FreeRadius Integration with OES2 for use with Group Membership

Contents

Overview

So you want to setup FreeRadius with eDirectory support running on OES2 Linux, and you just want a simple setup for hardware or software that uses the RADIUS protocol based upon Group Membership. For example: "If member of Group: VPN-Users", then you're allowed access to the network. This may also work for OES11, I have not tried or verified this. I imagine it should work, although possibly the FreeRadius code in OES11 is updated and the configuration files may be a bit different or located in different directories than what is listed here.

Scenario

The environment contains two pieces of hardware. A Nortel Contivity VPN and Cisco RAS dial-in use RADIUS. We want to use FreeRadius on OES with eDirectory instead of Cisco TACACS with Active Directory on Windows. We do not need to use RADIUS with Wireless for things like laptops, etc. We strictly desire to base the authorization based upon Group Membership to a group called: "VPN-Users". If you're a member of the group and you present

a valid eDirectory userid and password you are allowed to access the VPN. If not, then no soup for you. Because of this, we don't really need to extend the Schema for the radius attributes on each user object, since we are not using those.

Example:
OES2 Servername: co-oes2
Server IP: 10.10.1.4

Installing FreeRadius 2.1.8

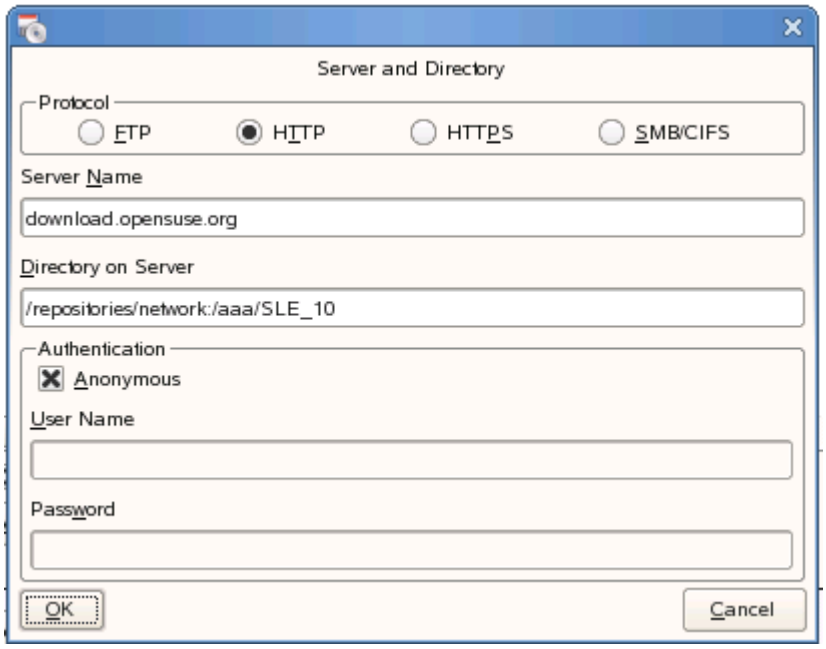
Use the Yast -> Software Management to verify that the FreeRadius packages are installed. If they are not installed, then you need to install them. Preferably on a fully patched system. I do believe that in OES11 the FreeRadius packages are updated which may change the config files used from what is in this documentation.

Here's the installation instructions and configuration we used for our OES2 environment (note: Some of the information below may not be applicable for OES11 anymore as I'm sure the version of FreeRadius has been updated in the SLES11/OES11 installations). This is just to give you a general idea of what to do. Note that in very old versions of the FreeRadius modules, you used a file called: "ldap" to hold the configuration. In the updated FreeRadius versions, that information is stored in the radiusd.conf file.

Novell shipped an ancient version of FreeRADIUS with OES2. This install obtains a more recent version.

First, make sure your OES2 server is fully patched.

Second, go into Yast -> Software -> Installation Source. Click Add:



