-in defaults
- '`raddb/config`' file.
- Command line arguments

Whenever a command line options has its equivalent in config file the use of this equivalent should be preferred (see Section 4.1 [config file], page 31).

The following command line options are accepted:

'`-A`'
'`--log-auth-detail`'
: Enable detailed authentication logging. When this option is specified each authentication request is logged to the file '`radacct/NASNAME/detail.auth`', where NASNAME is replaced by the short name of the NAS from '`raddb/naslist`' Chapter 1 [Naming Conventions], page 15.

  Config file equivalent: `auth { detail yes; };`.

'`-a DIR`'
'`--acct-directory DIR`'
: Specify accounting directory.

  Config file equivalent: `option { acct-dir DIR; };`.

'`-b`'
'`--dbm`'
: Enable DBM support.

  Config file equivalent: `usedbm yes;`.

'`-d DIR`'
'`--config-directory DIR`'
'`--directory D`'
: Specify alternate configuration directory. Default is '`/usr/local/etc/raddb`'.

'`-f`'
'`--foreground`'
: Stay in foreground. We recommend to use it for debugging purposes only.

'`-i IP`'
'`--ip-address`'
: Specifies the IP address `radiusd` will listen on. If this option is not specified, the program will listen on all IP addresses, assigned to the machine it runs on.

  Config file equivalent: `option { source-ip IP; };`.

  *Please note*, that `listen` statement in '`raddb/config`' provides a better control over IP addresses to listen on (See Section 4.1.3 [auth], page 35, and see Section 4.1.4 [acct], page 36).

'`-L`'
'`--license`'
: Display GNU General Public License and exit.

'-l DIR'
'--logging-directory DIR'
: Specify alternate logging directory.

  Config file equivalent: `option { log-dir DIR; };`.

'-mb'
'--mode b'
: "Builddbm" mode. Builds a DBM version of a plaintext users database. Section 10.9 [Builddbm], page 100.

'-mc'
'--mode c'
: Check configuration files and exit. All errors are reported via usual log channels.

'-mt'
'--mode t'
: Test mode. In this mode `radiusd` starts an interactive interpreter which allows to test various aspects of its configuration.

'-n'
'--auth-only'
: Process only authentication requests.

'-p PORTNO'
'--port PORTNO'
: Listen the UDP port PORTNO. The accounting port is computed as *PORTNO* + 1.

'-P DIR'
'--pid-file-dir DIR'
: Specifies the alternate path for the pidfile.

'-S'
'--log-stripped-names'
: Log usernames stripped off any prefixes/suffixes.

  Config file equivalent: `auth { strip-names yes };`.

'-s'
'--single-process'
: Run in single process mode. This is for debugging purposes only. We strongly recommend *against* using this option. Use it only when absolutely necessary.

'-v'
'--version'
: Display program version and compilation options.

'-x DEBUG_LEVEL'
'--debug DEBUG_LEVEL'
: Set debugging level. *DEBUG_LEVEL* is a comma-separated list of assignments in the forms

```
MODULE
MODULE = LEVEL
```

  where *MODULE* is the module name or any non-ambiguous assignment thereof, LEVEL is the debugging level in the range 0-100. Chapter 8 [Debugging], page 73

Config file equivalent:

```
logging {
        category debug {
                level DEBUG_LEVEL;
        };
};
```

'-y'
'--log-auth'
: Log authentications. With this option enabled, Radius will log any authentication attempt into its logfile Chapter 7 [Logging], page 71.

  Config file equivalent: `logging { category auth { detail yes; }; };` .

'-z'
'--log-auth-pass'
: Log passwords along with authentication information. *Do not use* this option. It is *very* insecure, since all users' passwords will be echoed in the logfile. This option is provided only for debugging purposes.

  Config file equivalent:

```
logging {
        category auth {
                print-pass yes;
        };
};
```

See Section 4.1 [config file], page 31.

# 4 Radius Configuration Files

This chapter describes the configuration files used by GNU Radius package.

These files are normally found in /usr/local/etc/raddb directory, which is defined at configuration time, although their location can be specified at runtime. In the discussion below we will refer to this directory by 'raddb'. See Chapter 1 [Naming Conventions], page 15.

## 4.1 Run-Time Configuration Options — 'raddb/config'

radiusd uses the configuration values from the following sources (in order of increasing precedence):

1. Compiled-in defaults
2. 'raddb/config' file.
3. Command line arguments

This order of precedence applies only on startup. When re-reading of the configuration is initiated either by SIGHUP signal or by SNMP channel any changes in the config file take precedence over command line arguments, since 'raddb/config' is the only way to change configuration of the running program.

This chapter discusses the 'raddb/config' file in detail.

The 'raddb/config' consists of statements and comments. Statements end with a semicolon. Many statements contain a block of sub-statements which also terminate with a semicolon.

Comments can be written in shell, C, or C++ constructs, i.e. any of the following represent a valid comment:

```
# A shell comment
/* A C-style
 * multi-line comment
 */
// A C++-style comment
```

These are the basic statements:

### 4.1.1 option block

## Syntax:

```
option {
        [ source-ip number ; ]
        [ max-requests number ; ]
        [ exec-program-user string ; ]
        [ username-chars string ; ]
        [ log-dir string ; ]
        [ acct-dir string ; ]
} ;
```

## Usage

The `option` block defines the global options to be used by `radiusd`.

## Numeric statements

`source-ip`
: Sets the source IP address. When this statement is not present, the IP address of the first available network interface on the machine will be used as source.

`max-requests`
: Sets the maximum number of the requests in queue.

## String statements

`exec-program-user`
: Sets effective user id for the programs executed as a result of `Exec-Program` and `Exec-Program-Wait`. The effective group id will be retrieved from the '`/etc/passwd`' entry for the given user.

`username-chars`
: Determines characters that are valid within a username. The alphanumeric characters are always allowed in a username, it is not necessary to specify them in this statement. By default the following characters are allowed in a username: '`.-_!@#$%^&\/"`'.

`log-dir` Specifies the logging directory.

`acct-dir` Specifies the accounting directory.

### 4.1.2 `logging` block

## Syntax:

```
logging {
        [ category category_spec {
                [ channel channel_name ; ]
                [ print-auth bool ; ]
                [ print-pass bool ; ]
                [ print-failed-pass bool ; ]
                [ level debug_level ; ]
        } ; ]
        [ channel channel_name {
               (  file string ;
                | syslog facility . priority ; )
                [ print-pid bool ; ]
                [ print-category bool ; ]
                [ print-cons bool ; ]
                [ print-level bool ; ]
                [ print-priority bool ; ]
```

```
                };  ]
        } ;
```

## Usage

The `logging` statement describes the course followed by `radiusd`'s logging information.

### 4.1.2.1 `category` statement

Each line of logging information generated by `radiusd` has an associated *category*. The `logging` statement allows each category of output to be controlled independently of the others. The logging category is defined by *category name* and a *severity*. *category name* determines what part of radiusd daemon is allowed to send its logging information to this channel. It can be any of `main`, `auth`, `acct`, `proxy`, `snmp`. *priority* determines the minimum priority of the messages displayed by this channel. The priorities in ascending order are: `debug`, `info`, `notice`, `warn`, `err`, `crit`, `alert`, `emerg`.

The full category specification, *category_spec*, can take any of the following three forms:

category_name
: Print the messages of given category.

priority
: Print messages of all categories, abridged by given priority. If the priority is prefixed with '`=`', only messages with given priority will be displayed. If it is prefixed with '`!`', the messages with priority other than the specified will be displayed. Otherwise, the messages with priorities equal to or greater than the specified will be displayed.

category_name . priority
: Print the messages of given category, abridged by given priority. The priority may be prefixed with either '`=`' or '`!`' as described above.

Additional category options valid for `auth` category are:

`print-auth`
: Log individual authentications.

`print-pass`
: Include passwords for successful authentications. It is *very* insecure, since all users' passwords will be echoed in the logfile. This option is provided only for debugging purposes.

`print-failed-pass`
: Include passwords for failed authentications.

### 4.1.2.2 `channel` statement

Channels represent methods for recording logging information. Each channel has a unique name, and any categories which specify that name in a `channel` statement will use that channel.

`radiusd` can write logging information to files or send it to syslog. The `file` statement sends the channel's output to the named file (see Chapter 1 [Naming Conventions], page 15).

The `syslog` statement sends the channel's output to syslog with the specified facility and severity.

Channel options modify the data flowing through the channel:

`print-pid`
: Add the process ID of the process generating the logging information.

`print-cons`
: Also send the logging information to the system console.

`print-category`
: Add the category name to the logging information.

`print-priority`
`print-level`
: Add the priority name to the logging information.

#### 4.1.2.3 Example of the `logging` statement

```
logging {
        channel default {
                file "radius.log";
                print-category yes;
                print-priority yes;
        };
        channel info {
                file "radius.info";
                print-pid yes;
                print-cons yes;
                print-priority yes;
        };
        channel notice {
                syslog auth.notice;
        };

        category auth {
                print-auth yes;
                print-failed-pass yes;
        };
        category notice {
                channel notice;
        };
        category info {
                channel info;
        };
        category debug {
                channel info;
                level radiusd=1,files;
        };

        category *.!debug {
```

```
                channel default;
        };
};
```

### 4.1.3 auth statement

## Syntax:

```
auth {
        [ listen addr-list ; ]
        [ port number ; ]
        [ spawn bool ; ]
        [ max-requests number ; ]
        [ time-to-live number ; ]
        [ request-cleanup-delay number ; ]
        [ detail bool ; ]
        [ strip-names bool ; ]
        [ checkrad-assume-logged bool ; ]
        [ password-expire-warning number ; ]
} ;
```

## Usage:

The `auth` statement configures the parameters of the authentication service.

### listen statement

This statement determines on which addresses radiusd will listen for incoming authentication requests. Its argument is a comma-separated list of items in the form *ip*:*port-number*. *ip* can be either an IP address in familiar "dotted-quad" notation or a hostname. :*port-number* part may be omitted, in which case the default authentication port is assumed.

If the `listen` statement is omitted, radiusd will accept incoming requests from any interface on the machine.

## Numeric statements

`port` Sets the number of UDP port to listen on for the authentication requests.

`max-requests`
: Sets the maximum number of authentication requests in the queue. Any surplus requests will be discarded.

`time-to-live`
: Sets the request time-to-live in seconds. The time-to-live is the time to wait for the completion of the request. If the request job isn't completed within this interval of time it is cleared, the corresponding child process killed and the request removed from the queue.

`request-cleanup-delay`
: Sets the request cleanup delay in seconds, i.e. determines how long will the completed authentication request reside in the queue.

`password-expire-warning`
: Sets the time interval for password expiration warning. If user's password expires within given number of seconds, radiusd will send a warning along with authentication-acknowledge response. Default is 0.

### Boolean statements

`spawn`
: Determines if `radiusd` should spawn a child to process the request.

`detail`
: When set to true, `radiusd` will produce the detailed log of each received packet in the file '`radacct/NASNAME/detail.auth`'. (see Chapter 1 [Naming Conventions], page 15).

`strip-names`
: Determines whether `radiusd` should strip any prefixes/suffixes off the username before logging.

`checkrad-assume-logged`
: `radiusd` consults the value of this variable when the NAS does not responds to checkrad queries (see Section 5.9 [Checking Simultaneous Logins], page 66). If this variable is set to `yes`, the daemon will proceed as if the NAS returned "yes", i.e. it will assume the user is logged in. Otherwise `radiusd` assumes the user *is not* logged in.

## 4.1.4 acct statement

### Syntax:

```
acct {
        [ listen addr-list ; ]
        [ port number ; ]
        [ spawn bool ; ]
        [ detail bool; ]
        [ max-requests number ; ]
        [ time-to-live number ; ]
        [ request-cleanup-delay number ; ]
} ;
```

### Usage:

The `acct` statement configures the parameters of the accounting service.

### listen statement

This statement determines on which addresses radiusd will listen for incoming accounting requests. Its argument is a comma-separated list of items in the form *ip:port-number*. *ip* can be either an IP address in familiar "dotted-quad" notation or a hostname. *:port-number* part may be omitted, in which case the default accounting port is assumed.

If the `listen` statement is omitted, radiusd will accept incoming requests from any interface on the machine.

### Numeric statements

`port` Sets the port number to listen for the authentication requests.

`max-requests`
: Sets the maximum number of accounting requests in the queue. Any surplus requests will be discarded.

`time-to-live`
: Sets the request time-to-live in seconds. The time-to-live is the time to wait for the completion of the request. If the request job isn't completed within this interval of time it is cleared, the corresponding child process killed and the request removed from the queue.

`request-cleanup-delay`
: Sets the request cleanup delay in seconds, i.e. determines how long will the completed account request reside in the queue.

### Boolean statements

`spawn` Determines if `radiusd` should spawn a child to process the request.

`detail` When set to `false`, disables detailed accounting (see Section 6.2 [Detailed Request Accounting], page 67).

## 4.1.5 `proxy` statement

### Syntax:

```
proxy {
        [ max-requests number ; ]
        [ request-cleanup-delay number ; ]
} ;
```

### Usage:

The `proxy` statement configures the parameters of the proxy service.

### Numeric statements

`max-requests`
: Sets the maximum number of accounting requests in the queue. Any surplus requests will be discarded.

`request-cleanup-delay`
: Sets the request cleanup delay in seconds, i.e. determines how long will the completed account request reside in the queue.

### 4.1.6 usedbm statement

#### Syntax:

```
usedbm ( yes | no ) ;
```

#### Usage

The usedbm statement determines whether the DBM support should be enabled.

no
: Do not use DBM support at all.

yes
: Use only the DBM database and ignore 'raddb/users'.

### 4.1.7 snmp statement

#### Syntax:

```
snmp {
        [ port portno ; ]
        [ spawn bool ; ]
        [ max-requests number ; ]
        [ time-to-live number ; ]
        [ request-cleanup-delay number ; ]
        [ ident string ; ]
        [ community name ( rw | ro ) ; ]
        [ network name network [ network ... ] ; ]
        [ acl {
                [ allow network_name community_name ; ]
                [ deny network_name ; ]
        } ; ]
};
```

#### Usage

The snmp statement configures the SNMP service.

#### Numeric statements

port
: Sets the port number to listen for the SNMP requests.

max-requests
: Sets the maximum number of SNMP requests in the queue. Any surplus requests will be discarded.

time-to-live
: Sets the request time-to-live in seconds. The time-to-live is the time to wait for the completion of the request. If the request job isn't completed within this interval of time it is cleared, the corresponding child process killed and the request removed from the queue.

`request-cleanup-delay`
: Sets the request cleanup delay in seconds, i.e. determines how long will the completed SNMP request reside in the queue.

### Boolean statements

`spawn` Determines if `radiusd` should spawn a child to process the SNMP request.

### String statements

`ident` Sets the SNMP server identification string.

### Community and network definitions

`community` *name* `( rw | ro )`
: Defines the community *name* as read-write (`rw`) or read-only (`ro`).

`network` *name network* `[` *network* `... ]`
: Groups several networks or hosts under one logical network name.

### Access-Control List definitions

`allow` *network_name community_name*
: allow hosts from the group *network_name* access to community *community_name*.

`deny` *NETWORK_NAME*
: Deny access to SNMP service from any host in the group *network_name*.

## 4.1.8 `guile` statement

The `guile` statement allows to configure server interface with Guile.

### Syntax

```
guile {
        [ debug bool ; ]
        [ load-path string ; ]
        [ load string ; ]
};
```

### Usage

### Boolean statements

`debug` When set to yes, enables debugging evaluator and backtraces on Guile scripts.

## String statements

`load-path`
: Add specified pathname to `%load-path` variable.

`load`
: Load the specified source file on startup.

For the detailed description of Guile extensions interface, See Section 9.2 [Guile], page 85.

### 4.1.9 `message` statement

The `message` statement allows to set up the messages that are returned to the user with authentication-response packets.

## Syntax

```
message {
        [ account-closed string ; ]
        [ password-expired string ; ]
        [ password-expire-warning string ; ]
        [ access-denied string ; ]
        [ realm-quota string ; ]
        [ multiple-login string ; ]
        [ second-login string ; ]
        [ timespan-violation string ; ]
};
```

All variables in `message` block take a string argument. In *string* you can use the usual C backslash notation to represent non-printable characters. The use of %C{} and %R{} sequences is also allowed (see Section 4.14 [Macro Substitution], page 60).

## String statements

`account-closed`
: This message will be returned to the user whose account is administratively closed.

`password-expired`
: This message will be returned to the user whose password has expired.

`password-expire-warning`
: This is a warning message that will be returned along with an authentication-acknowledge packet for the user whose password will expire in less than *n* seconds. The value of *n* is set by `password-expire-warning` variable in `auth` block. See Section 4.1.3 [auth], page 35. In this string, you can use the %R{Password-Expire-Days} substitution, to represent the actual number of *days* left to the expiration date. The default is

```
Password Will Expire in %R{Password-Expire-Days} Days\r\n
```

`access-denied`
: This message is returned to the user who supplies an incorrect password or a not-existent user-name as his authentication credentials.

`realm-quota`
: This message is returned when the user is trying to log in using a realm, and number of users that are currently logged in from this realm reaches maximum value. For a description of realms, See Section 2.4.1.2 [Realms], page 23.

`multiple-login`
: This message is returned to the user, who has logged in more than allowed number of times. For description of how to set the maximum number of concurrent logins, see Section 12.3.10 [Simultaneous-Use], page 126.

`second-login`
: This is a special case of `multiple-login`, which is used when the user's login limit is 1.

`timespan-violation`
: This message is returned to the user who is trying to login outside of allowed time interval. For description of how to limit user's login time, see Section 12.3.19 [Login-Time], page 131.

## 4.2 Dictionary of Attributes — 'raddb/dictionary'

The dictionary file '`raddb/dictionary`' defines the symbolic names for radius attributes and their values (see Section 2.1 [Attributes], page 17). The file consists of a series of statements. Each statement occupies one line.

In the detailed discussion below we use the following meta-syntactic characters:

*number*
: Denotes a decimal, octal or hexagesimal number. Usual C conventions are honored, i.e. if *number* starts with '`0x`' or '`0X`' it is read as a hex number, if it starts with '`0`' it is read as an octal number, otherwise it is read as a decimal one.

*type*
: Denotes an attribute type. These are valid attribute types:

  `string`
  : A string type.

  `integer`
  : An integer type.

  `ipaddr`
  : IP address in a dotted-quad form.

  `date`
  : A date in the format: "MON DD CCYY", where MON is the usual three-character abbreviation, DD is day of month (1-31), CCYY is the year, including the century.

There are following kinds of statements:

### 4.2.1 Comments

Comments are introduced by a pound sign ('`#`'). Everything starting from the first occurrence of '`#`' up to the end of line is ignored.

### 4.2.2 $INCLUDE Statement

## Syntax

```
$INCLUDE 'filename'
```

## Usage

The `$INCLUDE` statement causes the contents of the file '`filename`' to be read in and processed. The file is looked up in the Radius database directory. See Chapter 4 [Configuration Files], page 31.

### 4.2.3 VENDOR Statement

## Syntax

VENDOR Vendor-Name *number*

## Usage

A `VENDOR` statement defines the symbolic name for a Vendor-Id. This name can subsequently be used in `ATTRIBUTE` statements to define Vendor-Specific attribute translations. See Section 12.1.19 [Vendor-Specific], page 116.

## Example

```
VENDOR          Livingston      307
```

### 4.2.4 ATTRIBUTE statement

## Syntax

ATTRIBUTE *name* *number* *type* [*vendor* [*flags*]]

## Usage

The `ATTRIBUTE` statement defines the translation for an attribute. Its parts have the following meaning:

*name* The attribute name.

*number* The attribute ID (number).

*type* The attribute type.

*vendor* Vendor name for vendor-specific attributes. For usual attributes this field is empty or contains a dash ('`-`').

*flags* Flags, defining attribute properties (see Section 2.1 [Attributes], page 17).

The *attribute property flags* consist of a sequence of letters, whose meaning is determined by the following rules:[1]

[1] The *flags* are optional for compatibility with previous versions of GNU Radius. If they are omitted, the default is '`[LRLRLR]+`'

1. The attribute usage is described by three pairs of symbols, enclosed in square brackets. Each pair describes how the attribute can be used in each of three configuration files. The first pair corresponds to 'raddb/users', the second one corresponds to 'raddb/hints', and the third one corresponds to 'raddb/huntgroups'. Within each pair, the letter 'L' in first position means that the attribute is allowed in LHS of a rule. The letter 'R' in second position means that the attribute is allowed in RHS of a rule. The absense of any of these letters is indicated by dash ('-'). Thus, the following usage specification:

   ```
   [L--RLR]
   ```

   means that the attribute may be used in LHS of a rule in 'raddb/users', in RHS of a rule in 'raddb/hints', and in both sides of a rule in 'raddb/huntgroups'.
2. The attribute additivity is described by one of the following letters:

   = Additivity = Replace

   + Additivity = Append

   N Additivity = None
3. The presence of letter 'P' in property flags raises the propagation bit.

## Example

```
ATTRIBUTE  Service-Type  6 integer - [LR-RLR]=P
```

This statement assigns the translation string 'Service-Type' to the attribute number 6. It allows the use of this attribute in any part of matching rules, except in LHS of a 'raddb/hints' rule. The additivity of Service-Type is set to 'Replace'. The attribute will be propagated through the proxy chain.