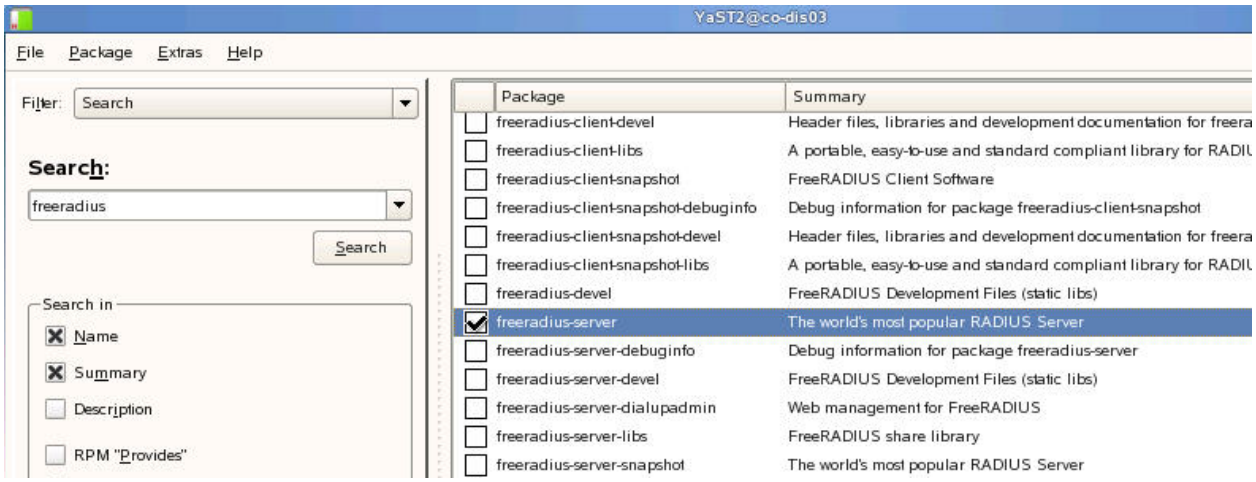
Click OK

Click Finish.

It may take a few minutes to pulldown the mirror catalogs.

Now select Software Management (Yast is still running).

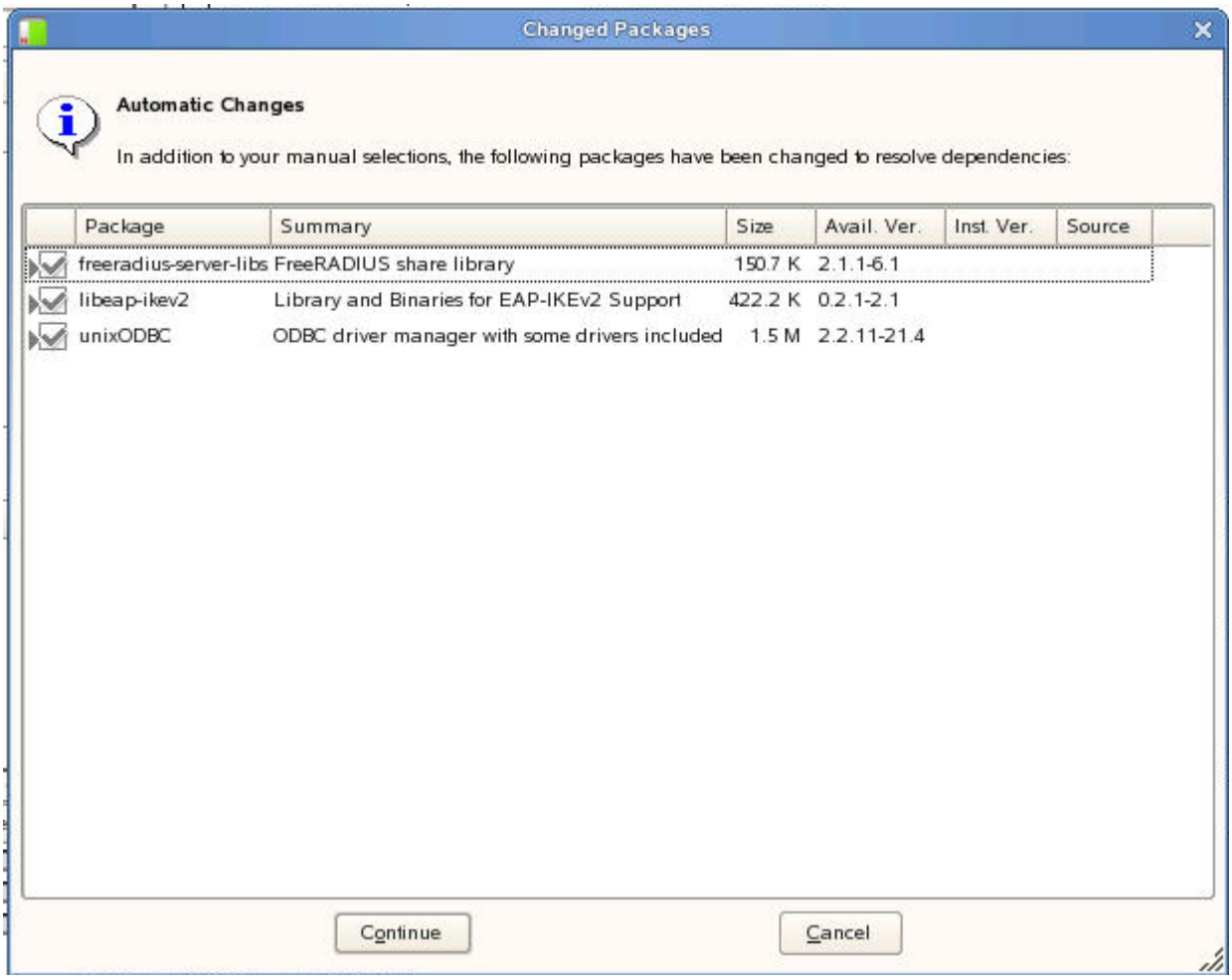
In the search field type: freeradius



Find the item:
freeradiusserver

Check the box next to it and click Accept.

You may get a pop-up:



Click Continue (you want the freeradius-server-libs)

You'll get a download screen. When it's done, you don't want to install any more packages.

Integrating FreeRADIUS with eDirectory

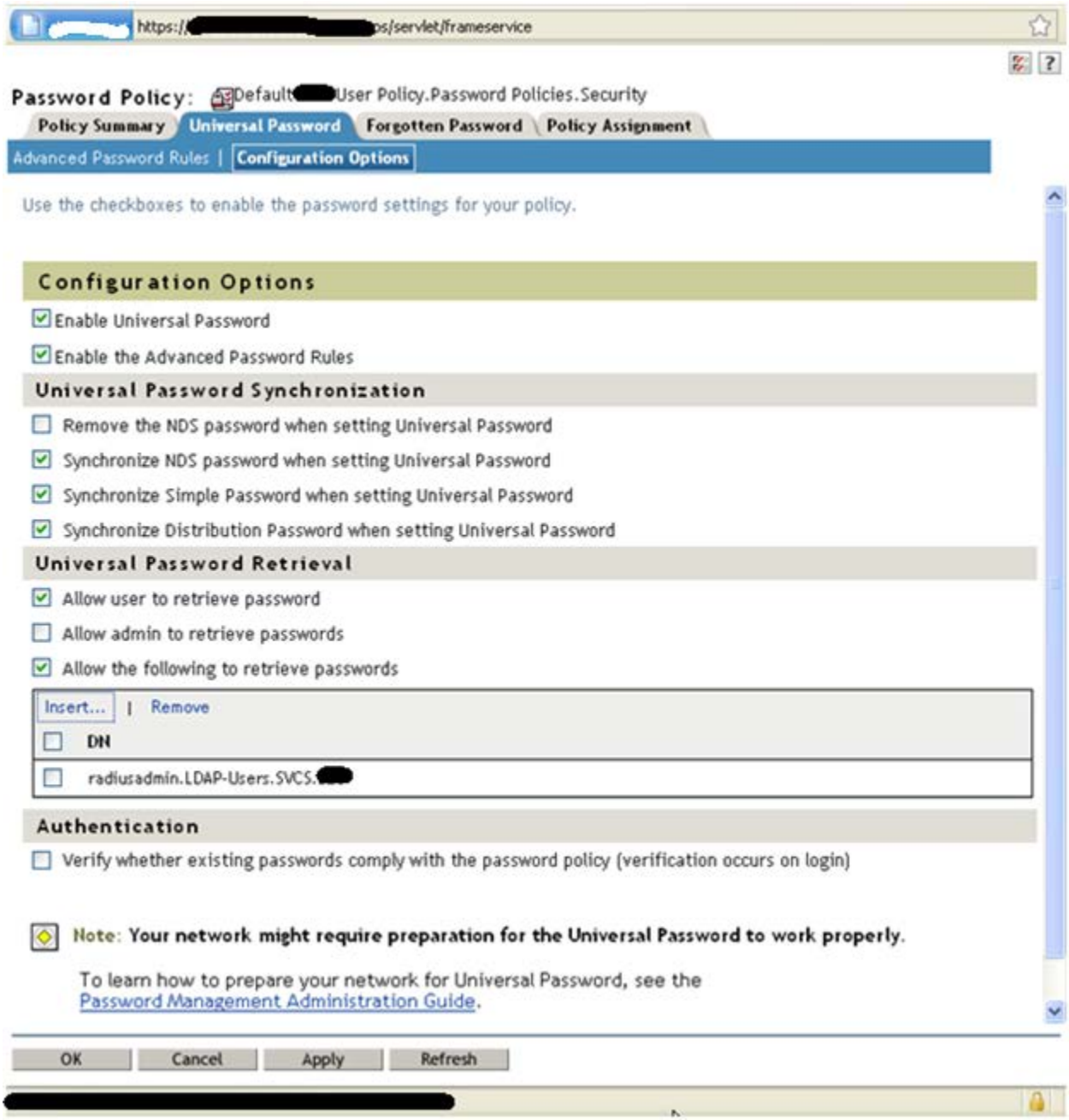
According to Novell's docs, we need to make sure that we have eDirectory, iManager, and Universal Password setup. We already have that in our environment, so we can skip that part of their docs. If you do not, then you will need to refer to Novell's documentation on how to setup Universal Password (eDirectory and iManager are probably a given on any OES environment).