### 4.2.5 VALUE Statement

## Syntax

```
VALUE  Attribute-Translation      Value-Translation       number
```

## Usage

The VALUE statement assigns a translation string to a given value of an integer attribute. Attribute-Translation specifies the attribute and the Value-Translation specifies the name assigned to the value *number* of this attribute.

## Example

The following assigns the translation string 'Login-User' to the value 1 of the attribute 'Service-Type'.

```
VALUE           Service-Type            Login-User            1
```

## 4.3 Clients List — 'raddb/clients'

The '`raddb/clients`' lists NASes which are allowed to make authentication requests. As usual, the '`#`' character introduces a comment. Each record in the file consists of two fields, separated by whitespace. The fields are:

NAS name
: Specifies a hostname or IP address of the NAS.

Key
: Lists the encryption key shared between the server and this NAS.

### 4.3.1 Example of 'clients' file

```
# This is a list of clients which are allowed to make authentication
# requests.
# Each record consists of two fields:
#       i.  Valid hostname.
#       ii. The shared encryption key for this hostname.
#
#Client Name           Key
#----------------      -------------------
myhost.dom.ain         guessme
merlin                 emrys
11.10.10.10            secRet
```

## 4.4 NAS List — 'raddb/naslist'

The '`raddb/naslist`' file contains a list of NASes known to the Radius server. Each record in the file consist of three fields:

NAS name
: Specifies a hostname or IP address of the NAS. The word '`DEFAULT`' may be used in this field to match any NAS.[2]

Short Name
: This field defines a short name under which this NAS will be listed in logfiles. The short name is also used as a name of the subdirectory where the detailed logs are stored.

Type
: Specifies the type of this NAS. Using this value `radiusd` determines the way to query NAS about the presence of a given user on it (see Section 5.9 [Checking Simultaneous Logins], page 66). The two special types: '`true`' and '`false`', can be used to disable NAS querying. When the type field contains '`true`', `radiusd` assumes the user is logged in to the NAS, when it contains '`false`', `radiusd` assumes the user *is not* logged in. Otherwise, the type is used as a link to '`nastypes`' entry (see Section 4.5 [nastypes file], page 45).

Arguments
: Additional arguments describing the NAS. Multiple arguments must be separated by commas. No intervening whitespace is allowed in this field.

---

[2] Logins from DEFAULT NASes are not reported by213.130.0.5 `raduse`, neither are they reflected in SNMP variables.

There are two groups of nas arguments: *nas-specific* arguments and *nas-querying* arguments. *Nas-specific* arguments are used to modify a behavior of `radiusd` when sending or receiving the information to or from a particular NAS.

*Nas-querying* arguments control the way `radiusd` queries a NAS for confirmation of a user's session (see Section 5.9 [Checking Simultaneous Logins], page 66). These arguments override the ones specified in '`nastypes`' and can thus be used to override the default values.

The *nas-specific* arguments currently implemented are:

broken_pass
: This is a boolean argument that controls the encryption of user passwords, longer than 16 octets. By default, `radiusd` uses method specified by RFC 2865. However some NASes, most notably MAX Ascend series, implement a broken method of encoding long passwords. This flag instructs `radiusd` to use broken method of password encryption for the given NAS.

For the list of nas-querying arguments, See Section 4.5 [Full list of allowed arguments], page 45.

### 4.4.1 Example of '`naslist`' file

```
# raddb/naslist: contains a list of Network Access Servers
#
# Each record consists of following fields:
#
#       i.      A valid hostname or IP address for the client.
#       ii.     The short name to use in the logfiles for this NAS.
#       iii.    Type of device. Valid values are 'true', 'false' and
#               those defined in raddb/nastypes file.

# NAS Name              Short Name      Type
#----------------       ----------      ----
myhost.dom.ain          myhost          unix
merlin                  merlin          max
11.10.10.10             arthur          livingston
```

## 4.5 NAS Types — '`raddb/nastypes`'

The '`raddb/nastypes`' file describes the ways to query NASes about active user sessions.

### 4.5.1 Syntax of '`raddb/nastypes`'

## Syntax

Each record consists of three fields separated by any amount of whitespace. The fields are:

Type
: Type of the NAS which is described in this record.

Method
: Method to use to query a NAS of given type.

Arguments
: Arguments to pass to this method. Each argument is a pair *arg*=*value*, where *arg* is its name and *value* is a value assigned to it. The list of predefined argument names follows. *Please note*, that no intervening whitespace is allowed in this field.

## Methods

Version 0.96 of GNU Radius supports two querying methods: finger and snmp.

## Arguments

In the discussion below *n* means numeric and *s* string value.

The following arguments are predefined:

### Common for all methods

function=*s*
: Specifies the check function to use with this method (see Section 9.1.5 [Login Verification Functions], page 80). This argument must be present. For description of how this function is applied, see Section 5.9 [Checking Simultaneous Logins], page 66.

port=*n*
: Use port number *n* instead of the default for the given method.

### Method snmp

password=*s*
: Use community *s* instead of the default. This argument must be present.

retries=*n*
: Retry *n* times before giving up.

timeout=*n*
: Timeout *n* seconds on each retry.

### Method finger

timeout=*n*
: Give up if the NAS does not respond within *n* seconds.

notcp
tcp=0
: Disable the use of T/TCP for hosts with a broken TCP implementation.

arg=*subst*
: Send *subst* to finger, instead of username. *subst* must be one of *macro variables*, described below.

### Macro variables

The following macro-variables are recognized and substituted when encountered in the *value* pair of an argument:

'%u' Expands to username.

'%s' Expands to session id.

'%d' Expands to session id converted to decimal representation.

'%p' Expands to port number.

'%P' Expands to port number + 1.

## 4.5.2 Example of nastypes file.

*Please note*, that in the following example the long lines are broken into several lines for readability.

```
# Type     Method         Args
# ----     ------         ----
unix       finger      function=check_unix
max-f      finger      function=check_max_finger
max        snmp        oid=.1.3.6.1.4.1.529.12.3.1.4.%d,
                       function=check_snmp_u
as5300-f   finger      function=check_as5300_finger
as5300     snmp        oid=.1.3.6.1.4.1.9.9.150.1.1.3.1.2.%d,
                       function=check_snmp_u
livingston snmp        oid=.1.3.6.1.4.1.307.3.2.1.1.1.5.%P,
                       function=check_snmp_s
```

## 4.5.3 Standard NAS types

The 'nastypes' shipped with version 0.96 of GNU Radius defines following NAS types:

unix — UNIX boxes running Finger

This type suits for UNIX boxes running finger service able to return information about dial-up users active on them. To enable finger checking of a unix host add following to your 'naslist' file:

```
#Hostname       Shortname   Type
#--------       ---------   ----
nas.name        T           unix
```

max-f — MAX Ascend with Finger

Use this type if you have MAX Ascend terminal server that answers finger queries. The 'naslist' entry for such NAS will look like:

```
#Hostname       Shortname   Type  Flags
#--------       ---------   ----  -----
nas.name        T           max-f broken_pass
```

*Please note* the use of `broken_pass` flag. It is needed for most MAX Ascend servers (see Section 4.4 [naslist file], page 44).

max — MAX Ascend, answering SNMP
: Use this type if you have MAX Ascend terminal server that answers SNMP queries. The 'naslist' entry for such NAS will look like:

```
#Hostname       Shortname   Type  Flags
#--------       ---------   ----  -----
nas.name        T           max-f broken_pass,community=comm
```

Replace *comm* with your actual SNMP community name.

as5300-f — Cisco AS5300 running finger

as5300 — Cisco AS5300 answering SNMP

livingston — Livingston Portmaster
: Type `livingston` queries portmaster using SNMP.

## 4.6 Request Processing Hints — 'raddb/hints'

The 'raddb/hints' file is used to modify the contents of the incoming request depending on the username. For a detailed description of this, See Section 2.4.2 [Hints], page 23.

The file contains data in *Matching Rule* format (see Section 2.3 [Matching Rule], page 21).

The only attributes that can be used in the check list are:

- `Suffix`
- `Prefix`
- `Group`
- `User-ID`

### 4.6.1 Example of 'hints' file

```
## If the username starts with 'U', append the UUCP hint
DEFAULT         Prefix = "U", Strip-User-Name = No
                Hint = "UUCP"
## If the username ends with '.slip', append the SLIP service data
## and remove the suffix from the user name.
DEFAULT         Suffix = ".slip",
                   Strip-User-Name = Yes
                Hint = "SLIP",
                   Service-Type = Framed-User,
                   Framed-Protocol = SLIP
```

## 4.7 Huntgroups — 'raddb/huntgroups'

The 'raddb/huntgroups' contains the definitions of the huntgroups. For a detailed description of huntgroup concept, See Section 2.4.3 [Huntgroups], page 24.

The file contains data in *Matching Rule* format (see Section 2.3 [Matching Rule], page 21).

### 4.7.1 Example of 'huntgroups' file.

```
## This defines the packet rewriting function for the server 11.10.10.11
DEFAULT NAS-IP-Address = 11.10.10.11, Rewrite-Function = "max_fixup"
        NULL
```

## 4.8 List of Proxy Realms — 'raddb/realms'

The 'raddb/realms' file lists remote Radius servers that are allowed to communicate with the local Radius server (see Section 2.4.1 [Proxying], page 22).

Each record consists of up to three fields, separated by whitespace. Two of them are mandatory. The fields are:

Realm name
: Specifies the name of the realm being defined, i.e. part of the login name after the '@' symbol. The special realm name 'NOREALM' defines the empty realm, the name 'DEFAULT' defines the default realm (see Section 2.4.1.2 [Realms], page 23).

Remote server
: Specifies the remote server to which the requests for this realm should be forwarded. The syntax for this field is

  *servername*[:*auth-port*[:*acct-port*]]

  Optional *auth-port* and *acct-port* are the authentication and accounting port numbers. If *acct-port* is omitted, it is computed as *auth-port* + 1. If *auth-port* is omitted, the default authentication port number is used.

Flags (optional)

The flags meaningful in 'raddb/realms' are

strip
: Boolean value. Controls whether the realm name should be stripped off the username before forwarding the request to the remote server. Setting `strip` enables stripping, setting `nostrip` disables it. Default is to always strip user names.

quota=*num*
: Set maximum number of concurrent logins allowed from this realm to the given value (*num*).

### 4.8.1 Example of 'realms' file

### Example 1.

```
# Realm                 Remote server[:port]         flags
#----------------       ---------------------        --------
that.net                radius.that.net              nostrip
dom.ain                 server.dom.ain:3000          strip,quota=20
```

### Example 2.

```
# Realm                 Remote server[:port]            flags
#----------------       ---------------------           --------
NOREALM                 radius.server.net
that.net                radius.that.net                 nostrip
dom.ain                 server.dom.ain:3000             strip,quota=20
```

## 4.9 User Profiles — 'raddb/users'

File 'raddb/users' contains the list of *User Profiles*. For a description of its purpose, See Section 2.4.4 [User Profiles], page 25.

### 4.9.1 Example of 'users' file

```
## The following entry is matched when the user appends ``.ppp'' to his
## username when logging in.
## The suffix is removed from the user name, then the password is
## looked up in the SQL database.
## Users may log in at any time. They get PPP service.
DEFAULT Suffix = ".ppp",
                Auth-Type = SQL,
                Login-Time = "Al",
                Simultaneous-Use = 1,
                Strip-User-Name = Yes
        Service-Type = Framed-User,
                Framed-Protocol = PPP

## This is for SLIP users.
## This entry is matched when the auth request matches ``SLIP'' hint
DEFAULT Hint = "SLIP",
                Auth-Type = Mysql
        Service-Type = Framed-User
                Framed-Protocol = SLIP

## The following authenticates users using system passwd files.
## The users are allowed to log in from 7:55 to 23:05 on any weekday,
## except the weekend, and from 07:55 to 12:00 on Sunday.
## Only one login is allowed per user.
## The program telauth is used to further check the authentication
## information and provide the reply pairs
## Note the use of backslashes to split a long line.
DEFAULT Auth-Type = System,
                Login-Time = "Wk0755-2305,Su0755-1200",
                Simultaneous-Use = 1
        Exec-Program-Wait = "/usr/local/sbin/telauth \
                            %C{User-Name} \
                            %C{Calling-Station-Id} \
                            %C{NAS-IP-Address} \
                            %C{NAS-Port-Id}"
```

```
## This particular user is authenticated via PAM. He is presented a
## choice from 'raddb/menus/menu1' file.
gray    Auth-Type = Pam
        Menu = menu1
```

## 4.10 List of Blocked Users — 'raddb/access.deny'

The '`raddb/access.deny`' file contains a list of user names which are not allowed to log in via Radius. Each user name is listed on a separate line. As usual, the '`#`' character introduces an end-of-line comment.

## 4.11 SQL Configuration — 'raddb/sqlserver'

The '`raddb/sqlserver`' file configures the connection to SQL server.

The file uses simple line-oriented '*keyword* `---` *value*' format. Comments are introduced by '`#`' character.

The '`sqlserver`' statements can logically be subdivided into following groups: *SQL Client Parameters*, configuring the connection between SQL client and the server, *Authentication Server Parameters*, *Authorization Parameters*, and *Accounting server parameters*.

### 4.11.1 SQL Client Parameters

These parameters configure various aspects of connection between SQL client and the server.

`interface` *iface-type*
: Specifies the SQL interface to use. Currently supported values for *iface-type* are `mysql` and `postgres`. Depending on this, the default communication port number is set: it is 3306 for `interface mysql` and 5432 for `interface postgres`. Use of this statement is only meaningful when the package was configured with both '`--with-mysql`' and '`--with-postgres`' option.

`server` *string*
: Specifies the hostname or IP address of the SQL server.

`port` *number*
: Sets the SQL communicaton port number. It can be omitted if your server uses the default port.

`login` *string*
: Sets the SQL user login name.

`password` *password*
: Sets the SQL user password.

`keepopen` *bool*
: Specify whether `radiusd` should try to keep the connection open. When set to no (the default), `radiusd` will open new connection before the transaction and close it right after finishing it. We recommend setting `keepopen` to `yes` for heavily loaded servers, since opening the new connection can take a substantial amount of time and slow down the operation considerably.

`idle_timeout` *number*
: Set idle timeout in seconds for an open SQL connection. The connection is closed if it remains inactive longer that this amount of time.

### 4.11.2 Authentication Server Parameters

These parameters configure the SQL authentication. The general syntax is:

`doauth` *bool*
: When set to `yes`, enables authentication via SQL. All `auth_` keywords are ignored if `doauth` is set to `no`.

`auth_max_connections` *bool*
: Specifies the maximum number of authentication SQL connections to keep open. This parameter is ignored if `keepopen` is set to `no`.

`auth_db` *string*
: Specifies the name of the database containing authentication information.

`auth_query` *string*
: Specifies the SQL query to be used to obtain user's password from the database. The query should return exactly one string value — the password.

`group_query` *string*
: Specifies the query that retrieves the list of user groups the user belongs to. This query is used when `Group` or `Group-Name` attribute appears in the LHS of a user's or hint's profile.

### Example of Authentication Server Parameters

Let's suppose the authentication information is kept in the tables `passwd` and `groups`.

The `passwd` table contains user passwords. A user is allowed to have different passwords for different services. The table structure is:

```
CREATE TABLE passwd (
  user_name           varchar(32) binary default '' not null,
  service             char(16) default 'Framed-PPP' not null,
  password            char(64)
);
```

Additionally, the table `groups` contains information about user groups a particular user belongs to. Its structure is:

```
CREATE TABLE groups (
  user_name           char(32) binary default '' not null,
  user_group          char(32)
);
```

The queries used to retrieve the information from these tables will then look like:

```
auth_query  SELECT password
            FROM passwd
            WHERE user_name = '%C{User-Name}'
            AND service = '%C{Auth-Data}'

group_query SELECT user_group
            FROM groups
            WHERE user_name = '%C{User-Name}'
```

It is supposed, that the information about the particular service a user is wishing to obtain, will be kept in `Auth-Data` attribute in LHS of a user's profile.

### 4.11.3 Authorization Parameters

These parameters define queries used to retrieve the authorization information from the SQL database. All the queries refer to the authentication database.

`check_attr_query` *string*

This query must return a list of triplets:

*attr-name*, *attr-value*, *opcode*

The query is executed before comparing the request with the profile entry. The values returned by the query are added to LHS of the entry. *opcode* here means one of valid operation codes: '=', '!=', '<', '>', '<=', '>='.

`reply_attr_query` *string*

This query must return pairs:

*attr-name*, *attr-value*

The query is executed after a successful match, the values it returns are added to the RHS list of the matched entry, and are therefore returned to the NAS in the reply packet.

## Example of Authorization Parameters

Suppose your attribute information is stored in a SQL table of the following structure:

```
CREATE TABLE attrib (
  user_name varchar(32) default '' not null,
  attr      char(32) default '' not null,
  value     char(128),
  op enum("=", "!=", "<", ">", "<=", ">=") default null
);
```

Each row of the table contains the attribute-value pair for a given user. If `op` field is `NULL`, the row describes LHS (check) pair. Otherwise, it describes a RHS (reply) pair. The authorization queries for this table will look as follows:

```
check_attr_query  SELECT attr,value,op \
                  FROM attrib \
                  WHERE user_name='%u' \
                  AND op IS NOT NULL

reply_attr_query  SELECT attr,value \
                  FROM attrib \
                  WHERE user_name='%u' \
                  AND op IS NULL
```

Now, let's suppose the 'raddb/users' contains only one entry:

```
DEFAULT Auth-Type = SQL
        Service-Type = Framed-User
```

And the `attrib` table contains following rows:

| user_name | attr | value | op |
|---|---|---|---|
| jsmith | NAS-IP-Address | 10.10.10.1 | = |
| jsmith | NAS-Port-Id | 20 | <= |
| jsmith | Framed-Protocol | PPP | NULL |
| jsmith | Framed-IP-Address | 10.10.10.11 | NULL |

Then, when the user `jsmith` is trying to authenticate, the following happens:

1. Radius finds the matching entry (`DEFAULT`) in the 'raddb/users'.
2. It queries the database using the `check_attr_query`. The triplets it returns are then added to the LHS of the profile entry. Thus, the LHS will contain:
   ```
   Auth-Type = SQL,
   NAS-IP-Address = 10.10.10.1,
   NAS-Port-Id <= 20
   ```
3. Radius compares the incoming request with the LHS pairs thus obtained. If the comparison fails, it rejects the authentication. *Please note*, that the `Auth-Type` attributes itself triggers execution of `auth_query`, described in the previous section.
4. After a successful authentication, Radius queries the database, using `reply_attr_query`, and adds its return to the list of RHS pairs. The RHS pairs will then be:
   ```
   Service-Type = Framed-User,
   Framed-Protocol = PPP,
   Framed-IP-Address = 10.10.10.11
   ```
   This list is returned to the NAS along with the authentication accept packet.

Thus, this configuration allows the user `jsmith` to use only NAS 10.10.10.1, ports from 1 to 20 inclusive. If the user meets these conditions, he is allowed to use PPP service, and is assigned IP address `10.10.10.11`.

### 4.11.4 Accounting Parameters

To perform the SQL accounting `radiusd` needs to know the database where it is to store the accounting information. This information is supplied by the following statements:

`doacct` *bool*

When set to `yes` enables SQL accounting. All `acct_` keywords are ignored if `doacct` is set to `no`.

`acct_db` *string*
: Specifies the name of the database where the accounting information is to be stored.

`acct_max_connections` *number*
: Specifies the maximum number of accounting SQL connections to keep open. This parameter is ignored if `keepopen` is set to `no`.

Further, `radiusd` needs to know which information it is to store into the database and when. Each of five accounting request types (see Section 2.2.2 [Accounting Requests], page 20) has a SQL query associated with it. Thus, when radius receives an accounting request, it determines the query to use by the value of `Acct-Status-Type` attribute.

Following statemens define the accounting queries:

`acct_start_query` *string*
: Specifies the SQL query to be used when *Session Start Packet* is received. Typically, this would be some `INSERT` statement (see Section 4.11.4.1 [Queries], page 55).

`acct_stop_query` *string*
: Specifies the SQL query to be used when *Session Stop Packet* is received. Typically, this would be some `UPDATE` statement.

`acct_stop_query` *string*
: Specifies the SQL query to be executed upon arrival of a *Keepalive Packet*. Typically, this would be some `UPDATE` statement.

`acct_nasup_query` *string*
: Specifies the SQL query to be used upon arrival of an *Accounting Off Packet*.

`acct_nasdown_query` *string*
: Specifies the SQL query to be used when a NAS sends *Accounting On Packet*.

None of these queries should return any values.