FreeRADIUS Administrative User Account

Next, we need to create a freeradius admin user. For security purposes, we will NOT use the Admin account. I used consoleone to create a user: radiusadmin in the .ldap-users.svcs.abc container. You could also use iManager to create the user account.

Add this user to the "no expiry" password policy in iManager -> Passwords -> Password Policies. (You will want to create a specific password policy that never expires the passwords if you don't already have one). However, if your existing UPP (Universal Password Policy) does not expire passwords, you could also assign this to the user in question. But I like to create a specific UPP for "admin" or service accounts like this as it gives me more flexibility.

Now find your Default password policy (in the same section above) and edit it. We need to grant the radiusadmin user the ability to retrieve Universal Passwords on behalf of the user.

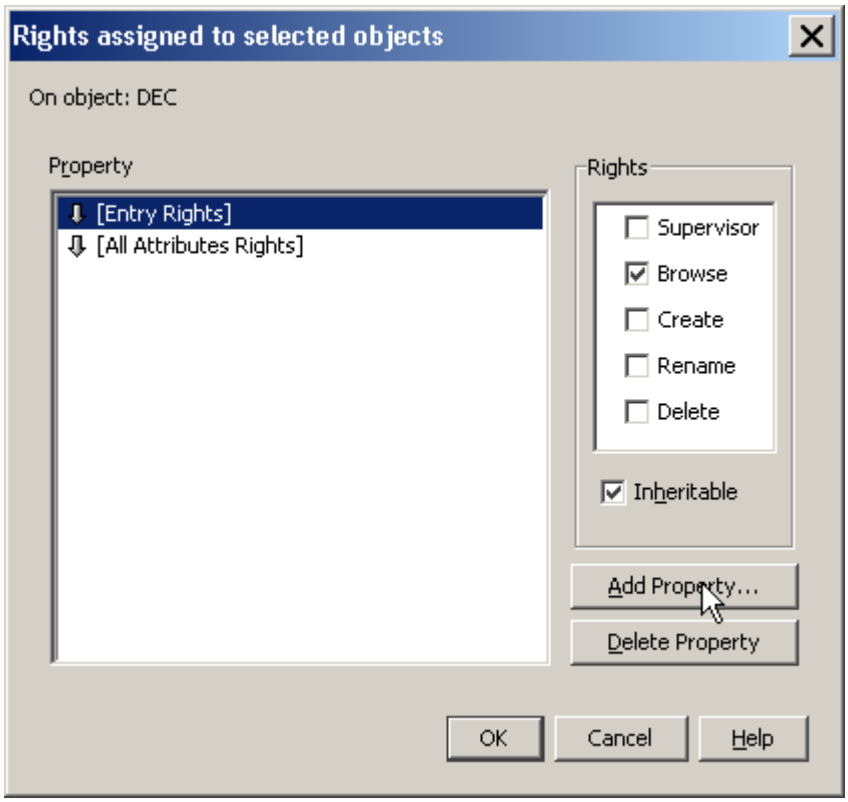


This is found in the Universal Password tab -> Configuration Options sub-section.

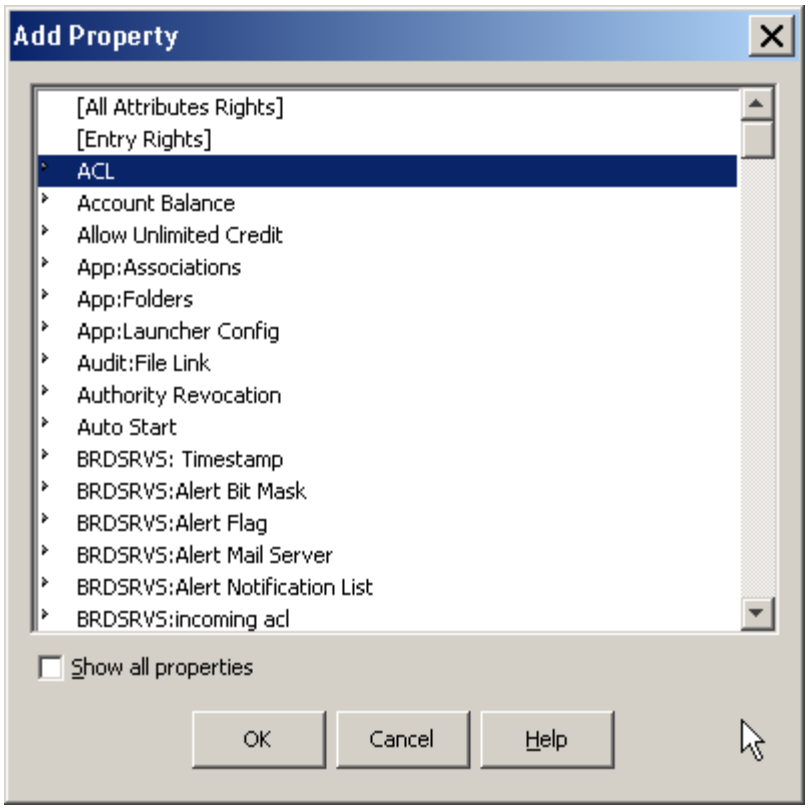
Use ConsoleOne (or iManager), find the O=whatever container, right-click and select Trustees of this object:

Click ADD Trustee

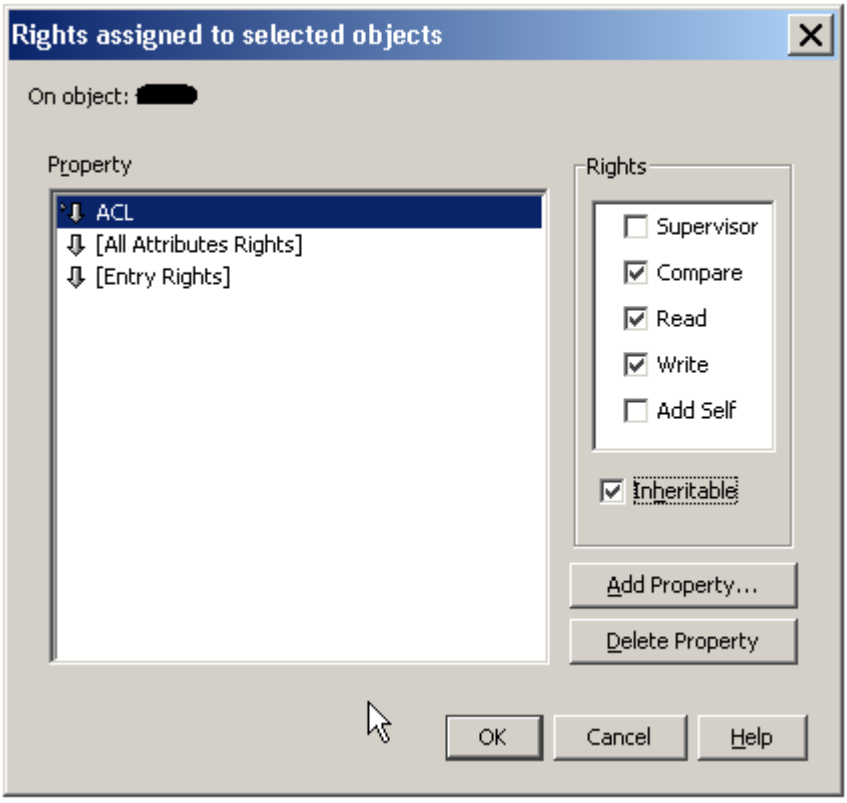
Browse for and find the: radiusadmin.ldap-users.svcs.abc account.



Click Add Property



Highlight the ACL property and click OK



I ADDED the “write” right and checked the Inheritable box and clicked OK. This is according to Novell’s docs.

Click OK again.

eDirectory Server Configuration

According to Novell’s docs, we need to obtain the DER certificate from eDirectory. For this case, we want the .b64 file

You can use this information to assist with exporting the CA eDirectory certificate (public key):
https://www.netiq.com/documentation/edir88/crtadmin88/data/a2ebop8.html#a2ppx57

Copy that file (I use WinSCP) into the radius server’s:
/etc/raddb/certs directory
I renamed the certificate to: rootder.b64

Files

There are three main configuration files that we need to configure for this purpose:

radiusd.conf
users
clients.conf

radiusd.conf

The main portions here are line 750 (see sample config file in the appendix) which starts with:

ldap {

Specifically we’ll want to adjust the lines:

```
ldap {
        server = "10.10.1.4"
        # identity = "cn=admin,o=My Org,c=UA"
        # password = mypass
        port=636

        identity = "cn=radiusadmin,ou=ldap-users,ou=svcs,o=abc"
        password = rkfKSTgm
        basedn = "o=abc"
        filter = "(uid=%{Stripped-User-Name:-%{User-Name}})"
```

Change the server line to whatever your OES Server IP is.
Enter the appropriate user account (we used a radiusadmin account) for the identity, along with the password. Set the basedn accordingly. This should be equal to your LDAP Search base (ie: Where in the eDirectory tree, do you want to start searching for things).