#### 4.11.4.1 Writing SQL Accounting Query Templates

Let's suppose you have an accounting table of the following structure:

```
CREATE TABLE calls (
  status              int(3),
  user_name           char(32),
  event_date_time     datetime DEFAULT '0000-00-00 00:00:00' NOT NULL,
  nas_ip_address      char(17),
  nas_port_id         int(6),
  acct_session_id     char(16) DEFAULT '' NOT NULL,
  acct_session_time   int(11),
  acct_input_octets   int(11),
  acct_output_octets  int(11),
  connect_term_reason int(4),
  framed_ip_address   char(17),
  called_station_id   char(32),
  calling_station_id  char(32)
);
```

On receiving the *Session Start Packet* we would insert a record into this table with `status` set to 1. At this point the columns `acct_session_time`, `acct_input_octets`, `acct_output_octets` as well as `connect_term_reason` are unknown, so we will set them to 0:

```
# Query to be used on session start
acct_start_query     INSERT INTO calls \
                     VALUES(%C{Acct-Status-Type},\
                            '%u',\
                            '%G',\
                            '%C{NAS-IP-Address}',\
                            %C{NAS-Port-Id},\
                            '%C{Acct-Session-Id}',\
                            0,\
                            0,\
                            0,\
                            0,\
                            '%C{Framed-IP-Address}',\
                            '%C{Called-Station-Id}',\
                            '%C{Calling-Station-Id}')
```

Then, when the *Session Stop Packet* request arrives we will look up the record having `status` = 1, `user_name` matching the value of `User-Name` attribute, and `acct_session_id` matching that of `Acct-Session-Id` attribute. Once the record is found, we will update it, setting

```
status = 2
acct_session_time = value of Acct-Session-Time attribute
acct_input_octets = value of Acct-Input-Octets attribute
acct_output_octets = value of Acct-Output-Octets attribute
connect_term_reason = value of Acct-Terminate-Cause attribute
```

Thus, every record with `status` = 1 will represent the active session and every record with `status` = 2 will represent the finished and correctly closed record. The constructed `acct_stop_query` is then:

```
# Query to be used on session end
acct_stop_query      UPDATE calls \
                     SET status=%C{Acct-Status-Type},\
                         acct_session_time=%C{Acct-Session-Time},\
                         acct_input_octets=%C{Acct-Input-Octets},\
                         acct_output_octets=%C{Acct-Output-Octets},\
                         connect_term_reason=%C{Acct-Terminate-Cause} \
                     WHERE user_name='%C{User-Name}' \
                     AND status = 1 \
                     AND acct_session_id='%C{Acct-Session-Id}'
```

Upon receiving a *Keepalive Packet* we will update the information stored with `acct_start_query`:

```
acct_alive_query  UPDATE calls \
                  SET acct_session_time=%C{Acct-Session-Time},\
                      acct_input_octets=%C{Acct-Input-Octets},\
                      acct_output_octets=%C{Acct-Output-Octets},\
                      framed_ip_address=%C{Framed-IP-Address} \
                  WHERE user_name='%C{User-Name}' \
                  AND status = 1 \
                  AND acct_session_id='%C{Acct-Session-Id}'
```

Further, there may be times when it is necessary to bring some NAS down. To correctly close the currently active sessions on this NAS we will define a `acct_nasdown_query` so that it would set `status` column to 2 and update `acct_session_time` in all records having `status` = 1 and `nas_ip_address` equal to IP address of the NAS. Thus, all sessions on a given NAS will be closed correctly when it brought down. The `acct_session_time` can be computed as difference between the current time and the time stored in `event_date_time` column:

```
# Query to be used when a NAS goes down, i.e. when it sends
# Accounting-Off packet
acct_nasdown_query UPDATE calls \
                   SET status=2,\
                       acct_session_time=unix_timestamp(now())-\
                               unix_timestamp(event_date_time) \
                   WHERE status=1 \
                   AND nas_ip_address='%C{NAS-IP-Address}'
```

We have not covered only one case: when a NAS crashes, e.g. due to a power failure. In this case it does not have a time to send `Accounting-Off` request and all its records remain open. But when the power supply is restored, the NAS will send an *Accounting On packet*, so we define a `acct_nasup_query` to set `status` column to 3 and update `acct_session_time` in all open records belonging to this NAS. Thus we will know that each record having `status` = 3 represents a crashed session. The query constructed will be:

```
# Query to be used when a NAS goes up, i.e. when it sends
# Accounting-On packet
acct_nasup_query   UPDATE calls \
                   SET status=3,\
                       acct_session_time=unix_timestamp(now())-\
                               unix_timestamp(event_date_time) \
                   WHERE status=1 \
                   AND nas_ip_address='%C{NAS-IP-Address}'
```

## 4.12 Rewrite functions — ‘raddb/rewrite’

The file ‘`raddb/rewrite`’ contains definitions of Rewrite extension functions. For information regarding Rewrite extension language See Section 9.1 [Rewrite], page 75.

## 4.13 Login Menus — ‘raddb/menus’

The menus is a way to allow user the choice between various services he could be provided. The menu functionality is enabled when Radius is compiled with ‘`--enable-livingston-menus`’ option.

A user is presented a menu after it is authenticated if the RHS of his profile record consists of a single A/V pair in the form:

```
Menu = <menu-name>
```

The menu files are stored in directory 'raddb/menus'.

### 4.13.1 A menu file syntax.

A menu file is a text file containing a menu declaration and any number of choice descriptions. The menus can be nested to an arbitrary depth.

A comment is introduced by a '#' character. All characters from this one up to the end of line are discarded.

The menu declaration is contained between the words 'menu' and 'end'. Each of these must be the only word on a line and must start in column 1.

Choice descriptions follow the menu declaration. Each description starts with a line containing choice identifier. A choice identifier is an arbitrary word identifying this choice, or a word 'DEFAULT'. It is followed by comma-separated list of A/V pairs which will be returned to the server when a user selects this choice.

### 4.13.2 An example of menu files

## Single-Level Menu

Suppose the following file is stored under 'raddb/menus/menu1':

```
menu
        *** Welcome EEE user! ***
Please select an option:

        1. Start CSLIP session
        2. Start PPP session
        3. Quit

        Option:
end
# CSLIP choice
# Framed-IP-Address of 255.255.255.254 indicates that the NAS should
# select an address for the user from its own IP pool.
1
        Service-Type = Framed-User,
        Framed-Protocol = SLIP,
        Framed-IP-Address = 255.255.255.254,
        Termination-Menu = "menu1"
# PPP choice
2
        Service-Type = Framed-User,
        Framed-Protocol = PPP,
        Framed-IP-Address = 255.255.255.254,
```

```
        Termination-Menu = "menu1"
# A special menu EXIT means abort the session
3
        Menu = "EXIT"
# Return to this menu if no valid choice have been entered
DEFAULT
        Menu = "menu1"
```

Now, suppose the 'raddb/users' contains the following profile entry:

```
DEFAULT Auth-Type = System
        Menu = "menu1"
```

and user 'jsmith' has a valid system account and tries to log in from some NAS. Upon authenticating the user, the Radius server sees that his reply pairs contain the Menu attribute. Radius then sends Access-Challenge packet to the NAS with the text of the menu in it. The 'jsmith' then sees on his terminal:

```
        *** Welcome EEE user! ***
Please select an option:

        1. Start CSLIP session
        2. Start PPP session
        3. Quit

        Option:
```

He then enters '2'. The NAS sends the Access-Request packet to the server, which sees that user wishes to use option 2 and replies to the NAS with an Access-Accept packet containing the following attributes:

```
        Service-Type = Framed-User,
        Framed-Protocol = PPP,
        Framed-IP-Address = 255.255.255.254,
        Termination-Menu = "menu1"
```

The Termination-Menu in this list makes sure the same process will continue when 'jsmith' logs out, i.e. he will be presented the same menu again until he enters choice '3' which will disconnect him.

## Nested menus

In this example, the 'other' choice refers to the menu above.

```
menu
        *** Welcome here! ***
Please enter an option:
        ppp     ---     Start PPP session
        telnet  ---     Begin guest login session
        other   ---     Select other option

        Enter your choice:
end
ppp
        Service-Type = Framed-User,
```

```
        Framed-Protocol = PPP
telnet
        Service-Type = Login-User,
        Login-IP-Host = 10.11.11.7,
        Login-Service = Telnet,
        Login-TCP-Port = 23
other
        Menu = "menu1"
DEFAULT
        menu = "menu2"
```

## 4.14 Macro Substitution

Some statements in the configuration files need to use the actual values of the attributes supplied with the request. These are:

- `Exec-Program` and `Exec-Program-Wait` assignments in '`users`' database
- SQL query templates in '`sqlserver`'

In these statements the following macros are replaced by the value of corresponding attributes:

`%Cnum` (num is a decimal number). This variable is replaced by the value of attribute number 'num'. The attribute is looked up in the incoming request pairlist.

`%C{attr-name}`
This is replaced by the value of attribute named 'attr-name'. The attribute is looked up in the incoming request pairlist.

`%Rnum` (num is a decimal number). This variable is replaced by the value of attribute number 'num'. The attribute is looked up in the reply pairlist.

`%R{attr-name}`
This is replaced by the value of attribute named 'attr-name'. The attribute is looked up in the reply pairlist.

`%D` This is replaced by current date/time (localtime).

`%G` This is replaced by current date/time in GMT.

The "'`{}`' form" allows to specify default value for the substitution. The default value will be used when no such attribute is encountered in the pairlist. The syntax for specifying the default value resembles that of shell environment variables.

The substitution `%C{`*attr-name*`:-`*defval*`}` is expanded to the value of *attr-name* attribute, if it is present in the request pairlist, and to *defval* otherwise. For example:

```
%C{Acct-Session-Time:-0}
```

will return the value of Acct-Session-Time attribute or 0 if it doesn't exist in the request pairlist.

The substitition `%C{`*attr-name*`:=`*defval*`}` is expanded to the value of *attr-name* attribute. If this attribute is not present in the request pairlist, it will be created and assigned the value *defval*. E.g.:

```
%C{Acct-Session-Time:=0}
```

The substitution `%C{`*attr-name*`:?`*message*`}` is expanded to the value of *attr-name* attribute, if it is present. Otherwise the diagnostic message "*attr-name*: *message*" is issued to the log error channel, and string "*message*" is returned.

The substitition `%C{`*attr-name*`:+`*retval*`}` is expanded to empty string if the attribute *attr-name* is present in the referenced pairlist. Othervise it is expanded to *retval*.

You can also use the following shortcuts:

`%p` Port number

`%n` NAS IP address

`%f` Framed IP address

`%u` User name

`%c` Callback-Number

`%i` Calling-Station-Id

`%t` MTU

`%a` Protocol (SLIP/PPP)

`%s` Speed (Connect-Info attribute)

# 5 Authentication

An *Authentication Type* specifies which credentials the user is required to supply in order to be authenticated and where the user's authentication data are stored. It is defined by the value of `Auth-Type` attribute in LHS of a '`users`' entry.

## 5.1 Accept Authentication Type

*Accept* is the simplest authentication type. Users with this authentication type will be authenticated successfully without checking any credentials. Actually this means that only username is required for authentication.

This authentication type is used for each '`users`' entry, whose LHS contains

```
Auth-Type = Accept
```

This authentication type can be used for guest accounts, e.g. the following profile in '`users`':

```
guest   Auth-Type = Accept,
                Password != "",
                Simultaneous-Use = 10
        Service-Type = Framed-User,
                Framed-Protocol = PPP
```

allows up to 10 simultaneous guest PPP accounts. To log in using such guest account it is sufficient to use username '`guest`' and any non-empty password.

## 5.2 Reject Authentication Type

The *Reject* authentication type causes the request to be rejected unconditionally. It can be used to disable a user account (For another method of disabling user accounts, see Section 4.10 [access.deny file], page 51).

This authentication type is used for each '`users`' entry, whose LHS contains

```
Auth-Type = Reject
```

## 5.3 Local Password Authentication Type

The *Local Password* authentication type allows to keep plaintext user passwords. Although the use of this authentication type is strongly discouraged for security reasons, this is the only authentication type that can be used with CHAP authentication.

There are two ways of using this authentication type

### Specifying Passwords in users File.

To keep the plaintext passwords in '`users`' file, the profile entry must follow this pattern:

```
user-name  Auth-Type = Local,
                     Password = plaintext
```

The *plaintext* is the user's plaintext password. Obviously, *user-name* may not be `DEFAULT` nor `BEGIN`.

### Specifying Passwords in SQL Database.

```
user-name   Auth-Type = Local,
                      Password-Location = SQL
```

When the user is authenticated using such profile, its password is retrieved from the authentication database using `auth_query`. The configuration of SQL authentication is described in detail in Section 4.11.2 [Authentication Server Parameters], page 52.

## 5.4 Encrypted Password Authentication Type

The *Encrypted Password* type allows to keep user's passwords encrypted via DES or MD5 algorythm. There are two ways of using this authentication type.

### Specifying Passwords in users File.

```
user-name  Auth-Type = Crypt-Local,
                     Password = crypt-pass
```

The `Crypt-Password` is a shortcut for the above notation:

```
user-name  Crypt-Password = crypt-pass
```

### Specifying Passwords in SQL Database.

```
user-name   Auth-Type = Crypt-Local,
                      Password-Location = SQL
```

Using this profile, the user's password is retrieved from the authentication database using `auth_query`. The configuration of SQL authentication is described in detail on Section 4.11.2 [Authentication Server Parameters], page 52.

The shortcut for this notation is `Auth-Type = SQL`.

In any case, the passwords used with this authentication type must be either DES or MD5 hashed.

## 5.5 System Authentication Type

The *System* authentication type requires that the user have a valid system account on the machine where the radius server is running. The use of this type is triggered by setting

```
Auth-Type = System
```

in the LHS of a '`users`' entry.

## 5.6 SQL Authentication Type

Setting `Auth-Type = SQL` or `Auth-Type = Mysql` in the LHS of a '`users`' entry is a synonim for

```
Auth-Type = Crypt-Local, Password-Location = SQL
```

and is provided as a shortcut and for backward compatibility with previous versions of GNU Radius.

For description of SQL authentication, See Section 5.4 [Encrypted Password Auth], page 64. The configuration of SQL subsystem is described in See Section 4.11 [sqlserver file], page 51.

## 5.7 PAM Authentication Type

*PAM* authentication type indicates that a user should be authenticated using PAM (Pluggable Authentication Module) framework. The simplest way of usage is:

```
Auth-Type = PAM
```

Any user whose '`users`' profile contains the above, will be authenticated via PAM, using service name '`radius`'. If you wish to use another service name, set it using `Auth-Data` attribute, e.g.:

```
Auth-Type = PAM,
    Auth-Data = pam-service
```

## 5.8 Defining Custom Authentication Types

The are three ways to define custom authentication types:

1. Write a PAM module.
2. Use a Guile procedure.
3. Use an external program

You can write a PAM module implementing the new authentication type. Then, specifying `Auth-Type = PAM` allows to apply it (see Section 5.7 [PAM Auth], page 65).

Alternatively, you may write a Scheme procedure implementing the new authentication type. To apply it, use `Scheme-Procedure` attribute in RHS. The `Auth-Type = Accept` can be used in LHS if the whole authentication burden is to be passed to the Scheme procedure. For example, if one wrote a procedure `my-auth`, to apply it to all users, one will place the following profile in his '`users`' file:

```
DEFAULT  Auth-Type = Accept
         Scheme-Procedure = "my-auth"
```

For a discussion of how to write Scheme authentication procedures, See Section 9.2.2 [Authentication with Scheme], page 86.

The third way to implemement your own authentication method is using an external program. This is less effective than the methods described above, but may be necessary sometimes. To invoke the program, use the following statement in the RHS of '`users`' entry:

```
Exec-Program-Wait = "progname args"
```

The *progname* must be the full path to the program, *args* — any arguments it needs. The usual substitutions may be used in *args* to pass any request attributes to the program (see Section 4.14 [Macro Substitution], page 60).

For a detailed description of `Exec-Program-Wait` attribute and an example of its use, See Section 12.3.15 [Exec-Program-Wait], page 129.

## 5.9 Checking Simultaneous Logins

The number of sessions a user can have open simultaneously can be restricted by setting `Simultaneous-Use` attribute in the user's profile LHS (see Section 12.3.10 [Simultaneous-Use], page 126). By default the number of simultaneous sessions is unlimited.

When a user with limited number of simultaneous logins authenticates himself, Radius first counts the number of the sessions that are already opened by this user. If this number is equal to the value of `Simultaneous-Use` attribute the authentication request is rejected.

To determine the number of open sessions, Radius scans the '`radlog/radutmp`' for any open entries marked with the user's login name Section 6.1 [UNIX Accounting], page 67. Such entries are created when Radius receives an Accounting-Request packet with `Acct-Status-Type` attribute set to `Start`. An entry is marked closed when a corresponding Accounting-Request packet arrives with `Acct-Status-Type` attribute set to `Stop`.

Since an open entry might be a result of missing `Stop` packet, Radius queries the NAS whether the session listed in the entry is currently active. If the NAS replies positive, the session count is incremented, if it replies negative, such entry is marked as closed and is not counted. There may also be cases when the NAS is unreachable due to some reasons. In such cases the Radius behavior is determined by the value of `checkrad-assume-logged` in '`config`' file Section 4.1.3 [auth statement (raddb/config)], page 35. If the value is `yes`, Radius assumes the session is still active and increases the session count, otherwise it proceeds as if the NAS returned negative reply.

To query a NAS, Radius first looks up its type and additional parameters in '`naslist`' file (see Section 4.4 [naslist file], page 44). If the NAS type is '`true`', Radius acts as if the NAS returned 1, if the type is '`false`', it acts as if the NAS returned 0, otherwise it looks up the entry in the '`nastypes`' which has matching type (see Section 4.5 [nastypes file], page 45). If such entry does not exist, Radius issues the error message and acts accordingly to the value of configuration variable `checkrad-assume-logged`. Otherwise, Radius determines the query method to use from the second field of this entry, and constructs its arguments by appending arguments from the '`naslist`' entry to those of `nastypes` entry. *Please note*, that the former take precedence over the latter, and can thus be used to override default values specified in '`nastypes`'.

Having determined the query method and its argument, Radius queries NAS and analyzes its output by invoking a user-supplied Rewrite function. The function to use is specified by the `function=` argument to the method. It is called each time a line of output is received from the NAS (for finger queries) or a variable is received (for SNMP queries). The process continues until the function returns 1 or the last line of output is read or a timeout occurs whichever comes first.

If the user-function returns 1 it is taken to mean the user's session is now active at the NAS, otherwise, if it replies 0 or if the end of output is reached, it is taken to mean the user's session is not active.

The syntax conventions for user-supplied functions are described in detail in Section 9.1.5 [Login Verification Functions], page 80.

# 6 Accounting

By default GNU Radius supports three types of accounting. Any additional accounting methods can be defined using extension mechanisms.

The accounting methods are applied to a request in a following sequence:

1. UNIX Accounting
2. Detailed Request Accounting
3. SQL Accounting
4. Custom Accounting

In this sequence, only *UNIX Accounting* is obligatory, all other methods are applied only when enabled.

If any accounting type in this sequence fails, the accounting is deemed to fail and all subsequent methods are not invoked.

## 6.1 UNIX Accounting

This accounting method is always enabled.

Radius keeps files '`radutmp`' and '`radwtmp`' in its logging directory and stores the accounting data there. The utilities `radwho` and `radlast` can be used to list information about users' sessions.

## 6.2 Detailed Request Accounting

Radius stores the detailed information about accounting packets it receives in files '`radacct/`*nasname*`/detail`' (see Chapter 1 [Naming Conventions], page 15), where *nasname* is replaceed with the short name of the NAS from the '`raddb/naslist`' file (see Section 4.4 [naslist file], page 44).

By default, this accounting type is always enabled, provided that '`radacct`' directory exists and is writable (see Chapter 1 [Naming Conventions], page 15). To turn the detailed accounting off, use `detail` statement in '`config`' file. For more information about it, See Section 4.1.4 [acct], page 36.

The accounting detail files consist of a record per each accounting request. A record includes the timestamp and detailed dump of attributes from the packet, e.g.:

```
Fri Dec 15 18:00:24 2000
        Acct-Session-Id = "2193976896017"
        User-Name = "e2"
        Acct-Status-Type = Start
        Acct-Authentic = RADIUS
        Service-Type = Framed-User
        Framed-Protocol = PPP
        Framed-IP-Address = 11.10.10.125
        Calling-Station-Id = "+15678023561"
        NAS-IP-Address = 11.10.10.11
        NAS-Port-Id = 8
```

```
          Acct-Delay-Time = 0
          Timestamp = 976896024
          Request-Authenticator = Unverified

Fri Dec 15 18:32:09 2000
          Acct-Session-Id = "2193976896017"
          User-Name = "e2"
          Acct-Status-Type = Stop
          Acct-Authentic = RADIUS
          Acct-Output-Octets = 5382
          Acct-Input-Octets = 7761
          Service-Type = Framed-User
          Framed-Protocol = PPP
          Framed-IP-Address = 11.10.10.125
          Acct-Session-Time = 1905
          NAS-IP-Address = 11.10.10.11
          NAS-Port-Id = 8
          Acct-Delay-Time = 0
          Timestamp = 976897929
          Request-Authenticator = Unverified
```

## 6.3 SQL Accounting

The SQL accounting method is enabled when Radius is configured with '`--enable-sql`' option and '`sqlserver`' file in its configuration directory is properly set up (see Section 4.11 [sqlserver file], page 51).

This version of GNU Radius (0.96) supports MySQL and PostgreSQL servers. Support for Oracle servers will be added in the nearest future.

With this accounting method enabled, `radiusd` will store the information about accounting requests in the configured SQL database. The accounting method is fully configurable: the Radius administrator defines both the types of requests to be accounted and the information to be stored into the database. (see Section 4.11 [sqlserver file], page 51).