We set the filter so that it will find the username. Most of the time, a "userid" in eDirectory will match the "cn" value, and will also match the "uid" value. There are cases where really old user accounts created with certain versions of NWadmin32 or ConsoleOne did not set the uid to match, so you may wish to do some cleanup of your eDirectory environment beforehand.

We now go down to line 821 and want to adjust the group membership section:

```
        groupname_attribute = cn
        #groupmembership_filter = "(member=%{Ldap-UserDn})"
        groupmembership_filter = "(&(objectClass=GroupOfNames)(member=%{Ldap-UserDn}))"
        groupmembership_attribute = VPN-users
        timeout = 4
        timelimit = 3
        net_timeout = 1
        # compare_check_items = yes
        # do_xlat = yes
        # access_attr_used_for_allow = yes
        access_attr_used_for_allow = yes

        #
        #  By default, if the packet contains a User-Password,
        #  and no other module is configured to handle the
        #  authentication, the LDAP module sets itself to do
        #  LDAP bind for authentication.
        #
        #  You can disable this behavior by setting the following
        #  configuration entry to "no".
        #
        #  allowed values: {no, yes}
        set_auth_type = yes
        # set_auth_type = no
}
```

We are enabling the groupname attribute "cn" (ie, the name of the group), and we're going to query the group object for the membership list (the groupmembership_filter is an LDAP query).

We designated that we want to use this for allowing access. This is set via the "access_attr_used_for_allow" line.

Make sure to set the auth type to “yes”.

This is the end of the LDAP section in the radiusd.conf file

Continue down to about line 1766 which takes you to the instantiate section:

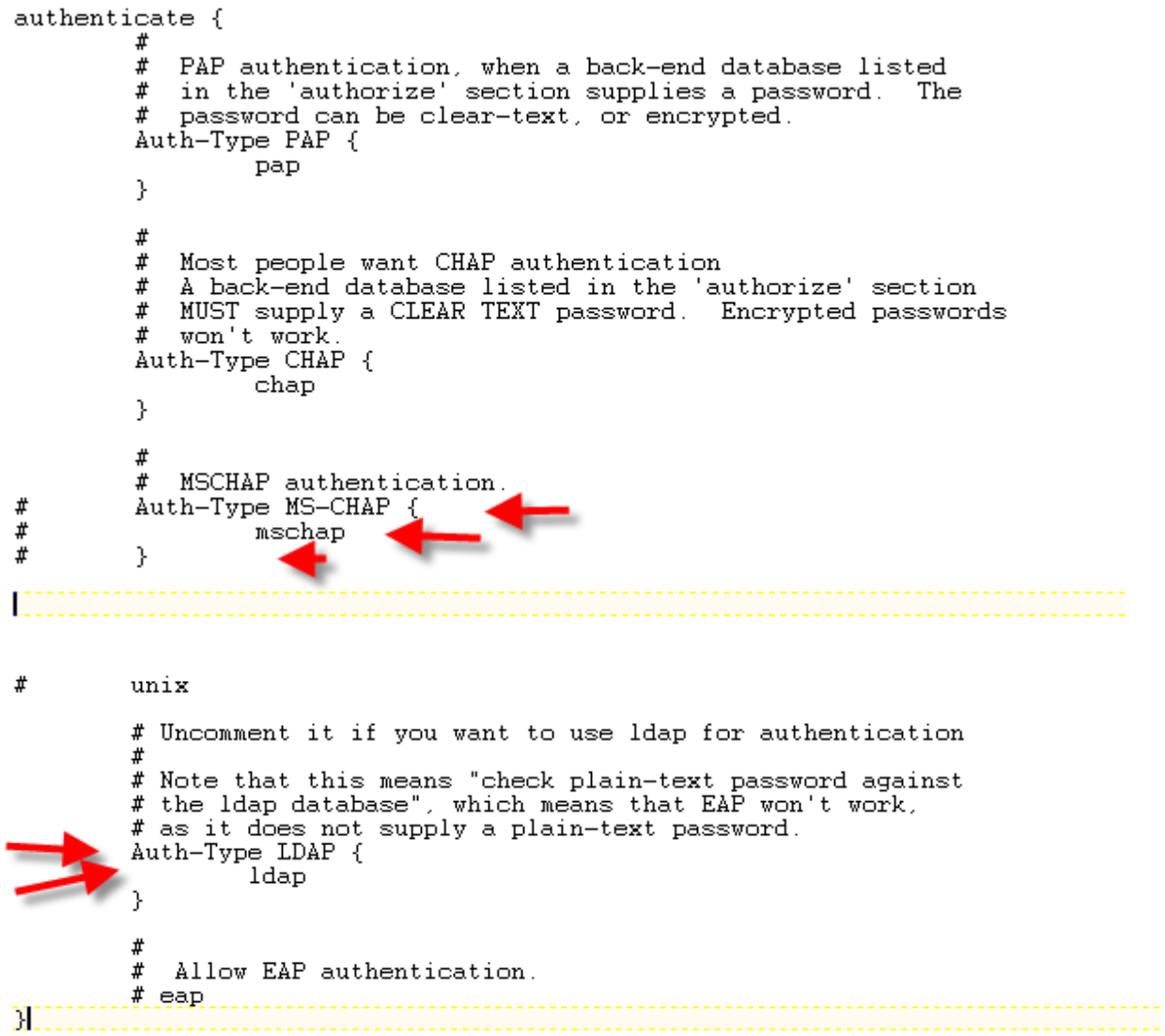
```
instantiate {
        #
        #  Allows the execution of external scripts.
        #  The entire command line (and output) must fit into 253 bytes.
        #
        #  e.g. Framed-Pool = `%{exec:/bin/echo foo}`
        exec

        #
        #  The expression module doesn't do authorization,
        #  authentication, or accounting.  It only does dynamic
        #  translation, of the form:
        #
        #       Session-Timeout = `%{expr:2 + 3}`
        #
        #  So the module needs to be instantiated, but CANNOT be
        #  listed in any other section.  See 'doc/rlm_expr' for
        #  more information.
        #
        expr

        #
        # We add the counter module here so that it registers
        # the check-name attribute before any module which sets
        # it
#       daily
        ldap
}
```