## 6.4 Defining Custom Accounting Types.

If the built-in accounting methods do not meet your requirements, you can implement your own accounting methods. There are two ways for doing so:

1. Using a Guile procedure.
2. Using an external program

To use a Guile procedure for accounting, the name of the procedure must be specified as a value to `Scheme-Acct-Procedure` attribute in RHS list of a '`hints`' entry, e.g.:

```
DEFAULT NULL Scheme-Acct-Procedure = "my-acct"
```

For a detailed description of Scheme accounting procedures, See Section 9.2.3 [Accounting with Scheme], page 88.

Another way of implemementing your own accounting method is using an external program. This is less effective than the methods described above, but may be necessary sometimes. To invoke the program, use the following statement in the LHS of '`hints`' entry:

`Acct-Ext-Program = "`*progname args*`"`

The *progname* must be the full path to the program, *args* — any arguments it needs. The usual substitutions may be used in *args* to pass any request attributes to the program (see Section 4.14 [Macro Substitution], page 60).

For a detailed description of `Acct-Ext-Program`, See Section 12.3.16 [Acct-Ext-Program], page 130.

# 7 Logging

GNU Radius reports every event worth mentioning. The events are segregated by their severity level. Radius discerns the following levels (in order of increasing severity):

`Debug`
: The debug messages Chapter 8 [Debugging], page 73.

`Auth`
: Under this level every authentication attempt is logged. This is enabled by setting

```
level auth;
```

in `category auth` statement of '`config`' file.

`Proxy`
: Messages regarding proxy requests (see Section 2.4.1 [Proxying], page 22).

`Info`
: Informational messages.

`Notice`
: Normal, but significant conditions.

`Warning`
: Warning conditions. These mean some deviations from normal work.

`Error`
: Error conditions. Usually these require special attention.

`CRIT`
: Critical conditions due to which Radius is no longer able to continue working. These require urgent actions from the site administrator.

By default, all messages in all levels are output to file '`radlog/radius.log`'. In addition, messages in level `CRIT` are also duplicated to the system console. These defaults can be overridden using `logging` statement in '`raddb/config`' file (See Section 4.1.2 [logging statement], page 32, for the description of logging statement syntax, see Chapter 1 [Naming Conventions], page 15 for information about the locations of different radius configuration files).

# 8 Debugging

GNU Radius provides extensive debugging features. These are enabled either by '-x' command option to the radiusd Chapter 3 [Invocation], page 27, or by level statement in debug category Section 4.1.2 [logging statement], page 32. Both cases require as an argument a valid debug specification.

A debug specification sets the module for which the debugging should be enabled and debugging level. The higher the level is, the more detailed information is provided. The module name and level are separated by an equal sign. If the level is omitted, the highest possible level (100) is assumed. The module name may be abbreviated to first N characters, in which case the first matching module is selected. Several such specifications can be specified, in which case they should be separated by a comma. For example, the following is a valid debug specification:

```
proxy.c=10,files.c,config.y=1
```

It sets debug level 10 for module `proxy.c`, level 100 for `files.c`, and 1 for `config.y`.

The modules and debugging levels are subject to change from release to release.

# 9 Extensions

The use of extension language allows to extend the functionality of GNU Radius without having to modify its source code. The two extension languages supported are Rewrite and Scheme. Use of Rewrite is always enabled. Use of Scheme requires Guile version 1.4 or higher.

## 9.1 Rewrite

Rewrite is the GNU Radius extension language. Its name reflects the fact that it was originally designed to rewrite the broken request packets, so they could be processed as usual (see Section 9.1.4 [Rewriting Incoming Requests], page 76). Beside this basic use, however, Rewrite functions are used in verifying the activity of user sessions (see Section 5.9 [Checking Simultaneous Logins], page 66).

### 9.1.1 Syntax Overview

Rewrite syntax resembles that of C. Rewrite has two basic data types: integer and string. It does not have global variables, all variables are automatic. The only exception are the A/V pairs from the incoming request, which are accessible to Rewrite functions via special notation `%[`*attr*`]`.

### 9.1.2 Quick Start

As an example, let's consider the following Rewrite function:

```
string
foo(integer i)
{
    string rc;
    if (i % 2)
        rc = "odd";
    else
        rc = "even";
    return "the number is " + rc;
}
```

The function takes an integer argument and returns string "the number is odd" or "the number is even", depending on the value of *i*. This illustrates the fact that in Rewrite the addition operator is defined on the string type. The result of such operation is the concatenation of operands.

Another example is a function that adds a prefix to the `User-Name` attribute:

```
integer
px_add()
{
        %[User-Name] = "pfx-" + %[User-Name];
        return 0;
}
```

The function manipulates the contents of the incoming request, its return value has no special meaning.

### 9.1.3 Interaction with Radius

A Rewrite function can be invoked in several ways, depending on its purpose. There are three major kinds of Rewrite functions:

- Functions used to rewrite the incoming requests.
- Functions designed for simultaneous login verification.
- Functions used to generate RADIUS attribute values.

### 9.1.4 Rewriting Incoming Requests

The need of rewriting the incoming requests arises from the fact that some NASes are very particular about the information they send with the requests. There are cases when the information they send is hardly usable or even just unusable. For example, a Cisco AS5300 terminal server used as a voice over IP router packs a lot of information into its `Acct-Session-Id` attribute. Though the information stored there is otherwise relevant, it makes proper accounting impossible since the `Acct-Session-Id` attributes in the start and stop packets of the same session become different, and thus Radius cannot determine the *Session Start* to which the given *Session Stop* request corresponds (see Section 12.2.5 [Acct-Session-Id], page 120).

In order to cope with such NASes, GNU Radius is able to invoke a Rewrite function upon arrival of the packet and before further processing it. This function can transform the packet so, that it obtains the form prescribed by RFCs and its further processing becomes possible.

For example, in the case of AS5300 router, a corresponding rewrite function parses the `Acct-Session-Id` attribute, breaks it down into fields, stores them into proper attributes, creating them if necessary, and, finally replaces `Acct-Session-Id` with its real value, which is the same for start and stop records corresponding to a single session. Thus all the information that came with the packet is preserved, but the packet itself is made usable for proper accounting.

A special attribute, `Rewrite-Function`, is used to trigger invocation of a Rewrite function. Its value is a name of the function to be invoked.

When used in a '`naslist`' profile, the attribute causes the function to be invoked when the incoming request matches the huntgroup (see Section 2.4.3 [Huntgroups], page 24). For example, to have a function `fixup` invoked for each packet from the NAS `10.10.10.11`, the following huntgroup rule may be used:

```
DEFAULT  NAS-IP-Address = 11.10.10.11
         Rewrite-Function = "fixup"
```

The `Rewrite-Function` attribute may also be used in a '`hints`' rule. In this case, it will invoke the function if the request matches the rule (see Section 2.4.2 [Hints], page 23). For example, this '`hints`' rule will cause the function to be invoked for each request containing the username starting with '`P`':

```
DEFAULT  Prefix = "P"
         Rewrite-Function = "fixup"
```

*Please note*, that in both cases the attribute can be used either in LHS or in RHS pairs of a rule.

The packet rewrite function must be declared as having no arguments, and returning integer value:

```
integer fixup()
{
}
```

The actual return value from such a function is ignored, the integer return type is just a matter of convention.

The following subsection present some examples of packet rewriting functions.

#### 9.1.4.1 Examples of Various Rewrite Functions.

The examples found in this chapter are working functions that can be used with various existing NAS types. They are taken from the '`rewrite`' file contained in distribution of GNU Radius.

## 1. Port rewriting for MAX Ascend terminal servers

Some MAX Ascend terminal servers pack additional information into `NAS-Port-Id` attribute. The port number is constructed as XYYZZ, where X = 1 for digital, X = 2 for analog, YY is line number (1 for first PRI/T1/E1, 2 for second, so on), and ZZ = channel number (on the PRI or Channelized T1/E1).

The following rewrite functions are intended to compute the integer port number in the range (1 .. *portcnt*), where *portcnt* represents the real number of physical ports available on the NAS. Such port number can be used, for example, with `Add-Port-To-IP-Address` attribute (see Section 12.3.13 [Add-Port-To-IP-Address], page 128).

```
/*
 * decode MAX port number
 * input: P        --  The value of NAS-Port-Id attribute
 *        portcnt  --  number of physical ports on the NAS
 */
integer
max_decode_port(integer P, integer portcnt)
{
    if (P > 9999) {
        integer s, l, c;

        s = P / 10000;
        l = (P - (10000 * s))/100;
        c = P - ((10000 * s) + (100 * l));
        return (c-1) + (l-1) * portcnt;
    }
    return P;
}
```

```
/*
 * Interface function for MAX terminal server with 23 ports.
 * Note that it saves the received NAS-Port-Id attribute in the
 * Orig-NAS-Port-Id attribute. The latter must be defined somewhere
 * in the dictionary
 */
integer
max_fixup()
{
    %[Orig-NAS-Port-Id] = %[NAS-Port-Id]; # Preserve original data
    %[NAS-Port-Id] = max_decode_port(%[NAS-Port-Id], 23);
    return 0;
}
```

## 2. Session ID parsing for Cisco AS 5300 series

Cisco VOIP IOS encodes a lot of other information into its `Acct-Session-Id`. The pieces of information are separated by '`/`' character. The part of `Acct-Session-Id` up to first '`/`' character is the actual session ID.

On the other hand, its accounting packets lack `NAS-Port-Id`, though they may contain the vendor-specific pair with code 2 (vendor PEC 9), which is the string in the form '`ISDN 9:D:999`' ('`9`' represents a decimal digit). The number after the last '`:`' character can be used as a port number.

The following code parses `Acct-Session-Id` attribute and stores the information it contains in various other attributes, generates normal `Acct-Session-Id` and attempts to generate `NAS-Port-Id` attribute.

```
/*
 * The port rewriting function for Cisco AS5300 used for VoIP.
 * This function is used to generate NAS-Port-Id pair on the basis
 * of vendor-specific pair 2. If the latter is in the form
 * "ISDN 9:D:999" (where each 9 represents a decimal digit), then
 * the function returns the number after the last colon. This is
 * used as a port number.
 */
integer
cisco_pid(string A)
{
    if (A =~
        ".*\([0-9][0-9]*\):[A-Z0-9][A-Z0-9]*:\([0-9][0-9]*\)") {
        return (integer)\2;
    }
    return -1;
}
```

```
/*
 * This function parses the packed session id.
 * The actual sid is the number before the first slash character.
 * Other possibly relevant fields are also parsed out and saved
 * in the Voip-* A/V pairs. The latter should be defined somewhere
 * in the dictionary.
 * Please note, that the regular expression in this example
 * spans several lines for readability. It should be on one
 * line in real file.
 */
string
cisco_sid(string S)
{
   if (S =~ "\(.[^/]*\)/[^/]*/[^/]*/\([^/]*\)/\([^/]*\)/
             \([^/]*\)/\([^/]*\)/\([^/]*\)/\([^/]*\)
             /\([^/]*\).*") {
        %[Voip-Connection-ID] = \2;
        %[Voip-Call-Leg-Type] = \3;
        %[Voip-Connection-Type] = \4;
        %[Voip-Connect-Time] = \5;
        %[Voip-Disconnect-Time] = \6;
        %[Voip-Disconnect-Cause] = \7;
        %[Voip-Remote-IP] = \8;
        return \1;
   }
   return S;
}
/*
 * Normalize cisco AS5300 packets
 */
integer
cisco_fixup()
{
    integer pid;

    if ((pid = cisco_pid(%[Cisco-PRI-Circuit])) != -1) {
        if (*%[NAS-Port-Id])
            %[Orig-NAS-Port-Id] = %[NAS-Port-Id];
        %[NAS-Port-Id] = pid;
    }
    if (*%[Acct-Session-Id]) {
        %[Orig-Acct-Session-Id] = %[Acct-Session-Id];
        %[Acct-Session-Id] = cisco_sid(%[Acct-Session-Id]);
    }
    return 0;
}
```

## 3. Username rewriting for NT machines.

Users coming from Windows NT machines often authenticate themselves as 'NT_DOMAIN\username'. The following function selects the username part and stores it in the User-Name attribute:

```
integer
login_nt(string uname)
{
    integer i;

    if ((i = index(uname, '\\')) != -1)
        return substr(uname, i+1, -1);
    return uname;
}

integer
nt_rewrite()
{
    %[Orig-User-Name] = %[User-Name];
    %[User-Name] = login_nt(%[User-Name]);
    return 0;
}
```

### 9.1.5 Login Verification Functions

A login verification function is invoked to process the output from the NAS. This process is described in Section 5.9 [Checking Simultaneous Logins], page 66. The function to be invoked for given NAS is defined by `function` flag in '`raddb/nastypes`' or '`raddb/naslist`' files (see Section 4.5 [nastypes file], page 45). It must be defined as follows:

```
integer check(string str, string name, integer pid, string sid)
{
}
```

Its arguments are:

*str* Input string. If the query method is `finger`, this is the string of output received from the NAS with trailing newline stripped off. If the query method is `snmp`, this is the received variable value converted to its string representation.

*name* User name.

*pid* Port Id of the session.

*sid* Session ID.

The function should return non-0 if its arguments match user's session and 0 otherwise.

#### 9.1.5.1 Examples of login verification functions

As an example, let's consider the function for analyzing a line line of output from a standard UNIX finger service. In each line of finger output the first field contains username, the third field — tty number (Port ID), and the seventh field contains session ID. The function must return 1 if the three fields match the input user name, port and session IDs.

```
integer
check_unix(string str, string name, integer pid, string sid)
{
    return field(str, 1) == name
           && field(str, 3) == pid
           && field(str, 7) == sid;
}
```

Next example is a function to analyze a line of output from an SNMP query returning a user name. This function must return 1 if entire input line matches the user name.

```
integer
check_username(string str, string name, integer pid, string sid)
{
    return str == name;
}
```

### 9.1.6 Attribute Creation Functions

These are the functions, used to create RADIUS reply attributes. An attribute creation function can take any number of arguments. The type of its return is determined by the type of RADIUS attribute the value will be assigned to. To invoke the function, write its name in the A/V pair of RHS in '`raddb/users`' file, e.g.:

```
DEFAULT Auth-Type = SQL
        Service-Type = Framed-User,
                Framed-IP-Address = "=get_ip_addr(10.10.10.1)"
```

The function `get_ip_addr` will be invoked after successful authentication and it will be passed IP address `10.10.10.1` as its argument. An example of a useful function, that can be invoked this way:

```
integer
get_ip_address(integer base)
{
    return base + %[NAS-Port-Id] - %[NAS-Port-Id]/16;
}
```

### 9.1.7 Full Syntax Description

#### 9.1.7.1 Rewrite Data Types

There are only two data types: `integer` and `string`, the two being coercible to each other in the sense that a string can be coerced to an integer if it contains a valid ASCII representation of a decimal, octal or hex number, and the integer can always be coerced to a string, the result of such coercion being the ASCII string with decimal representation of the number.

#### 9.1.7.2 Rewrite Symbols

A symbol is a lexical token. The following symbols are recognized:

Arithmetical operators
: These are '+', '-', '*', '/' representing the basic arithmetical operations and '%' meaning remainder.

Comparison operators
: These are: '==', '!=', '<', '<=', '>', '>=' with the same meaning they have in C. Special operators are provided for regular expression matching. Binary operator '=~' returns true, if its left-hand-side operand matches the regular expression on its right-hand side. '!~' returns true if the left-hand side operand *does not match* the regexp on right-hand side. The right-hand side operand of '!~' or '=~' must be a literal string, i.e. the regular expression must be known at compile time.

Unary operators.
: Unary operators are '-' and '+' for unary plus and minus, '!' for boolean negation and '*' for testing for the existence of an attribute.

Boolean operators.
: These are: '&&' and '||'.

Parentheses '(' and ')'
: These are used to change the precedence of operators, to introduce type casts (type coercions), to declare functions and to pass actual arguments to functions.

Curly braces ('{' and '}')
: These are used to delimit blocks of code.

Numbers
: Numbers follow usual C convention for integers. A number consisting of a sequence of digits, is taken to be octal if it begins with '0' (digit zero) and decimal otherwise. If the sequence of digits is preceded by '0x' or '0X', it is taken to be a hexadecimal integer.

Characters
: These follow usual C convention for characters, i.e. either an ASCII character or its value enclosed in a pair of single quotes. The character value begins with '\' (backslash) and consists either of three octal or of two hexadecimal digits. A character does not form a special data type, it is represented internally by an integer.

Quoted strings
: These follow usual C conventions for strings.

Attribute values
: The incoming request is passed implicitly to functions, invoked via `Rewrite-Function` attribute. It is kept as an associative array, whose entries can be accessed using the following syntax:

    '%[' *attribute-name* ']'

  Thus notation returns the value of the attribute *attribute-name*. *attribute-name* should be a valid Radius dictionary name (see Section 4.2 [dictionary file], page 41).

Identifiers
: Identifiers represent functions and variables. These are described in the next section.

Regexp group references

A sequence of characters in the form:

'*number*'

refers to the contents of parenthesized group number *number* obtained as a result of the last executed '=~' command. The regexp group reference has always string data type. E.g.

```
string
basename(string arg)
{
    if (arg =~ ".*/\(.*\)\..*")
        return \1;
    else
        return arg;
}
```

This function strips from *arg* all leading components up to the last slash character, and all trailing components after the last dot character. It returns *arg* unaltered, if it does not contain slashes and dots. Roughly, it is analogous to the system `basename` utility.

### 9.1.7.3 Rewrite Identifiers

A valid identifier is a string of characters meeting the following requirements:

1. It starts with either a lower- or uppercase letter of the Latin alphabet or any of the following symbols: '_', '$'.
2. It consists of alphanumeric characters, underscores('_') and dollar signs ('$').

### 9.1.7.4 Rewrite Declarations

## Function declarations

The Rewrite function is declared as follows:

*type function-name* (*parameter-list*)

where *type* specifies the return type of the function, *function-name* declares the symbolic name of the function and *parameter-list* declares the formal parameters to the function. It is a comma-separated list of declarations in the form:

*type parm-name*

*type* being the parameter type, and *parm-name* being its symbolic name. Both *function-name* and *parm-name* should be valid identifiers.

## Variable declarations

There are no global variables in Rewrite. All variables are local. The local variables are declared right after the opening curly brace ('{') and before any executable statements. The declaration syntax is:

*type ident_list* ;

Here *ident_list* is either a valid Rewrite identifier, or a comma- separated list of such identifiers. *Please note* that, unlike in C, no assignments are allowed in variable declarations.

#### 9.1.7.5 Rewrite Statements

The Rewrite statements are: expressions, assignments, conditional statements and return statements. A statement is terminated by semicolon.

### Expressions

An *expression* is:

- A variable identifier
- A type coercion expression
- An arithmetic expression
- A boolean expression
- An assignment
- A function call

### Type coercion

The type coercion is like a type cast in C. Its syntax is

'`(`' *type* '`)`' *ident*

the result of type coercion is as follows:

| *type* | Variable type | Resulting conversion |
|---|---|---|
| integer | integer | No conversion. This results in the same integer value. |
| integer | string | If the string value of the variable is a valid ASCII representation of the integer number (either decimal, octal or hex) it is converted to the integer, otherwise the result of the conversion is undefined. |
| string | integer | The ASCII representation (in decimal) of the integer number. |
| string | string | No conversion. This results in the same string value. |

### Assignment

An assignment is:

*ident* = *expression* ;

The variable *ident* is assigned the value of *expression*.

### Function calls

These take the form:

*ident* ( *arg-list* )

where *ident* is the identifier representing the function, *arg-list* is a comma-separated list of expressions supplying actual arguments to the function. The function *ident* references can be either a compiled function or a built-in function.

*Please note* that, unlike in C, the mismatch between the number of actual arguments and number of formal parameters in the compiled function declaration is not an error but rather a warning.

### 9.1.7.6 Rewrite Built-in Functions

The following built-in functions are provided:

`integer` **length** (`string` *s*) — Function

Returns the length of string *s*.

`integer` **index** (`string` *s*, `integer` *c*) — Function

Returns the index of the first occurrence of the character *c* in the string *s*. Returns -1 if no such occurrence is found.

`integer` **rindex** (`string` *s*, `integer` *i*) — Function

Returns the index of the last occurrence of the character *c* in the string *s*. Returns -1 if no such occurrence is found.

`string` **substr** (`string` *s*, `integer` *start*, `integer` *length*) — Function

Returns the at most *length* substring of *s* starting at position *start*.

`integer` **logit** (`string` *msg*) — Function

Outputs its argument to the radius log channel `info`. Returns 0. The function is intended for debugging purposes.

All character positions in strings are counted from 0.

## 9.2 Guile

The name Guile stands for *GNU's Ubiquitous Intelligent Language for Extensions*. It provides the Scheme interpreter conforming to R4RS language specification. This section describes use of Guile as an extension language for GNU Radius. It assumes that the reader is sufficiently familiar with the Scheme language. Please, refer to section "Top" in *Revised(4) Report on the Algorithmic Language Scheme*, for the information about the language. If you wish to know more about the Guile, See section "Overview" in *The Guile Reference Manual*.