ADD the line in red for ldap

Now, continue down to line 1896 which is the authenticate section. Now, depending upon your methods used by your software or hardware appliances, this may vary for you. We disabled MSCHAP and enabled LDAP



```
authenticate {
        #
        #  PAP authentication, when a back-end database listed
        #  in the 'authorize' section supplies a password.  The
        #  password can be clear-text, or encrypted.
        Auth-Type PAP {
                pap
        }

        #
        #  Most people want CHAP authentication
        #  A back-end database listed in the 'authorize' section
        #  MUST supply a CLEAR TEXT password.  Encrypted passwords
        #  won't work.
        Auth-Type CHAP {
                chap
        }

        #
        #  MSCHAP authentication.
#       Auth-Type MS-CHAP {
#               mschap
#       }


#       unix

        # Uncomment it if you want to use ldap for authentication
        #
        # Note that this means "check plain-text password against
        # the ldap database", which means that EAP won't work,
        # as it does not supply a plain-text password.
        Auth-Type LDAP {
                ldap
        }

        #
        #  Allow EAP authentication.
        # eap
}
```

(we removed the comment lines to enable LDAP)

Go to the PostAuth section around line 2041:

```
#  Post-Authentication
#  Once we KNOW that the user has been authenticated, there are
#  additional steps we can take.
post-auth {
        #  Get an address from the IP Pool.
#       main_pool
#       sqlippool

        #
        #  If you want to have a log of authentication replies,
        #  un-comment the following line, and the 'detail reply_log'
        #  section, above.
#       reply_log

        #
        #  After authenticating the user, do another SQL query.
        #
        #  See "Authentication Logging Queries" in sql.conf
#       sql

        #
        #  Instead of sending the query to the SQL server,
        #  write it into a log file.
        #
#       sql_log

        #
        #  Un-comment the following if you have set
        #  'edir_account_policy_check = yes' in the ldap module sub-section of
        #  the 'modules' section.
        #
        ldap
        #
        #  Access-Reject packets are sent through the REJECT sub-section of the
        #  post-auth section.
        #  Uncomment the following and set the module name to the ldap instance
        #  name if you have set 'edir_account_policy_check = yes' in the ldap
        #  module sub-section of the 'modules' section.
        #
        Post-Auth-Type REJECT {
                ldap
        }
```

Uncomment out the ldap line

This concludes the radiusd.conf file

Users

The Users file I had some trouble with. Not sure if this is the proper way to edit it, but it's how I got it to work.

```
#
# Deny access for a specific user.  Note that this entry MUST
# be before any other 'Auth-Type' attribute which results in the user
# being authenticated.
#
# Note that there is NO 'Fall-Through' attribute, so the user will not
# be given any additional resources.
#
#lameuser       Auth-Type := Reject
#               Reply-Message = "Your account has been disabled."

#
# Deny access for a group of users.
#
# Note that there is NO 'Fall-Through' attribute, so the user will not
# be given any additional resources.
#
#DEFAULT        Group == "disabled", Auth-Type := Reject
#               Reply-Message = "Your account has been disabled."
#

DEFAULT Ldap-Group == "cn=VPN-Users,o=abc", Auth-Type := LDAP
        Fall-Through = 0

DEFAULT Auth-Type = REJECT
        Fall-Through = 1
#

#
# This is a complete entry for "steve".  Note that there is no Fall-Through
```

Here, you want the name of the group to be in the FQDN LDAP format, and it must match the "cn" of the group that was in the radiusd.conf file.