Scheme procedures can be called for processing both authentication and accounting requests. The invocation of a scheme procedure for an authentication request is triggered by `Scheme-Procedure` attribute, the invocation for an accounting request is triggered by `Scheme-Acct-Procedure` attribute. The following sections address these issues in more detail.

### 9.2.1 Data Representation

A/V pair lists are the main object scheme functions operate upon. Scheme is extremely convenient for representation of such objects. A Radius A/V pair is represented by a Scheme pair, e.g.

```
Session-Timeout = 10
```

is represented in Guile as

```
(cons "Session-Timeout" 10)
```

The CAR of the pair can contain either the attribute dictionary name, or the attribute number. Thus, the above pair may also be written in Scheme as

```
(cons 27 10)
```

(`Session-Timeout` corresponds to attribute number 27).

Lists of A/V pairs are represented by Scheme lists. For example, the following Radius pair list

```
User-Name = "jsmith",
        Password = "guessme",
        NAS-IP-Address = 10.10.10.1,
        NAS-Port-Id = 10
```

is written in Scheme as:

```
(list
  (cons "User-Name" "jsmith")
  (cons "Password" "guessme")
  (cons "NAS-IP-Address" "10.10.10.1")
  (cons "NAS-Port-Id" 10))
```

### 9.2.2 Authentication with Scheme

The Scheme procedure used for authentication must be declared as follows:

**auth-function** *request-list check-list reply-list* — Function Template

Its arguments are:

*request-list*
: The list of A/V pairs from the incoming request

*check-list*
: The list of A/V pairs from the LHS of the profile entry that matched the request.

*reply-list*
: The list of A/V pairs from the RHS of the profile entry that matched the request.

The function return value determines whether the authentication will succeed. The function must return either a boolean value or a pair. The return of `#t` causes authentication to succeed. The return of `#f` causes it to fail.

If the function wishes to add something to the reply A/V pairs, it should return a pair in the form:

```
(cons return-code list)
```

Where *return-code* is a boolean value of the same meaning as described above. The *list* is a list of A/V pairs to be added to the reply list. For example, the following function will always deny the authentication, returning appropriate message to the user:

```
(define (decline-auth request-list check-list reply-list)
  (cons #f
        (list
         (cons "Reply-Message"
               "\r\nSorry, you are not allowed to log in\r\n"))))
```

As a more constructive example, let's consider a function that allows the authentication only if a user name is found in its internal database.

```
(define staff-data
  (list
   (list "scheme"
         (cons
          (list (cons "NAS-IP-Address" "127.0.0.1"))
          (list (cons "Framed-MTU" "8096")))
         (cons
          '()
          (list (cons "Framed-MTU" "256"))))))

(define (auth req check reply)
  (let* ((username (assoc "User-Name" req))
         (reqlist (assoc username req))
         (reply-list '()))
    (if username
        (let ((user-data (assoc (cdr username) staff-data)))
          (rad-log L_INFO (format #f "~A" user-data))
          (if user-data
              (call-with-current-continuation
               (lambda (xx)
                 (for-each
                  (lambda (pair)
                    (cond
                     ((avl-match? req (car pair))
                      (set! reply-list (avl-merge
                                        reply-list
                                        (cdr pair)))
                      (xx #t))))
                  (cdr user-data))
                 #f)))))
    (cons
     #t
     reply-list)))
```

To trigger the invocation of the Scheme authentication function, assign its name to `Scheme-Procedure` attribute in RHS of a corresponding '`raddb/users`' profile. E.g.:

```
DEFAULT Auth-Type = SQL
        Scheme-Procedure = "auth"
```

### 9.2.3 Accounting with Scheme

The Scheme accounting procedure must be declared as follows:

**acct-function-name** *request-list* Function Template

Its arguments are:

*request-list*
: The list of A/V pairs from the incoming request

The function must return a boolean value. The accounting succeeds only if it returned `#t`.

Here is an example of Scheme accounting function. The function dumps the contents of the incoming request to a file:

```
(define radius-acct-file "/var/log/acct/radius")

(define (acct req)
  (call-with-output-file radius-acct-file
    (lambda (port)
      (for-each (lambda (pair)
                  (display (car pair) port)
                  (display "=" port)
                  (display (cdr pair) port)
                  (newline port))
                req)
      (newline port)))
  #t)
```

### 9.2.4 Radius-Specific Functions

**avl-delete** *av-list attr* Scheme Function

Delete from *av-list* the pairs with attribute *attr*.

**avl-merge** *dst src* Scheme Function

Merge *src* into *dst*.

**avl-match?** *target list* Scheme Function

Return `#t` if all pairs from *list* are present in *target*.

**rad-dict-name->attr** *name* Scheme Function

Return a dictionary entry for the given attribute *name* or `#f` if no such name was found in the dictionary.

A dictionary entry is a list in the form:

**dict-entry** *name-string attr-number type-number vendor* Scheme List

Where

*name-string*
: The attribute name

*value-number*
: The attribute number

*type-number*
: The attribute type

*vendor*
: is the vendor PEC, if the attribute is a Vendor-Specific one, or #f otherwise.

**rad-dict-value->name** *attr value* Scheme Function

Returns a dictionary name of the given *value* for an integer-type attribute *attr*. *attr* can be either an attribute number or its dictionary name.

**rad-dict-name->value** *attr value* Scheme Function

Convert a symbolic attribute value name into its integer representation

**rad-dict-pec->vendor** *pec* Scheme Function

Convert PEC to the vendor name

**rad-log-open** *prio* Scheme Function

Open radius logging to the severity level *prio*.

**rad-log-close** Scheme Function

Close radius logging channel open by a previous call to `rad-log-open`.

**rad-rewrite-execute-string** *string* Scheme Function

Interpret *string* as an invocation of a function in Rewrite language and execute it.

Return value: return of the corresponding Rewrite call, translated to the Scheme data type.

**rad-rewrite-execute** *arglist* Scheme Function

Execute a Rewrite language function. `(car` *arglist*`)` is interpreted as a name of the Rewrite function to execute, and `(cdr` *arglist*`)` as a list of arguments to be passed to it.

Return value: return of the corresponding Rewrite call, translated to the Scheme data type.

**rad-openlog** *ident option facility* Scheme Function

Scheme interface to the system `openlog()` call.

**rad-syslog** *prio text* Scheme Function

Scheme interface to the system `syslog()` call.

**rad-closelog** Scheme Function

Scheme interface to the system `closelog()` call.

**rad-utmp-putent** *status delay list radutmp_file radwtmp_file* Scheme Function

Write the supplied data into the radutmp file. If RADWTMP_FILE is not nil the constructed entry is also appended to WTMP_FILE. *list* is:

**utmp-entry** *user-name orig-name port-id port-type session-id caller-id framed-ip nas-ip proto* Scheme List

*user-name* The user name

*orig-name* Original user name from the request

*port-id* The value of `NAS-Port-Id` attribute.

*port-type* A number or character indicating the port type.

*session-id* Session ID.

*caller-id* The value of `Calling-Station-Id` attribute from the request.

*framed-ip* The framed IP address assigned to the user.

*nas-ip* The NAS IP address.

*proto* Number or character indicating type of the connection.

# 10 Utility Programs

## 10.1 Radwho

`Radwho` displays the list of users currently logged in by the Radius server.

Default output information is made compatible with that of the standard UNIX finger(1) utility. For each user the following information is displayed: login name, name, connection protocol, NAS port, login date, NAS name, assigned IP address or corresponding network name.

When used with '`-l`' option, the long output format is used. In this format the following information is output:

'`Login`' Login name of the user

'`SessionID`'
: Unique session ID assigned by the terminal server.

'`Proto`' Connection prototype.

'`Port`' Port number

'`When`' Login date and time

'`From`' The name of NAS that accepted the connection.

'`Location`'
: Framed IP address or the corresponding network name.

'`Caller`' Caller station ID ad reported by the NAS.

'`Duration`'
: Duration of the session.

### 10.1.1 Radwho Command Line Options

The following command line options can be used to modify the behavior of the program:

'`-A`'
'`--all`' Display the information about logged-out users as well. The logged-out users are shown with Proto field set to HUP.

'`-c`'
'`--calling-id`'
: Display the calling station ID in the second column.

'`-d NAME`'
'`--directory NAME`'
: Set the radius configuration directory name.

'`-D {short|abbr|full}`'
'`--date-formap {short|abbr|full}`'
: Set the date representation. By default dates are output as `DOW HH:MM`, where DOW means the day of week abbreviation, HH and MM mean hours and minutes respectively. This corresponds to option '`-D short`'. Other available formats are:

'abbr' Abbreviated date: MM/DD HH:MM, where MM is a two-digit month number, DD — a two-digit day of month.

'full' Full data output, like this:

```
Mon Dec 18 12:29:38 EET 2000
```

'-e STRING'
'--empty STRING'
Display any empty field as STRING. This is useful when the output of `radwho` is fed to some analyzing program, as it helps to keep the same number of columns on each line of output.

'-F'
'--finger'
Start in fingerd mode. In this mode `radwho` emulates the behavior of fingerd(8) utility. Use this option if starting `radwho` from the /etc/inetd.conf line like this:

```
finger stream tcp nowait nobody /usr/sbin/radwho radwho -fL
```

This mode is also enabled by default if `radwho` notices that its name (argv[0]) is 'fingerd' or 'in.fingerd'.

'-H'
'--no-header'
Don't display header line.

'-i'
'--session-id'
Display session ID instead of GECOS in the second column.

'-I {smart|ip|nodomain}'
'--ip-format {smart|ip|nodomain}'
Change IP address representation. The meaning of the argument is as follows:

'smart' Select the best representation. The following rules apply:

1. For a NAS use its short name from 'naslist'. If there is no short name, use its long name. If there is no long name either, go to 2.
2. Resolve IP address to FQDN.
3. If the IP cannot be resolved, use dotted-quad representation of the IP

'ip' Display IP in dotted-quad form.

'nodomain'
If the IP can be resolved to a fully qualified domain name, use the hostname part of it, i.e. any characters up to the first dot.

'-u'
'--local-also'
Display information about local users from the system 'utmp' file. May prove useful when running `radwho` as a finger daemon.

'-n'
'--no-resolve'
: Do not resolve IP address. It is a synonym for -I ip.

'-o FORMAT'
'--format FORMAT'
: Select customized output format. This can also be changed by setting the value of environment variable RADWHO_FORMAT. The format string is a comma-separated list of format specifications in one of the following forms:

*field* Output *field* with its default width, heading and alignment. The field names and corresponding default values are discussed in detail below.

*field*:*width*
: Output *field*, use column width *width*. If *width* starts with '+', the field will be right-aligned, if it starts with '-', the field will be left-aligned. Otherwise the default alignment will be used

*field*:*width*:*heading*
: The same as above, but also supplies the heading for the column

The field names are:

`login` Login name

`orig` Original login name as supplied with the request.

`port` NAS port number

`sid` The Session ID

`nas` The NAS name or IP address.

`ip` Framed IP address assigned to the user, if it is provided framed service.

`proto` Connection protocol. Its possible values are:
- 'PPP' for a point-to-point link
- 'SLIP' for a SLIP link
- 'HUP' for closed session
- 'shell' for shell user

`date` Date/time when the session started

`delay` Delay time Section 12.2.2 [Acct-Delay-Time], page 119.

`type` Entry type in decimal.

`ptype` Port type. This is one of the following letters:

| Type | Meaning |
|---|---|
| 'L' | Local connection |
| 'R' | Rlogin connection |
| 'S' | SLIP connection |
| 'C' | CSLIP connection |

| | |
|---|---|
| 'P' | PPP connection |
| 'A' | Auto PPP connection |
| 'E' | Telnet session |
| 'T' | "Clear TCP" connection |
| 'U' | TCP login service |
| '!' | Console session |
| 'X' | Shell |

`time` Total time of the session duration.

`clid` The calling station ID.

`uname` The GECOS field from local /etc/passwd, corresponding to the login name. If the user does not have a local account, his login name is output.

'-s'
'--secure'
Run in secure mode. Queries without a user name are rejected.

## 10.2 Radlast

The `radlast` utility lists sessions of specified users, NASes, NAS ports and hosts, in reverse time order. By default, each line of output contains the login name, NAS short name and port number from where the session was conducted, host IP address or name, the start and stop times for the session, and the duration of the session. If the session is still continuing, `radlast` will so indicate.

When specified the '-l' option, `radlast` produces long output. It includes following fields:

- login name
- NAS short name
- port number
- prototype
- port type
- Session ID
- Caller ID
- Framed IP address
- Session Start Time
- Session Stop Time
- Duration of the Session

### 10.2.1 Radlast Command Line Options

Use following command line options to control the behavior of `radlast` utility:

'-*number*'
'-c *number*'
'--count *number*'
: When given this option `radlast` will output at most this many lines of information.

'-f'
'--file *name*'
: Read the specified file instead of the default '`/var/log/radwtmp`'.

'-h *hostname*'
'--host *hostname*'
: Report the logins from given host. Host can be either a name or a "dotted quad" internet address.

'-n *shortname*'
'--nas *shortname*'
: Report the logins from given Network Access Server (NAS).

'-l'
'--long-format'
: "Long" output format. Report all the information stored in '`radwtmp`' file.

'-p *number*'
'--port *number*'
: Report the logins on a given port. The port may be specified either fully or abbreviated, e.g. `radlast -p 3` or `radlast -p S03`.

'-s'
'--show-seconds'
: Report the duration of the login session in seconds instead of the default days, hours and minutes.

'-t' The same as '`-p`'. This flag is provided for compatibility with last(1).

'-w'
'--wide' Widen the duration field to show seconds as well as the default days, hours and minutes.

If multiple arguments are given, the logical OR operation between them is assumed, i.e. the information selected by each argument is printed. This, however, does not apply to '`-c`' option. This option is always combined with the rest of command line by logical AND.

The pseudo-user '`~reboot`' logs in on every reboot of network access server.

If `radlast` is interrupted, it indicates to what date the search was progressed.

## 10.3 Raduse

The `raduse` utility shows the usage of dialup lines in the realtime.

### 10.3.1 Display

At the top of output the summary information is displayed. It consists of two lines. First line shows the statistic collection uptime and current date/time. Second line shows total number of lines, number of active lines, number of idle (inactive) lines and load percentage.

The dialup statistics is displayed in the area below. For each dialup line three lines of data are shown.

First line shows the network access server name, port number on that server, number of logins registered on this line, status of the line, amount of time the line keeps the current status, and date and time where the line has switched to the current status.

If the line is currently active, the status field displays login name of the user logged in on this line. If the line is inactive, the word '`[Idle]`' is displayed.

Second and third lines display active and idle usage summary. They show following data: total time the line is in the given state, maximum amount of time in this state, and starting date and time when maximum duration was perceived.

The example of default display:

```
uptime    90+20:35         Sun Dec 17 12:21
235 lines,  71 active, 164 idle. Pool load 0.30



max          001  2796 [idle]                 00:05 Sun Dec 17 12:16
             43+00:17     1+22:39 Fri Sep 22 18:04 - 16:44
             47+20:22       06:25 Thu Oct 05 02:24 - 08:50
max          002  2877 [idle]                 00:09 Sun Dec 17 12:11
             41+06:56       10:55 Sat Oct 28 21:20 - 07:15
             49+13:35       05:32 Mon Oct 02 00:33 - 06:05
max          003  3000 [idle]                 00:08 Sun Dec 17 12:12
             39+14:42       19:44 Thu Nov 02 14:52 - 10:36
             50+11:22       07:29 Wed Oct 11 23:30 - 06:59
max          004  2829 jsmith                 00:05 Sun Dec 17 12:15
             41+21:11     1+00:04 Sun Sep 24 12:17 - 12:21
             48+23:28       04:51 Sat Oct 07 03:42 - 08:33
max          005  2913 gray                   00:41 Sun Dec 17 11:40
             40+12:01       15:24 Mon Dec 11 19:18 - 10:43
             50+08:03       11:58 Wed Nov 29 13:43 - 01:41
max          006  3014 roland                 00:39 Sun Dec 17 11:41
             42+02:10       22:28 Sun Sep 24 13:46 - 12:15
             48+17:39       05:30 Fri Nov 24 01:57 - 07:28
max          007  2937 [idle]                 00:06 Sun Dec 17 12:15
```

This default display can be altered using command line options or interactive commands

### 10.3.2 Raduse Command Line Options

The following options modify the behavior of `raduse`:

'-b'
'--brief' Start up in brief mode. In this mode only first line of information for each dialup line is displayed.

'-d *count*'
'--display *count*'
Show only count displays, then exit. A display is considered to be one update of the screen.

'-D'
'--dump' Dump the statistics database to the standard output and then exit. This is for debugging purposes only.

'-I'
'--no-idle-lines'
Do not display idle lines. By default `raduse` displays all dialup lines.

'-i'
'--interactive'
Use interactive mode. In this mode any input is immediately read for processing. Section 10.3.3 [raduse commands], page 97 section for the description of commands usable in interactive mode. After processing each command the screen is updated immediately, no matter was the command understood or not. This mode is the default when the standard output is an intelligent terminal.

'-n'
'--no-interactive'
Use non-interactive mode.

'-s *num*'
'--delay *num*'
Specify delay in seconds between screen updates.

'-w'
'--widen' Widen the time display fields to show the seconds.

'-l'

'--list-nas'
List the names and IP numbers of network access servers and then exit.

'-h'
'--help' Display short usage summary.

### 10.3.3 Raduse Interactive Commands

The following commands are understood when `raduse` is in interactive mode. Some commands require an argument. Such commands are followed by the word *arg*. When `raduse` gets such command it displays a prompt and waits for user to enter the necessary data.

After processing each command the screen is updated immediately, no matter was the command understood or not.

⟨RET⟩ Refresh the screen immediately

⟨SPC⟩ Refresh the screen immediately

*C-l* Clear and redraw the display.

⟨^⟩ (Caret) go to the first page.

⟨b⟩ Toggle brief display mode.

*C-b* Move one page backwards.

*C-f* Move one page forwards.

⟨i⟩ Toggle idle line display on or off.

⟨j⟩ Move one line forwards.

⟨k⟩ Move one line backwards.

⟨G⟩
⟨$⟩ Move to the last page.

⟨q⟩ Quit the program

⟨s⟩ *arg* Change the number of seconds to delay between screen updates.

⟨t⟩ *arg* Display lines on a given Network Access Servers. The user is prompted to input the NAS names. The names should be separated by whitespace. The word '`all`' returns to display of all NAS lines.

## 10.4 Radzap

`radzap` searches the Radius accounting database for matching login records and closes them.

At least one of '`-n`', '`-p`' options or username must be specified. If they are used in conjunction, they are taken as if joined by logical AND operation.

'`radzap`' operates in two modes: silent and confirm. The silent mode is enabled by default. When run in this mode, radzap deletes every record that matched the search conditions given.

In confirm mode '`radzap`' will ask for a confirmation before zapping each matching record. Every line beginning with a 'y' is taken as positive response, otherwise it is taken as negative response.

The confirm mode is toggled by the command line option '`-c`'.

### Syntax

```
radzap [options] [username]
```

Options are:

'`-c`'
'`--confirm`'
    Enable confirm mode.

'-q'
'--quiet' Disable confirm mode.

'-h'
'--help' Display short help summary and exit.

'-n *NAME*'
'--nas *NAME*'
Specify NAS name to zap user from.

'-p *PORT*'
'--port *PORT*'
Specify the port number of the session to be zapped. The port number can be specified either in its full form, i.e `radzap -p S02` or in its short form, like `radzap -p 2`.

## 10.5 Radgrep

This utility allows to quickly lookup the user in the radius accounting database using a regular expression match.

`radgrep` scans the output of `radwho` utility and outputs only the lines that match given regular expressions.

### Syntax

`radgrep` accepts two sets of options separated by '`--`' (double-dash) sign. First subset is passed as command line to `radwho` utility. The second one is passed to `grep`.

## 10.6 Radping

This utility is a shell program that determines the user's framed IP address and runs `ping` on that address.

### Syntax

```
radping username
radping -c calling-station-id
```

The second way of invoking the program allows to use calling station ID in order to indicate the user.

## 10.7 Radauth

The `radauth` utility sends the Radius server Access-Request packet and displays the result it gets. It can be used to test the configuration files. The usage is:

```
raduse [-v] username password
```

The '`-v`' or '`--verbose`' option forces `radauth` to be verbose on output.

If you enter '`.`' (dot) instead of the password, the program will disable echoing on the screen, prompt you to enter the password, and turn the echoing on again, thus preventing the password from being compromised.

The program determines which Radius server to use, the authentication port number and shared secret following the procedure common for all client scripts (see Section 11.1 [config], page 105).

## 10.8 Radctl

`Radctl` is a control interface to `radiusd` daemon. It allows user running it to query `radiusd` about various aspects of its work and issue administrative commands to it.

`radctl -s` *command* `[`*args*`]`

Where *command* is a command telling `radctl` which actions to take, and *args* are optional arguments to the command. Only one command can be specified per invocation.

The valid *command*s are as follows:

`start [`*args*`]`
: If `radiusd` is not running already, it is started. When present, *args* are passed as the command line to the server.

`stop`
: Stops running `radiusd`.

`restart [`*args*`]`
: Stops the server and then starts it again. When present, *args* are passed as the command line to the server.

`reload`
: Causes running `radiusd` server to re-read its configuration files.

`dumpdb`
: Tells `radiusd` to dump its user hash table into the file '*radlog*`/radius.parse`'. This can be used for debugging configuration files.

`status`
: `radiusd` reports its memory usage statistics. The information is logged under `Info` log level.

`which`
: Reports the information about the running copy of `radiusd`.

## 10.9 Builddbm

### Usage

`Builddbm` converts the plaintext Radius users database into DBM files. Some versions of Radius daemon have used this to speed up the access to the users database.

However, with GNU Radius things go the other way around. The server reads entire plaintext database, converts it into internal form and stores into hash table which provides for fast access. Actually, using DBM version of the users database slows down the access unless the machine which runs Radius daemon is short of address space for the daemon to store the users database into.

### Syntax

When used without arguments, `builddbm` utility attempts to convert file '`raddb/users`' into '`raddb/users.db`' or '`raddb/users.pag`', '`raddb/users.dir`' pair, depending on the version of DBM library used.

If used with one argument, the argument is taken as the name of the plaintext database file to operate upon.

Use the following command line options to modify the operation of `buildbm`:

'`-d` *dir*' Specifies alternate directory for the Radius configuration files. This defaults to '`/usr/local/etc/raddb`'.

'`-h`' Outputs short usage summary and exits with 0 exit code.

## 10.10 Radscm: A guile interface to radius functions.

`Radscm` is a Scheme interpreter based on Guile with the addition of special functions and variables for communicating with `radiusd`. This chapter concentrates on the special features provided by `radscm`. Please refer to Guile documentation for information about Scheme and Guile See section "Overview" in *The Guile Reference Manual*.

### Variables

**%raddb-path** Variable

A path to radius configuration directory.

**rad-server-list** Function

A list of radius servers. Each element of the list is:

```
(list ID-STR HOST-STR SECRET-STR AUTH-NUM ACCT-NUM CNTL-NUM)
```

where:

| | |
|---|---|
| ID-STR | Server ID, |
| HOST-STR | Server hostname or IP address, |
| SECRET-STR | Shared secret key to use, |
| AUTH-NUM | Authentication port number, |
| ACCT-NUM | Accounting port number, |
| CNTL-NUM | Control channel port number. |

Thus, each entry can be used as an argument to rad-client-set-server or rad-client-add-server.

### Functions

**rad-send-internal** *PORT-NUMBER CODE-NUMBER PAIR-LIST* Function

Sends the request to currently selected server. Arguments are:

*PORT-NUMBER*
Port number to use. These values are allowed:

| | |
|---|---|
| 0 | Authentication port, |
| 1 | Accounting port, |
| 2 | Control port. |

The actual port numbers are those configured for the given server.

*CODE-NUMBER*
Request code.

*PAIR-LIST*
List of Attribute-value pairs. Each pair is either

```
(cons ATTR-NAME-STR VALUE)
```

or

```
(cons ATTR-NUMBER VALUE)
```

Return:

On success

```
(list RETURN-CODE-NUMBER PAIR-LIST)
```

On failure:

```
'()
```

**rad-send** *PORT-NUMBER CODE-NUMBER PAIR-LIST . VERBOSE* Function
Sends a radius request. Actually it does the same work as rad-send-internal but if VERBOSE is specified, the verbose report about interaction with the radius server is printed.

**rad-client-list-servers** Function
List currently configured servers. Two columns for each server are displayed: Server ID and IP address.

**rad-get-server** Function
Returns the ID of the currently selected server.

**rad-client-set-server** *LIST* Function
Selects for use the server described by LIST. A LIST should be:

```
(list ID-STR HOST-STR SECRET-STR AUTH-NUM ACCT-NUM CNTL-NUM)
```

where:

| | |
|---|---|
| ID-STR | Server ID, |
| HOST-STR | Server hostname or IP address, |
| SECRET-STR | Shared secret key to use, |
| AUTH-NUM | Authentication port number, |
| ACCT-NUM | Accounting port number, |
| CNTL-NUM | Control channel port number. |

**rad-client-add-server** *LIST* Function

Adds the server described by LIST to the list of active servers. A LIST should be:

```
(list ID-STR HOST-STR SECRET-STR AUTH-NUM ACCT-NUM CNTL-NUM)
```

where:

| | |
|---|---|
| ID-STR | Server ID, |
| HOST-STR | Server hostname or IP address, |
| SECRET-STR | Shared secret key to use, |
| AUTH-NUM | Authentication port number, |
| ACCT-NUM | Accounting port number, |
| CNTL-NUM | Control channel port number. |

**rad-read-no-echo** *PROMPT-STR* Function

Prints the given PROMPT-STR, disables echoing, reads a string up to the next newline character, restores echoing and returns the string entered. This is the interface to the C getpass(3) function.

**rad-client-source-ip** *IP-STR* Function

Sets the IP address to be used as source. IP-STR can be either an IP address in dotted-quad form or a hostname.

**rad-client-timeout** *NUMBER* Function

Sets the timeout for waiting to the server reply.

**rad-client-retry** *NUMBER* Function

Sets the number of retries for sending requests to a radius server.

**rad-format-code** *DEST-BOOL CODE-NUMBER* Function

Format a radius reply code into a human-readable form. DEST-BOOL has the same meaning as in format.

**rad-format-pair** *DEST-BOOL PAIR* Function

Format a radius attribute/value pair for output. DEST-BOOL has the same meaning as in format. PAIR is either

```
(cons NAME-STR VALUE)
```

or

```
(cons ATTR-NUMBER VALUE)
```

where VALUE may be of any type appropriate for the given attribute.

**rad-print-pairs** *DEST-BOOL PAIR-LIST* Function

Output the radius attribute/value pairs from the PAIR-LIST. DEST-BOOL has the same meaning as in format. PAIR-LIST is a list of pairs in the form

```
(cons NAME-STR VALUE)
```

or

```
(cons ATTR-NUMBER VALUE)
```

where VALUE may be of any type appropriate for the given attribute.

All "Reply-Message" pairs from the list are concatenated and displayed as one.

**rad-format-reply-msg** *PAIR-LIST . TEXT* Function

Concatenate and print text from all "Reply-Message" pairs from the PAIR-LIST. If TEXT is specified, it is printed before the concatenated text.

**rad-list-servers** Function

For each server from `rad-server-list` print its ID and hostname or IP address.

**rad-select-server** *ID-STR* Function

Select the server identified by ID-STR as a current server. The server data are looked up in rad-server-list variable.

**rad-add-server** *ID-STR* Function

Add the server identified by ID-STR to the list of current servers. The server data are looked up in rad-server-list variable.

# 11 Client Package

Beside the radius server and accompanying utilities, GNU Radius provides a set of utilities to be used as radius clients.

Following sections describe in detail the parts of the radius client package.

## 11.1 Client Configuration

All programs from the client package share the same configuration file: '`raddb/client.conf`'. The file uses simple line-oriented syntax. Empty lines are ignored, the '`#`' introduces an end-of-line comment.

The source IP address is introduced with `source_ip` statement. Its syntax is:

`source_ip` *ip-addr*

where *ip-addr* must be the IP address in "dotted-quad" notation.

The radius server where to send the requests to is introduced with `server` statement:

`server` *name ip-addr secret auth-port acct-port*

Its parts are:

*name* The server name. It is reserved for further use.

*ip-addr* The server IP address.

*secret* The shared secret to be used when sending requests to this server

*auth-port* Authentication port number.

*acct-port* Accounting port number.

If several `server` statement are present, they are tried in turn until any of them replies to the request.

The amount of time a client program waits for the reply from a server is configured using `timeout` statement:

`timepout` *number*

If the program does not receive any response within *number* seconds, it assumes the server does not respond and either retries the transmission or tries next available server. Number of retries is set with `retry` statement:

`retry` *number*

The example of '`raddb/client.conf`' follows:

```
server first 10.11.10.1 secret 1645 1646
server second 10.11.10.1 secret 1645 1646
source_ip 127.0.0.1
timeout 3
retry 5
```

## 11.2 radsession

`radsession` is a Guile script that sends authentication and accounting requests to the radius server. To invoke the script, run

```
radsession options action
```

Possible actions are:

'--auth' Send authentication request.

'--start' Send accounting start request.

'--stop' Send accounting stop request.

Options determine the contents of the request's pairlist. They are:

'-l STRING'
'--login STRING'
Set login name.

'-p STRING'
'--passwd STRING'
Set password.

'-n IP'
'--nas IP' Set the value of `NAS-IP-Address` attribute.

'-s STRING'
'--sid STRING'
Set the session id (`Acct-Session-Id` attribute).

'-P NUMBER'
'--port NUMBER'
Set the port number (`NAS-Port-Id` attribute).

'-h'
'--help' Print short usage message and exit.

'-v'
'--verbose'
Verbosely list the contents of the received reply.

## 11.3 nas.scm

`nas.scm` is a Guile program that allows to convert a GNU/Linux box into a NAS. It requires Guile version 1.4 or better and ppp version 2.3.7 or better.

To use it, you will basically need to do the following:

1. Create links:

   ```
   ln -s libexec/nas.scm /etc/ppp/ip-up
   ln -s libexec/nas.scm /etc/ppp/ip-down
   ```

   Here, *libexec* denotes the location of your libexec directory, where `nas.scm` is installed. If not overridden at configure time, it defaults to '*prefix*/libexec'. These links assure the `ppp` will invoke `nas.scm` when the user's session starts and ends, thus giving it a possibility to send accounting requests.

2. Configure file '`raddb/client.conf`'
3. Edit file '`raddb/nas.rc`'. The supplied '`nas.rc`' template is tailored to work in most environments. The only variables you may need to change are: `nas-log-facility`, specifying the syslog facility to be used for logging and `pppd-args`, keeping the arguments to be given to `ppp`.
4. Configure your '`/etc/inittab`' and `getty`.

   For example, if you use `mgetty`, then the '`inittab`' entries for dial-up lines will look like:

   ```
   d0:345:respawn:/sbin/mgetty ttyS0 vt100
   d1:345:respawn:/sbin/mgetty ttyS1 vt100
   ...
   ```

   The mgetty's '`login.config`' will then contain the following line:

   ```
   *       -       -       /usr/local/libexec/nas.scm
   ```

   If you use `agetty`, then the '`inittab`' will contain (with the long lines split for readability):

   ```
   d0:345:respawn:/sbin/agetty -mt60 \
      -l /usr/local/libexec/nas.scm 38400,19200,9600 \
      ttyS0 vt100
   d1:345:respawn:/sbin/agetty -mt60 \
      -l /usr/local/libexec/nas.scm 38400,19200,9600 \
      ttyS1 vt100
   ...
   ```

## 11.4 pam_radius.so

`pam_radius.so` is a PAM module for radius authentication. The module understands following command line options:

'`audit`'
: Enable audit information.

'`debug[=`*level*`]`'
: Enable debugging information. The higher *level* is, the more debugging info is output. When omitted, *level* defaults to 100. *Please note*, that debugging levels equal to or greater than 10 compromise users' passwords, so use them sparingly.

'`use_authtok`'
: Use authentication token passed from the previous module in stack.

'`confdir=`*path*'
: Look for configuration files in *path*. Default is '`$sysconfdir/etc/raddb`'.

'`service_type=`*type*'
: Add `Service-Type=`*type* to the authentication request. *type* must be a valid value, described in dictionary file.

`pam_radius.so` module logs its messages under `LOG_AUTH` syslog facility.

# 12 Attribute List

The following sections describe the most frequently used RADIUS attributes. Each attribute is described as follows:

`ATTRIBUTE` *name* *value* *type*

| | |
|---|---|
| Users: | *user-flags* |
| Hints: | *hints-flags* |
| Huntgroups: | *huntgroup-flags* |
| Additivity: | *additivity* |
| Proxy propagated: | *prop* |

These values have the following meaning:

*name* The attribute name.

*value* The attribute number.

*type* The attribute type.

*user-flags* Syntax flags defining in which part of '`raddb/users`' entry this attribute may be used. The flags consist of two letters: '`L`' means the attribute can be used in LHS, '`R`' means it can be used in RHS.

*hints-flags* Syntax flags defining in which part of '`raddb/hints`' entry this attribute may be used.

*huntgroup-flags*
Syntax flags defining in which part of '`raddb/huntgroups`' entry this attribute may be used.

*additivity* *Additivity* of the attribute determines what happens if a rule attempts to add to the pair list the attribute, which is already present in this list. Depending on its value, the actions of the server are:

Append New attribute is appended to the end of the list.

Replace New attribute replaces the old.

Drop New attribute is dropped. The old one remains in the list.

*prop* Is the attribute propagated back to the NAS if the server works in proxy mode.

The value of N/A in any of this fields signifies "not applicable".

## 12.1 Authentication Attributes

These are the attributes the NAS uses in authentication packets and expects to get back in authentication replies. These can be used in matching rules.

### 12.1.1 User-Name

ATTRIBUTE User-Name 1 string

| | |
|---|---|
| Users: | LR |
| Hints: | -R |
| Huntgroups: | LR |
| Additivity: | Replace |
| Proxy propagated: | Yes |

This Attribute indicates the name of the user to be authenticated or accounted. It is used in Access-Request and Accounting attributes. The length of the username is usually limited by some arbitrary value. By default, Radius supports usernames up to 32 characters long. This value can be modified by redefining `RUT_USERNAME` macro in include/radutmp.h file in the distribution directory and recompiling the program.

Some NASes have peculiarities about sending long usernames. For example, Specialix Jetstream 8500 24 port access server inserts a '/' character after the 10th character if the username is longer than 10 characters. In such cases, we recommend to apply rewrite functions in order to bring username to its "normal" form (see Section 4.12 [rewrite file], page 57).

### 12.1.2 Password

ATTRIBUTE Password 2 string

| | |
|---|---|
| Users: | L- |
| Hints: | -- |
| Huntgroups: | -- |
| Additivity: | N/A |
| Proxy propagated: | No |

This Password attribute indicates the password of the user to be authenticated, or the user's input following an Access-Challenge. It is only used in Access-Request packets.

On transmission, the password is hidden. The password is first padded at the end with nulls to a multiple of 16 octets. A one- way MD5 hash is calculated over a stream of octets consisting of the shared secret followed by the Request Authenticator. This value is XORed with the first 16 octet segment of the password and placed in the first 16 octets of the String field of the Password Attribute.

If the password is longer than 16 characters, a second one-way MD5 hash is calculated over a stream of octets consisting of the shared secret followed by the result of the first xor. That hash is XORed with the second 16 octet segment of the password and placed in the second 16 octets of the String field of the Password Attribute.

If necessary, this operation is repeated, with each xor result being used along with the shared secret to generate the next hash to xor the next segment of the password, to no more than 128 characters.

### 12.1.3 CHAP-Password

ATTRIBUTE CHAP-Password 3 string

| | |
|---|---|
| Users: | L- |

| | |
|---|---|
| Hints: | `--` |
| Huntgroups: | `--` |
| Additivity: | N/A |
| Proxy propagated: | No |

This Attribute indicates the response value provided by a PPP Challenge-Handshake Authentication Protocol (CHAP) user in response to the challenge. It is only used in Access-Request packets.

The CHAP challenge value is found in the CHAP-Challenge Attribute (60) if present in the packet, otherwise in the Request Authenticator field.

### 12.1.4 NAS-IP-Address

```
ATTRIBUTE  NAS-IP-Address  4  ipaddr
```

| | |
|---|---|
| Users: | `L-` |
| Hints: | `-R` |
| Huntgroups: | `LR` |
| Additivity: | Append |
| Proxy propagated: | No |

This Attribute indicates the identifying IP address of the NAS which is requesting authentication of the user. It is only used in Access-Request packets. Each Access-Request packet should contain either `NAS-IP-Address` or `NAS-Identifier` attribute Section 12.1.25 [NAS-Identifier], page 118.

### 12.1.5 NAS-Port-Id

```
ATTRIBUTE  NAS-Port-Id  5  integer
```

| | |
|---|---|
| Users: | `LR` |
| Hints: | `-R` |
| Huntgroups: | `LR` |
| Additivity: | Append |
| Proxy propagated: | No |

This attribute indicates the physical port number of the NAS which is authenticating the user. It is only used in Access-Request packets. Note that this is using "port" in its sense of a physical connection on the NAS, not in the sense of a TCP or UDP port number.

Some NASes try to encode various information in the `NAS-Port-Id` attribute value. For example MAX Ascend terminal server constructs `NAS-Port-Id` concatenating line type (one-digit), line number (two-digits), and the channel number (two-digits) thus producing a 5-digit port number. In order to "normalize" such encoded port numbers we recommend to use a rewrite function (see Section 4.12 [rewrite file], page 57). A rewrite function for MAX Ascend servers is provided in the distribution.

### 12.1.6 Service-Type

```
ATTRIBUTE  Service-Type  6  integer
```

| | |
|---|---|
| Users: | `LR` |
| Hints: | `-R` |

| | |
|---|---|
| Huntgroups: | `LR` |
| Additivity: | Replace |
| Proxy propagated: | Yes |

```
VALUE      Service-Type      Login-User            1
VALUE      Service-Type      Framed-User           2
VALUE      Service-Type      Callback-Login-User   3
VALUE      Service-Type      Callback-Framed-User  4
VALUE      Service-Type      Outbound-User         5
VALUE      Service-Type      Administrative-User   6
VALUE      Service-Type      NAS-Prompt-User       7
VALUE      Service-Type      Authenticate-Only     8
VALUE      Service-Type      Call-Check            10
```

This attribute indicates the type of service the user has requested, or the type of service to be provided. It may be used in both Access-Request and Access-Accept packets.

When used in an Access-Request the Service type represents a hint to the Radius server that the NAS has reason to believe the user would prefer the kind of service indicated.

When used in an Access-Accept, the Service type is an indication to the NAS that the user must be provided this type of service.

The meaning of various service-types is as follows:

`Login-User`
: The user should be connected to a host.

`Framed-User`
: A Framed Protocol should be started for the User, such as PPP or SLIP. The `Framed-IP-Address` attribute (see Section 12.1.8 [Framed-IP-Address], page 113) would supply the IP address to be used.

`Callback-Login-User`
: The user should be disconnected and called back, then connected to a host.

`Callback-Framed-User`
: The user should be disconnected and called back, then a Framed Protocol should be started for the User, such as PPP or SLIP.

`Outbound-User`
: The user should be granted access to outgoing devices.

`Administrative-User`
: The user should be granted access to the administrative interface to the NAS from which privileged commands can be executed.

`NAS-Prompt`
: The user should be provided a command prompt on the NAS from which non-privileged commands can be executed.

`Authenticate-Only`
: Only Authentication is requested, and no authorization information needs to be returned in the Access-Accept

`Call-Check`
`Callback-NAS-Prompt`
: The user should be disconnected and called back, then provided a command prompt on the NAS from which non-privileged commands can be executed.

### 12.1.7 Framed-Protocol

```
ATTRIBUTE  Framed-Protocol  7  integer
```

| | |
|---|---|
| Users: | `LR` |
| Hints: | `-R` |
| Huntgroups: | `LR` |
| Additivity: | Replace |
| Proxy propagated: | Yes |

```
VALUE     Framed-Protocol   PPP              1
VALUE     Framed-Protocol   SLIP             2
```

This Attribute indicates the framing to be used for framed access. It may be used in both Access-Request and Access-Accept packets.

### 12.1.8 Framed-IP-Address

```
ATTRIBUTE  Framed-IP-Address  8  ipaddr
```

| | |
|---|---|
| Users: | `LR` |
| Hints: | `-R` |
| Huntgroups: | `LR` |
| Additivity: | Replace |
| Proxy propagated: | No |

This Attribute indicates the address to be configured for the user. It may be used in Access-Accept packets. It may be used in an Access-Request packet as a hint by the NAS to the server that it would prefer that address, but the server is not required to honor the hint.

The value `0xFFFFFFFF` (`255.255.255.255`) indicates that the NAS should allow the user to select an address. The value `0xFFFFFFFE` (`255.255.255.254`) indicates that the NAS should select an address for the user (e.g. assigned from a pool of addresses kept by the NAS). Other valid values indicate that the NAS should use that value as the user's IP address.

When used in a RHS, the value of this attribute can optionally be followed by a plus sign. This usage means that the value of NAS-Port-Id must be added to this IP address before replying. For example

```
Framed-IP-Address = 10.10.0.1+
```

Also Section 12.3.13 [Add-Port-To-IP-Address], page 128.

### 12.1.9 Framed-IP-Netmask

```
ATTRIBUTE  Framed-IP-Netmask  9  ipaddr
```

| | |
|---|---|
| Users: | `LR` |
| Hints: | `-R` |

| | |
|---|---|
| Huntgroups: | `LR` |
| Additivity: | Replace |
| Proxy propagated: | No |

This Attribute indicates the IP netmask to be configured for the user when the user is a router to a network. It may be used in Access-Accept packets. It may be used in an Access-Request packet as a hint by the NAS to the server that it would prefer that netmask, but the server is not required to honor the hint.

### 12.1.10 Framed-Routing

```
ATTRIBUTE  Framed-Routing  10  integer
```

| | |
|---|---|
| Users: | `-R` |
| Hints: | `-R` |
| Huntgroups: | `-R` |
| Additivity: | Replace |
| Proxy propagated: | No |

```
VALUE      Framed-Routing    None                0
VALUE      Framed-Routing    Broadcast           1
VALUE      Framed-Routing    Listen              2
VALUE      Framed-Routing    Broadcast-Listen    3
```

This Attribute indicates the routing method for the user, when the user is a router to a network. It is only used in Access-Accept packets.

### 12.1.11 Framed-MTU

```
ATTRIBUTE  Framed-MTU  12  integer
```

| | |
|---|---|
| Users: | `LR` |
| Hints: | `-R` |
| Huntgroups: | `-R` |
| Additivity: | Replace |
| Proxy propagated: | Yes |

This Attribute indicates the Maximum Transmission Unit to be configured for the user, when it is not negotiated by some other means (such as PPP). It is only used in Access-Accept packets.

### 12.1.12 Framed-Compression

```
ATTRIBUTE  Framed-Compression  13  integer
```

| | |
|---|---|
| Users: | `LR` |
| Hints: | `-R` |
| Huntgroups: | `LR` |
| Additivity: | Replace |
| Proxy propagated: | Yes |

```
VALUE      Framed-Compression  None                  0
VALUE      Framed-Compression  Van-Jacobson-TCP-IP   1
```

This Attribute indicates a compression protocol to be used for the link. It may be used in Access-Accept packets. It may be used in an Access-Request packet as a hint to the server that the NAS would prefer to use that compression, but the server is not required to honor the hint.

More than one compression protocol Attribute may be sent. It is the responsibility of the NAS to apply the proper compression protocol to appropriate link traffic.

### 12.1.13 Reply-Message

```
ATTRIBUTE  Reply-Message  18  string
```

| | |
|---|---|
| Users: | `-R` |
| Hints: | `--` |
| Huntgroups: | `--` |
| Additivity: | Append |
| Proxy propagated: | Yes |

This Attribute indicates text which may be displayed to the user.

When used in an Access-Accept, it is the success message.

When used in an Access-Reject, it is the failure message. It may indicate a dialog message to prompt the user before another Access-Request attempt.

When used in an Access-Challenge, it may indicate a dialog message to prompt the user for a response.

Multiple `Reply-Message` attributes may be included and if any are displayed, they must be displayed in the same order as they appear in the packet.

### 12.1.14 Callback-Number

```
ATTRIBUTE  Callback-Number  19  string
```

| | |
|---|---|
| Users: | `-R` |
| Hints: | `--` |
| Huntgroups: | `--` |
| Additivity: | Replace |
| Proxy propagated: | No |

This Attribute indicates a dialing string to be used for callback. It may be used in Access-Accept packets. It may be used in an Access-Request packet as a hint to the server that a Callback service is desired, but the server is not required to honor the hint.

### 12.1.15 Callback-Id

```
ATTRIBUTE  Callback-Id  20  string
```

| | |
|---|---|
| Users: | `-R` |
| Hints: | `--` |
| Huntgroups: | `--` |
| Additivity: | Replace |
| Proxy propagated: | No |

This Attribute indicates the name of a place to be called, to be interpreted by the NAS. It may be used in Access-Accept packets.

### 12.1.16 Framed-Route

```
ATTRIBUTE  Framed-Route  22  string
```

| | |
|---|---|
| Users: | `-R` |
| Hints: | `--` |
| Huntgroups: | `--` |
| Additivity: | Replace |
| Proxy propagated: | No |

This Attribute provides routing information to be configured for the user on the NAS. It is used in the Access-Accept packet and can appear multiple times.

### 12.1.17 State

```
ATTRIBUTE  State  24  string
```

| | |
|---|---|
| Users: | `LR` |
| Hints: | `LR` |
| Huntgroups: | `LR` |
| Additivity: | Append |
| Proxy propagated: | No |

This Attribute is available to be sent by the server to the client in an Access-Challenge and MUST be sent unmodified from the client to the server in the new Access-Request reply to that challenge, if any.

This Attribute is available to be sent by the server to the client in an Access-Accept that also includes a Termination-Action Attribute with the value of RADIUS-Request. If the NAS performs the Termination-Action by sending a new Access-Request upon termination of the current session, it MUST include the State attribute unchanged in that Access-Request.

In either usage, no interpretation by the client should be made. A packet may have only one `State` Attribute.

### 12.1.18 Class

```
ATTRIBUTE  Class  25  string
```

| | |
|---|---|
| Users: | `LR` |
| Hints: | `LR` |
| Huntgroups: | `LR` |
| Additivity: | Append |
| Proxy propagated: | No |

This Attribute is available to be sent by the server to the client in an Access-Accept and should be sent unmodified by the client to the accounting server as part of the Accounting-Request packet if accounting is supported.

### 12.1.19 Vendor-Specific

```
ATTRIBUTE  Vendor-Specific  26  string
```

| | |
|---|---|
| Users: | `LR` |
| Hints: | `-R` |

| | |
|---|---|
| Huntgroups: | `-R` |
| Additivity: | Append |
| Proxy propagated: | No |

This Attribute is available to allow vendors to support their own extended Attributes not suitable for general usage.

### 12.1.20 Session-Timeout

```
ATTRIBUTE  Session-Timeout  27  integer
```

| | |
|---|---|
| Users: | `-R` |
| Hints: | `--` |
| Huntgroups: | `--` |
| Additivity: | Replace |
| Proxy propagated: | Yes |

This Attribute sets the maximum number of seconds of service to be provided to the user before termination of the session or prompt. The server may send this attribute to the client in an Access-Accept or Access-Challenge.

### 12.1.21 Idle-Timeout

```
ATTRIBUTE  Idle-Timeout  28  integer
```

| | |
|---|---|
| Users: | `-R` |
| Hints: | `--` |
| Huntgroups: | `--` |
| Additivity: | Replace |
| Proxy propagated: | Yes |

This Attribute sets the maximum number of consecutive seconds of idle connection allowed to the user before termination of the session or prompt. The server may send this attribute to the client in an Access-Accept or Access-Challenge.

### 12.1.22 Termination-Action

```
ATTRIBUTE  Termination-Action  29  integer
```

| | |
|---|---|
| Users: | `LR` |
| Hints: | `-R` |
| Huntgroups: | `-R` |
| Additivity: | Replace |
| Proxy propagated: | No |

```
VALUE      Termination-Action  Default            0
VALUE      Termination-Action  RADIUS-Request     1
```

This Attribute indicates what action the NAS should take when the specified service is completed. It is only used in Access-Accept packets.

### 12.1.23 Called-Station-Id

```
ATTRIBUTE  Called-Station-Id  30  string
```

| | |
|---|---|
| Users: | `L-` |
| Hints: | `-R` |
| Huntgroups: | `LR` |
| Additivity: | Append |
| Proxy propagated: | No |

This Attribute allows the NAS to send in the Access-Request packet the phone number that the user called, using Dialed Number Identification (DNIS) or similar technology. Note that this may be different from the phone number the call comes in on. It is only used in Access-Request packets.

### 12.1.24 Calling-Station-Id

```
ATTRIBUTE  Calling-Station-Id  31  string
```

| | |
|---|---|
| Users: | `L-` |
| Hints: | `-R` |
| Huntgroups: | `LR` |
| Additivity: | Append |
| Proxy propagated: | No |

This Attribute allows the NAS to send in the Access-Request packet the phone number that the call came from, using Automatic Number Identification (ANI) or similar technology. It is only used in Access-Request packets.

### 12.1.25 NAS-Identifier

```
ATTRIBUTE  NAS-Identifier  32  string
```

| | |
|---|---|
| Users: | `L-` |
| Hints: | `-R` |
| Huntgroups: | `LR` |
| Additivity: | Append |
| Proxy propagated: | No |

This Attribute contains a string identifying the NAS originating the Access-Request. It is only used in Access-Request packets. Either `NAS-IP-Address` or `NAS-Identifier` should be present in an Access-Request packet.

See Section 12.1.4 [NAS-IP-Address], page 111.

### 12.1.26 NAS-Port-Type

```
ATTRIBUTE  NAS-Port-Type  61  integer
```

| | |
|---|---|
| Users: | `--` |
| Hints: | `--` |
| Huntgroups: | `--` |
| Additivity: | Append |
| Proxy propagated: | No |

```
VALUE      NAS-Port-Type     Async               0
VALUE      NAS-Port-Type     Sync                1
VALUE      NAS-Port-Type     ISDN                2
VALUE      NAS-Port-Type     ISDN-V120           3
VALUE      NAS-Port-Type     ISDN-V110           4
```

This Attribute indicates the type of the physical port of the NAS which is authenticating the user. It can be used instead of or in addition to the `NAS-Port-Id` Section 12.1.5 [NAS-Port-Id], page 111 attribute. It is only used in Access-Request packets. Either `NAS-Port` or `NAS-Port-Type` or both should be present in an Access-Request packet, if the NAS differentiates among its ports.

## 12.2 Accounting Attributes

These are attributes the NAS sends along with accounting requests. These attributes can not be used in matching rules.

### 12.2.1 Acct-Status-Type

```
ATTRIBUTE  Acct-Status-Type  40  integer
```

| | |
|---|---|
| Users: | -- |
| Hints: | -- |
| Huntgroups: | -- |
| Additivity: | N/A |
| Proxy propagated: | N/A |

```
VALUE         Acct-Status-Type       Start               1
VALUE         Acct-Status-Type       Stop                2
VALUE         Acct-Status-Type       Alive               3
VALUE         Acct-Status-Type       Accounting-On       7
VALUE         Acct-Status-Type       Accounting-Off      8
```

This attribute indicates whether this Accounting-Request marks the beginning of the user service (Start) or the end (Stop).

It may also be used to mark the start of accounting (for example, upon booting) by specifying Accounting-On and to mark the end of accounting (for example, just before a scheduled reboot) by specifying Accounting-Off.

A special value Alive or Interim-Update indicates the packet that contains some additional data to the initial Start record or to the last Alive record.

### 12.2.2 Acct-Delay-Time

```
ATTRIBUTE  Acct-Delay-Time  41  integer
```

| | |
|---|---|
| Users: | -- |
| Hints: | -- |
| Huntgroups: | -- |
| Additivity: | N/A |
| Proxy propagated: | N/A |

This attribute indicates how many seconds the client has been trying to send this record for, and can be subtracted from the time of arrival on the server to find the approximate time of the event generating this Accounting-Request. (Network transit time is ignored.)

### 12.2.3 Acct-Input-Octets

```
ATTRIBUTE  Acct-Input-Octets  42  integer
```

| | |
|---|---|
| Users: | -- |
| Hints: | -- |
| Huntgroups: | -- |
| Additivity: | N/A |
| Proxy propagated: | N/A |

This attribute indicates how many octets have been received from the port over the course of this service being provided, and can only be present in Accounting-Request records where the Acct-Status-Type is set to Stop.

### 12.2.4 Acct-Output-Octets

```
ATTRIBUTE  Acct-Output-Octets  43  integer
```

| | |
|---|---|
| Users: | -- |
| Hints: | -- |
| Huntgroups: | -- |
| Additivity: | N/A |
| Proxy propagated: | N/A |

This attribute indicates how many octets have been sent to the port in the course of delivering this service, and can only be present in Accounting-Request records where the Acct-Status-Type is set to Stop.

### 12.2.5 Acct-Session-Id

```
ATTRIBUTE  Acct-Session-Id  44  string
```

| | |
|---|---|
| Users: | -- |
| Hints: | -- |
| Huntgroups: | -- |
| Additivity: | N/A |
| Proxy propagated: | N/A |

This attribute is a unique Accounting ID to make it easy to match start and stop records in a log file. The start and stop records for a given session must have the same Acct-Session-Id. An Accounting-Request packet must have an Acct-Session-Id. An Access-Request packet may have an Acct-Session-Id; if it does, then the NAS must use the same Acct-Session-Id in the Accounting-Request packets for that session.

### 12.2.6 Acct-Authentic

```
ATTRIBUTE  Acct-Authentic  45  integer
```

| | |
|---|---|
| Users: | -- |

| | |
|---|---|
| Hints: | -- |
| Huntgroups: | -- |
| Additivity: | N/A |
| Proxy propagated: | N/A |

```
VALUE          Acct-Authentic          RADIUS         1
VALUE          Acct-Authentic          Local          2
VALUE          Acct-Authentic          Remote         3
```

This attribute may be included in an Accounting-Request to indicate how the user was authenticated, whether by Radius, the NAS itself, or another remote authentication protocol. Users who are delivered service without being authenticated should not generate Accounting records.

### 12.2.7 Acct-Session-Time

```
ATTRIBUTE  Acct-Session-Time  46  integer
```

| | |
|---|---|
| Users: | -- |
| Hints: | -- |
| Huntgroups: | -- |
| Additivity: | N/A |
| Proxy propagated: | N/A |

This attribute indicates how many seconds the user has received service for, and can only be present in Accounting-Request records where the Acct-Status-Type is set to Stop.

### 12.2.8 Acct-Input-Packets

```
ATTRIBUTE  Acct-Input-Packets  47  integer
```

| | |
|---|---|
| Users: | -- |
| Hints: | -- |
| Huntgroups: | -- |
| Additivity: | N/A |
| Proxy propagated: | N/A |

This attribute indicates how many packets have been received from the port over the course of this service being provided to a Framed User, and can only be present in Accounting-Request records where the Acct-Status-Type is set to Stop.

### 12.2.9 Acct-Output-Packets

```
ATTRIBUTE  Acct-Output-Packets  48  integer
```

| | |
|---|---|
| Users: | -- |
| Hints: | -- |
| Huntgroups: | -- |
| Additivity: | N/A |
| Proxy propagated: | N/A |

This attribute indicates how many packets have been sent to the port in the course of delivering this service to a Framed User, and can only be present in Accounting-Request records where the Acct-Status-Type is set to Stop.

### 12.2.10 Acct-Terminate-Cause

```
ATTRIBUTE  Acct-Terminate-Cause  49  integer
```

| | |
|---|---|
| Users: | -- |
| Hints: | -- |
| Huntgroups: | -- |
| Additivity: | N/A |
| Proxy propagated: | N/A |

```
VALUE        Acct-Terminate-Cause    User-Request            1
VALUE        Acct-Terminate-Cause    Lost-Carrier            2
VALUE        Acct-Terminate-Cause    Lost-Service            3
VALUE        Acct-Terminate-Cause    Idle-Timeout            4
VALUE        Acct-Terminate-Cause    Session-Timeout         5
VALUE        Acct-Terminate-Cause    Admin-Reset             6
VALUE        Acct-Terminate-Cause    Admin-Reboot            7
VALUE        Acct-Terminate-Cause    Port-Error              8
VALUE        Acct-Terminate-Cause    NAS-Error               9
VALUE        Acct-Terminate-Cause    NAS-Request             10
VALUE        Acct-Terminate-Cause    NAS-Reboot              11
VALUE        Acct-Terminate-Cause    Port-Unneeded           12
VALUE        Acct-Terminate-Cause    Port-Preempted          13
VALUE        Acct-Terminate-Cause    Port-Suspended          14
VALUE        Acct-Terminate-Cause    Service-Unavailable     15
VALUE        Acct-Terminate-Cause    Callback                16
VALUE        Acct-Terminate-Cause    User-Error              17
VALUE        Acct-Terminate-Cause    Host-Request            18
```

This attribute indicates how the session was terminated, and can only be present in Accounting-Request records where the Acct- Status-Type is set to Stop.

## 12.3 Radius Internal Attributes

These are attributes, used by GNU Radius during the processing of a request. They are never returned to NAS. Mostly, they are used in matching rules.

### 12.3.1 Auth-Type

```
ATTRIBUTE  Auth-Type  1000  integer
```

| | |
|---|---|
| Users: | L- |
| Hints: | -R |
| Huntgroups: | -R |
| Additivity: | Append |
| Proxy propagated: | No |

```
VALUE      Auth-Type        Local             0
VALUE      Auth-Type        System            1
VALUE      Auth-Type        Crypt-Local       3
VALUE      Auth-Type        Reject            4
VALUE      Auth-Type        SQL               252
```

```
VALUE      Auth-Type          Pam                  253
VALUE      Auth-Type          Accept               254
```

This attribute tells the server which type of authentication to apply to a particular user. It can be used in LHS of the user's profile. See Chapter 5 [Authentication], page 63.

Radius interprets values of `Auth-Type` attribute as follows:

`Local` The value of the `Password` attribute from the record is taken as a cleantext password and is compared against the `Password` value from the input packet.

`System` This means that a user's password is stored in a system password type. Radius queries the operating system to determine if the username/password supplied in the incoming packet are OK.

`Crypt-Local`
The value of the `Password` attribute from the record is taken as an MD5 hash on the user's password. Radius generates MD5 hash on the supplied `Password` value and compares both strings.

`Reject` Authentication fails.

`Accept` Authentication succeeds.

`SQL`
`Mysql` The MD5-encrypted user's password is queried from the SQL database Section 5.6 [SQL Auth], page 64. `Mysql` is an alias maintained for compatibility with other versions of Radius.

`Pam` The username/password combination is checked using PAM.

### 12.3.2 Auth-Data

```
ATTRIBUTE  Auth-Data  2006  string
```

| | |
|---|---|
| Users: | `L-` |
| Hints: | `-R` |
| Huntgroups: | `-R` |
| Additivity: | Replace |
| Proxy propagated: | N/A |

The `Auth-Data` can be used to pass additional data to the authentication methods that need them. In version 0.96 of GNU Radius, this attribute may be used in conjunction with `SQL` and `Pam` authentication types. When used with `Pam` authentication type, this attribute holds the name of PAM service to use. This attribute is temporarily appended to the authentication request, so its value can be referenced to as `%C{Auth-Data}`. See Section 4.11.2 [Authentication Server Parameters], page 52, for an example of of using `Auth-Data` attribute in '`raddb/sqlserver`':

### 12.3.3 Menu

```
ATTRIBUTE  Menu  1001  string
```

| | |
|---|---|
| Users: | `-R` |
| Hints: | `--` |

| | |
|---|---|
| Huntgroups: | `--` |
| Additivity: | Replace |
| Proxy propagated: | No |

This attribute should be used in the RHS. If it is used, it should be the only reply item.

The `Menu` attribute specifies the name of the menu to be presented to the user. The corresponding menu code is looked up in '`RADIUS_DIR/menus/`' directory (see Section 4.13 [menus file], page 57).

### 12.3.4 Termination-Menu

```
ATTRIBUTE  Termination-Menu  1002  string
```

| | |
|---|---|
| Users: | `-R` |
| Hints: | `--` |
| Huntgroups: | `--` |
| Additivity: | Replace |
| Proxy propagated: | No |

This attribute should be used in the RHS. If it is used, it should be the only reply item.

The `Termination-Menu` specifies the name of the menu file to be presented to the user after finishing his session. The corresponding menu code is looked up in '`RADIUS_DIR/menus/`' directory (see Section 4.13 [menus file], page 57).

### 12.3.5 Prefix

```
ATTRIBUTE  Prefix  1003  string
```

| | |
|---|---|
| Users: | `L-` |
| Hints: | `L-` |
| Huntgroups: | `LR` |
| Additivity: | Append |
| Proxy propagated: | No |

The `Prefix` attribute indicates the prefix which the username should contain in order for a particular record in the profile to be matched. This attribute should be specified in LHS of the '`users`' or '`hints`' file.

For example, if the '`users`' file contained:

```
DEFAULT Prefix = "U", Auth-Type = System
                 Service-Type = Login-User
```

then usernames '`Ugray`' and '`Uyoda`' would match this record, whereas '`gray`' and '`yoda`' would not.

Both `Prefix` and `Suffix` attributes may be specified in a profile. In this case the record is matched only if the username contains both prefix and suffix specified.

Section 12.3.6 [Suffix], page 124 Section 12.3.11 [Strip-User-Name], page 126

### 12.3.6 Suffix

`ATTRIBUTE Suffix 1004 string`

| | |
|---|---|
| Users: | `L-` |
| Hints: | `L-` |
| Huntgroups: | `LR` |
| Additivity: | Append |
| Proxy propagated: | No |

The `Suffix` attribute indicates the suffix which the username should contain in order for a particular record in the profile to be matched. This attribute should be specified in LHS of the '`users`' or '`hints`' file.

For example, if the '`users`' file contained:

```
DEFAULT Suffix = ".ppp", Auth-Type = System, Strip-User-Name = Yes
        Service-Type = Framed-User,
                Framed-Protocol = PPP
```

then usernames '`gray.ppp`' and '`yoda.ppp`' would match this record, whereas '`gray`' and '`yoda`' would not.

Both `Prefix` and `Suffix` attributes may be specified in a profile. In this case the record is matched only if the username contains both prefix and suffix specified.

Section 12.3.5 [Prefix], page 124 Section 12.3.11 [Strip-User-Name], page 126

### 12.3.7 Group

`ATTRIBUTE Group 1005 string`

| | |
|---|---|
| Users: | `L-` |
| Hints: | `L-` |
| Huntgroups: | `LR` |
| Additivity: | Append |
| Proxy propagated: | No |

### 12.3.8 Crypt-Password

`ATTRIBUTE Crypt-Password 1006 string`

| | |
|---|---|
| Users: | `L-` |
| Hints: | `--` |
| Huntgroups: | `--` |
| Additivity: | Append |
| Proxy propagated: | No |

This attribute is intended to be used in user's profile LHS. It specifies the MD5 hash of the user's password. When this attribute is present, Auth-Type = Crypt-Local is assumed. If both `Auth-Type` and `Crypt-Password` are present, the value of `Auth-Type` is ignored.

See Section 12.3.1 [Auth-Type], page 122.

### 12.3.9 Huntgroup-Name

`ATTRIBUTE Huntgroup-Name 221 string`

| | |
|---|---|
| Users: | `L-` |

| | |
|---|---|
| Hints: | -R |
| Huntgroups: | LR |
| Additivity: | Append |
| Proxy propagated: | No |

The `Huntgroup-Name` can be used either in LHS of the 'users' file record or in RHS of the 'huntgroups' file record.

When encountered in a LHS of a particular 'users' profile, this attribute indicates the huntgroup name to be matched. Radius looks up the corresponding record in the 'huntgroups' file. If such record is found, each A/V pair from its reply-list is compared against the corresponding pair from the request being processed. The request matches only if it contains all the attributes from the specified huntgroup, and their values satisfy the conditions listed in the huntgroup pairs.

For example, suppose that the authentication request contained the following attributes:

```
User-Name = "john",
Password = "guess",
NAS-IP-Address = 10.11.11.1,
NAS-Port-Id = 24
```

Let us further suppose that the 'users' file contains the following entry:

```
john    Huntgroup-Name = "users_group",
                Auth-Type = System
        Service-Type = Login
```

and, finally, 'huntgroups' contains the following entry:

```
users_group     NAS-IP-Address = 10.11.11.1
                NAS-Port-Id < 32
```

Then the authentication request would succeed since it contains `NAS-Port-Id` attribute and its value is less than 32.

See Section 4.7 [huntgroups file], page 48.

### 12.3.10 Simultaneous-Use

```
ATTRIBUTE  Simultaneous-Use  1034  integer
```

| | |
|---|---|
| Users: | L- |
| Hints: | -R |
| Huntgroups: | -R |
| Additivity: | Append |
| Proxy propagated: | No |

This attribute specifies the maximum number of simultaneous logins a given user is permitted to have. When the user is logged in this number of times any surplus attempts to log in are rejected.

See Section 5.9 [Checking Simultaneous Logins], page 66.

### 12.3.11 Strip-User-Name

```
ATTRIBUTE  Strip-User-Name  1035  integer
```

| | |
|---|---|
| Users: | `LR` |
| Hints: | `LR` |
| Huntgroups: | `-R` |
| Additivity: | Append |
| Proxy propagated: | No |

```
VALUE      Strip-User-Name   No                  0
VALUE      Strip-User-Name   Yes                 1
```

The value of `Strip-User-Name` indicates whether Radius should strip any prefixes/suffixes specified in the user's profile from the user name. When set to `Yes` the usernames would be logged and accounted without any prefixes/suffixes.

A user may have several usernames for different kind of services. In this case differentiating the usernames by their prefixes and stripping them off before accounting would help keeping accounting records consistent.

For example, let's suppose the '`users`' file contains:

```
DEFAULT Suffix = ".ppp",
                Strip-User-Name = Yes,
                Auth-Type = SQL
        Service-Type = Framed-User,
                Framed-Protocol = PPP

DEFAULT Suffix = ".slip",
                Strip-User-Name = Yes,
                Auth-Type = SQL
        Service-Type = Framed-User,
                Framed-Protocol = SLIP
```

Now, user '`johns`' having a valid account in SQL database logs in as '`johns.ppp`'. He then is provided the PPP service, and his PPP session is accounted under username '`johns`'. Later on, he logs in as '`johns.slip`'. In this case he is provided the SLIP service and again his session is accounted under his real username '`johns`'.

### 12.3.12 Fall-Through

```
ATTRIBUTE  Fall-Through  1036  integer
```

| | |
|---|---|
| Users: | `LR` |
| Hints: | `LR` |
| Huntgroups: | `--` |
| Additivity: | Append |
| Proxy propagated: | No |

```
VALUE      Fall-Through      No                  0
VALUE      Fall-Through      Yes                 1
```

The `Fall-Through` attribute should be used in reply-list. If its value is set to `Yes` in a particular record, it indicates to Radius that it should continue looking up another records even when this record matches the request. It can be used to provide default values for several profiles.

Consider the following example. Let's suppose the '`users`' file contains the following:

```
johns   Auth-Type = SQL
                Framed-IP-Address = 11.10.10.251,
                Fall-Through = Yes

smith   Auth-Type = SQL
                Framed-IP-Address = 11.10.10.252,
                Fall-Through = Yes

DEFAULT NAS-IP-Address = 11.10.10.1
        Service-Type = Framed-User,
                Framed-Protocol = PPP
```

Then after successful matching of a particular user's record, the matching will continue until it finds the DEFAULT entry, which would add its RHS to the reply pairs for this request. The effect is, that if user 'johns' authenticates successfully it gets the following reply pairs:

```
Service-Type = Framed-User,
Framed-Protocol = PPP,
Framed-IP-Address = 11.10.10.251
```

whereas user `smith` gets

```
Service-Type = Framed-User,
Framed-Protocol = PPP,
Framed-IP-Address = 11.10.10.252
```

*Please note* that the attribute `Fall-Through` itself is never returned to the NAS.

### 12.3.13 Add-Port-To-IP-Address

```
ATTRIBUTE  Add-Port-To-IP-Address  1037  integer
```

| | |
|---|---|
| Users: | -R |
| Hints: | -R |
| Huntgroups: | -- |
| Additivity: | Replace |
| Proxy propagated: | No |

```
VALUE      Add-Port-To-IP-Address  No                   0
VALUE      Add-Port-To-IP-Address  Yes                  1
```

If this attribute is present in the RHS and has the value of `Yes`, then the value of `NAS-Port-Id` attribute from the authentication request will be added to the value of `Framed-IP-Address` attribute from the RHS, and resulting value will be returned in `Framed-IP-Address` attribute to the NAS.

This provides the simplest form of organizing IP address pools.

This attribute is implicitly added to the RHS when the value of a `Framed-IP-Address` attribute ends with '+' sign. For example the following:

```
Framed-IP-Address = 10.10.0.1+
```

is equivalent to

```
Framed-IP-Address = 10.10.0.1,
Add-Port-To-IP-Address = Yes
```

### 12.3.14 Exec-Program

```
ATTRIBUTE  Exec-Program  1038  string
```

| | |
|---|---|
| Users: | `-R` |
| Hints: | `--` |
| Huntgroups: | `--` |
| Additivity: | Replace |
| Proxy propagated: | No |

When present in RHS, the `Exec-Program` attribute specifies the full pathname and arguments for the program to be executed when the entry matches.

The command line can reference any attributes from both check and reply pairlists using attribute macros (see Section 4.14 [Macro Substitution], page 60).

Before the execution of the program `radiusd` switches to uid and gid of user `daemon`, group `daemon`. You can override these defaults by setting variables `exec-program-user` and `exec-program-group` in configuration file to proper values Section 4.1.1 [The option statement], page 31.

The daemon does not wait for the process to terminate.

## Example

Suppose the '`users`' file contains the following entry:

```
DEFAULT Auth-Type = System,
                Simultaneous-Use = 1
        Exec-Program = "/usr/local/sbin/logauth \
                       %C{User-Name} \
                       %C{Calling-Station-Id}"
```

Then, upon successful matching, the program '`/usr/local/sbin/logauth`' will be executed. It will get as its arguments the values of `User-Name` and `Calling-Station-Id` attributes from the request pairs.

### 12.3.15 Exec-Program-Wait

```
ATTRIBUTE  Exec-Program-Wait  1039  string
```

| | |
|---|---|
| Users: | `-R` |
| Hints: | `--` |
| Huntgroups: | `--` |
| Additivity: | Replace |
| Proxy propagated: | No |

When present in RHS, the `Exec-Program-Wait` attribute specifies the full pathname and arguments for the program to be executed when the entry matches.

The command line can reference any attributes from both check and reply pairlists using attribute macros Section 4.14 [Macro Substitution], page 60.

Before the execution of the program `radiusd` switches to uid and gid of user `daemon`, group `daemon`. You can override these defaults by setting variable `exec-program-user` in configuration file to a proper value. Section 4.1.1 [The option statement], page 31.

The daemon will wait until the program terminates. The return value of its execution determines whether the entry matches. If the program exits with a non-zero code then the match fails. If it exits with a zero code, the match succeeds. In this case the standard output of the program is read and parsed as if it was a pairlist. The attributes thus obtained are added to the entry's reply attributes.

### Example

Suppose the 'users' file contains the following entry:

```
DEFAULT Auth-Type = System,
                Simultaneous-Use = 1
        Exec-Program-Wait = "/usr/local/sbin/telauth \
                            %C{User-Name} \
                            %C{Calling-Station-Id}"
```

Then, upon successful matching, the program '/usr/local/sbin/telauth' will be executed. It will get as its arguments the values of User-Name and Calling-Station-Id attributes from the request pairs.

The '/usr/local/sbin/telauth' can, for example, contain the following:

```
#! /bin/sh

DB=/var/db/userlist

if grep "$1:$2" $DB; then
        echo "Service-Type = Login,"
        echo "Session-Timeout = 1200"
        exit 0
else
        echo "Reply-Message = \"You are not authorized to log in\""
        exit 1
fi
```

It is assumed that '/var/db/userlist' contains a list of username:caller-id pairs for those users that are authorized to use login service.

## 12.3.16 Acct-Ext-Program

```
ATTRIBUTE  Acct-Ext-Program  2008  string
```

| | |
|---|---|
| Users: | -- |
| Hints: | -R |
| Huntgroups: | -- |
| Additivity: | Replace |
| Proxy propagated: | N/A |

The Acct-Ext-Program attribute can be used in RHS of an 'raddb/hints' entry to specify the full path and attributes of an external program to be executed for each accounting request.

The command line can reference any attributes from both check and reply pairlists using attribute macros (see Section 4.14 [Macro Substitution], page 60).

Before the execution of the program `radiusd` switches to uid and gid of user `daemon`, group `daemon`. You can override these defaults by setting variables `exec-program-user` and `exec-program-group` in configuration file to proper values Section 4.1.1 [The option statement], page 31.

The accounting program must exit with status 0 to indicate a successive accounting.

### 12.3.17 Hint

```
ATTRIBUTE  Hint  1040  string
```

| | |
|---|---|
| Users: | `L-` |
| Hints: | `-R` |
| Huntgroups: | `-R` |
| Additivity: | Append |
| Proxy propagated: | No |

Use `Hint` attribute to specify additional matching criterium depending on the hint (see Section 4.6 [hints file], page 48).

Let the '`hints`' contain:

```
DEFAULT         Prefix = "S", Strip-User-Name = No      Hint = "SLIP"
```

and the '`users`' file contain:

```
DEFAULT Hint = "SLIP",
                NAS-IP-Address = 11.10.10.12,
                Auth-Type = System
        Service-Type = Framed-User,
                Framed-Protocol = SLIP
```

Then any user having a valid system account and coming from NAS '`11.10.10.12`' will be provided SLIP service if his username starts with '`S`'.

### 12.3.18 Pam-Auth

```
ATTRIBUTE  Pam-Auth  1041  string
```

| | |
|---|---|
| Users: | `L-` |
| Hints: | `-R` |
| Huntgroups: | `-R` |
| Additivity: | Append |
| Proxy propagated: | No |

The `Pam-Auth` attribute can be used in conjunction with

```
Auth-Type = Pam
```

to supply the PAM service name instead of the default '`radius`'. It is ignored if `Auth-Type` attribute is not set to `Pam`.

### 12.3.19 Login-Time

```
ATTRIBUTE  Login-Time  1042  string
```

| | |
|---|---|
| Users: | `L-` |
| Hints: | `--` |

| | |
|---|---|
| Huntgroups: | `--` |
| Additivity: | Append |
| Proxy propagated: | No |

The `Login-Time` attribute specifies the time range when the user is allowed to log in. The attribute should be specified in LHS.

Format of the `Login-Time` string is the same as that of UUCP time ranges. The following description of time range format is adopted from the documentation for Taylor UUCP package:

A time string may be a list of simple time strings separated with a vertical bar '`|`' or a comma '`,`'.

Each simple time string must begin either with a day of week abbreviation (one of: '`Su`', '`Mo`', '`Tu`', '`We`', '`Th`', '`Fr`', or '`Sa`'), or '`Wk`' for any day between Monday and Friday inclusive, or '`Any`' or '`Al`' for any day.

Following the day may be a range of hours separated with a hyphen using 24 hour time. The range of hours may cross 0; for example '`2300-0700`' means any time except 7 AM to 11 PM. If no time is given, calls may be made at any time on the specified day(s).

The time string may also be the single word '`Never`', which does not match any time.

Here are a few sample time strings with an explanation of what they mean.

'`Wk2305-0855,Sa,Su2305-1655`'
: This means weekdays before 8:55 AM or after 11:05 PM, any time Saturday, or Sunday before 4:55 PM or after 11:05 PM. These are approximately the times during which night rates apply to phone calls in the U.S.A. Note that this time string uses, for example, '`2305`' rather than '`2300`'; this will ensure a cheap rate phone call even if the computer clock is running up to five minutes ahead of the real time.

'`Wk0905-2255,Su1705-2255`'
: This means weekdays from 9:05 AM to 10:55 PM, or Sunday from 5:05 PM to 10:55 PM. This is approximately the opposite of the previous example.

'`Any`'
: This means any day. Since no time is specified, it means any time on any day.

### 12.3.20 Replace-User-Name

```
ATTRIBUTE  Replace-User-Name  2001  string
```

| | |
|---|---|
| Users: | `LR` |
| Hints: | `LR` |
| Huntgroups: | `--` |
| Additivity: | Append |
| Proxy propagated: | No |

```
VALUE      Replace-User-Name  No                 0
VALUE      Replace-User-Name  Yes                1
```

Use this attribute to modify username from the incoming packet. The `Replace-User-Name` can reference any attributes from both LHS and RHS pairlists using attribute macros Section 4.14 [Macro Substitution], page 60.

For example the following 'users' entry

```
guest   NAS-IP-Address = 11.10.10.11,
                Calling-Station-Id != ""
                Auth-Type = Accept
        Replace-User-Name = "guest#%C{Calling-Station-Id}",
                Service-Type = Framed-User,
                Framed-Protocol = PPP
```

Allows usage of PPP service for username guest, coming from NAS '11.10.10.11' with non-empty `Calling-Station-Id` attribute. The string consisting of '#' character followed by `Calling-Station-Id` value is appended to the username.

### 12.3.21 Rewrite-Function

```
ATTRIBUTE  Rewrite-Function  2004  string
```

| | |
|---|---|
| Users: | LR |
| Hints: | LR |
| Huntgroups: | LR |
| Additivity: | Append |
| Proxy propagated: | No |

The `Rewrite-Function` attribute specifies the name of the rewriting function to be applied to the request. The attribute may be specified in either pairlist in the entries of 'hints' or 'huntgroups' configuration files.

The corresponding function should be defined in 'rewrite' as

`integer` *name*`()`

i.e. it should return integer value and should not take any arguments.

See Section 4.12 [Packet rewriting rules], page 57. See Section 4.6 [hints file], page 48. See Section 4.7 [huntgroups file], page 48.

### 12.3.22 Match-Profile

```
ATTRIBUTE  Match-Profile  2004  string
```

| | |
|---|---|
| Users: | LR |
| Hints: | -R |
| Huntgroups: | -R |
| Additivity: | Append |
| Proxy propagated: | No |

The Match-Profile attribute can be used in LHS and RHS lists of a user profile. Its value is the name of another user's profile (target profile). When Match-Profile is used in the LHS, the incoming packet will match this profile only if it matches the target profile. In this case the reply pairs will be formed concatenating the RHS lists from both profiles. When used in RHS, this attribute causes the reply pairs from the target profile to be appended to the reply from the current profile if the target profile matches the incoming request.

For example:

```
IPPOOL  NAS-IP-Address = 10.10.10.1
                Framed-Protocol = PPP, Framed-IP-Address = "10.10.10.2"
```

```
IPPOOL  NAS-IP-Address = 10.10.11.1
                Framed-Protocol = PPP, Framed-IP-Address = "10.10.11.2"

guest   Auth-Type = SQL
                Service-Type = Framed-User,
        Match-Profile = IPPOOL
```

In this example, when user "guest" comes from NAS `10.10.10.1` he is assigned IP address `10.10.10.2`, otherwise if he is coming from NAS `10.10.11.1` he is assigned IP address `10.10.11.2`.

### 12.3.23 Scheme-Procedure

```
ATTRIBUTE  Scheme-Procedure  2009  string
```

| | |
|---|---|
| Users: | `-R` |
| Hints: | `--` |
| Huntgroups: | `--` |
| Additivity: | Append |
| Proxy propagated: | N/A |

The `Scheme-Procedure` attribute is used to set the name of Scheme authentication procedure. See Section 9.2.2 [Authentication with Scheme], page 86, for the information about how to write Scheme authentication procedures.

### 12.3.24 Scheme-Acct-Procedure

```
ATTRIBUTE  Scheme-Acct-Procedure  2010  string
```

| | |
|---|---|
| Users: | `--` |
| Hints: | `-R` |
| Huntgroups: | `--` |
| Additivity: | Replace |
| Proxy propagated: | N/A |

The `Scheme-Acct-Procedure` attribute is used to set the name of Scheme accounting procedure. See Section 9.2.3 [Accounting with Scheme], page 88, for the information about how to write Scheme accounting procedures.

### 12.3.25 Log-Mode-Mask

```
ATTRIBUTE  Log-Mode-Mask  2007  integer
```

| | |
|---|---|
| Users: | `L-` |
| Hints: | `-R` |
| Huntgroups: | `-R` |
| Additivity: | Append |
| Proxy propagated: | N/A |

```
VALUE Log-Mode-Mask Log-Auth 1
VALUE Log-Mode-Mask Log-Auth-Pass 2
VALUE Log-Mode-Mask Log-Failed-Pass 4
VALUE Log-Mode-Mask Log-Pass 6
```

```
VALUE Log-Mode-Mask Log-All 7
```

`Log-Mode-Mask` is used to control the verbosity of authentication log messages for given user or class of users. The meaning of its values is:

`Log-Auth` Do not log successful authentications.

`Log-Auth-Pass`
Do not show password with the log message from a successful authentication.

`Log-Failed-Pass`
Do not show failed password.

`Log-Pass` Do not show plaintext password, either failed or succeeded.

`Log-All` Do not log authentications at all.

Technical details: After authentication, the server collects all `Log-Mode-Mask` attributes from the incoming request and LHS of the user's entry. The values of these attributes OR'ed together form a mask which is applied via XOR operation to the current log mode. The value thus obtained is used as effective log mode.

### 12.3.26 Acct-Type

```
ATTRIBUTE  Acct-Type  2003  integer
```

| | |
|---|---|
| Users: | `L-` |
| Hints: | `-R` |
| Huntgroups: | `-R` |
| Additivity: | Append |
| Proxy propagated: | N/A |

```
VALUE Acct-Type None 0
VALUE Acct-Type System 1
VALUE Acct-Type Detail 2
VALUE Acct-Type SQL 3
```

The `Acct-Type` allows to control which accounting methods must be used for a given user or a group of users. In the absense of this attribute, all currently enabled accounting types are used. See Chapter 6 [Accounting], page 67, for more information about accounting types.

# 13 Reporting Bugs

It is possible you will encounter a bug in one of Radius programs. If this happens we would like to hear about it. As the purpose of bug reporting is to improve software, please be sure to include maximum information when reporting a bug. The information needed is:

- Conditions under which the bug appears.
- Version of the package you are using.
- Compilation options used when configuring the package.
- If the bug is found in `radiusd` daemon, run '`radiusd -v`' and include the output it produces.
- Contents of radius configuration directory (/usr/local/etc/raddb or whatever you have set it to while configuring).
- Log messages produced.

Send your report to `bug-gnu-radius@gnu.org`. Allow me a couple of days to answer.

# 14 Where to Get Info About GNU Radius

The two places to look for any news regarding GNU Radius are the Radius homepage at `http://www.gnu.org/software/radius`, and the Radius project page at `http://savannah.gnu.org/projects/radius`.

The following mailing lists are related to GNU Radius:

`info-gnu-radius@gnu.org`
: This list distributes announcements and progress reports on GNU Radius. This is a moderated list. Please, do not send bug reports or requests for help to this list, there exist special mailing lists for these purposes. To subscribe to the list, visit `http://mail.gnu.org/mailman/listinfo/info-gnu-radius`.

`help-gnu-radius@gnu.org`
: This list is the place for users and installers of GNU Radius to ask for help. The list is not moderated, but postings are allowed for list members only. To subscribe to the list, visit `http://mail.gnu.org/mailman/listinfo/help-gnu-radius`.

`bug-gnu-radius@gnu.org`
: This list distributes bug reports, bugfixes, and suggestions for improvements in Radius. User discussion of Radius bugs also occurs here. The list is not moderated, postings are allowed for anybody. To subscribe to the list, visit `http://mail.gnu.org/mailman/listinfo/bug-gnu-radius`.

# Program Index

# Keyword Index

## V

# Examples Index

# Index of Functions and Variables

# Attribute Index

## V

# Concept Index

## N



## P



## R



## S



## U



## W

# Short Contents

# Table of Contents