In our case, we have an eDir group called: cn=VPN-Users,o=abc that is the FQDN (in LDAP format) that we insert into the Users file. In the Radiusd.conf file we simply specified the CN: VPN-Users

The above entries are at line 71 in the Users file. We had to add the two "DEFAULT" lines as shown above.

Clients.conf

The Clients.conf is where you list the IP's of the devices that will be connecting to the Radius server. In my case, I added my PC (so that I can run the radping software to test), along with our two Datacom devices: An old Nortel Contivity VPN, and a Cisco RAS router/switch.

```
        #
#       login           = !root
#       password        = someadminpas
}

client 10.10.114.4 {
        secret          = 4radius2use
        shortname       = Contivity
}

client 10.10.116.11 {
        secret          = 4radius2use
        shortname       = RAS
}

client 10.10.217.217 {
        secret          = 4radius2use
        shortname       = Kevin PC
}

#client some.host.org {
#       secret          = testing123
#       shortname       = localhost
#}
```

You also add the secret and a shortname (shortname can be anything, I believe), but the secret must "match" whatever you input into the other end. (ie, in the Cisco/Nortel software, you enter the same secret name there as you would into this config file).

Appendix

Sample radiusd.conf

```
##
## radiusd.conf         -- FreeRADIUS server configuration file.
##
##      http://www.freeradius.org/
##      $Id: radiusd.conf.in,v 1.188.2.4.2.18 2007/07/16 10:53:13 pnixon Exp $
##

#       The location of other config files and
#       logfiles are declared in this file
#
```

```
#       Also general configuration for modules can be done
#       in this file, it is exported through the API to
#       modules that ask for it.
#
#       The configuration variables defined here are of the form ${foo}
#       They are local to this file, and do not change from request to
#       request.
#
#       The per-request variables are of the form %{Attribute-Name}, and
#       are taken from the values of the attribute in the incoming
#       request.  See 'doc/variables.txt' for more information.

prefix = /usr
exec_prefix = ${prefix}
sysconfdir = /etc
localstatedir = /var
sbindir = ${exec_prefix}/sbin
logdir = ${localstatedir}/log/radius
raddbdir = ${sysconfdir}/raddb
radacctdir = ${logdir}/radacct

#  Location of config and logfiles.
confdir = ${raddbdir}
run_dir = ${localstatedir}/run/radiusd

#
#  The logging messages for the server are appended to the
#  tail of this file.
#
log_file = ${logdir}/radius.log

#
# libdir: Where to find the rlm_* modules.
#
#   This should be automatically set at configuration time.
#
#   If the server builds and installs, but fails at execution time
#   with an 'undefined symbol' error, then you can use the libdir
#   directive to work around the problem.
#
#   The cause is usually that a library has been installed on your
#   system in a place where the dynamic linker CANNOT find it.  When
#   executing as root (or another user), your personal environment MAY
#   be set up to allow the dynamic linker to find the library.  When
#   executing as a daemon, FreeRADIUS MAY NOT have the same
#   personalized configuration.
```

```
#
#  To work around the problem, find out which library contains that symbol,
#  and add the directory containing that library to the end of 'libdir',
#  with a colon separating the directory names.  NO spaces are allowed.
#
#  e.g. libdir = /usr/local/lib:/opt/package/lib
#
#  You can also try setting the LD_LIBRARY_PATH environment variable
#  in a script which starts the server.
#
#  If that does not work, then you can re-configure and re-build the
#  server to NOT use shared libraries, via:
#
#        ./configure --disable-shared
#        make
#        make install
#
libdir = /usr/lib/freeradius

#  pidfile: Where to place the PID of the RADIUS server.
#
#  The server may be signalled while it's running by using this
#  file.
#
#  This file is written when ONLY running in daemon mode.
#
#  e.g.:  kill -HUP `cat /var/run/radiusd/radiusd.pid`
#
pidfile = ${run_dir}/radiusd.pid


# user/group: The name (or #number) of the user/group to run radiusd as.
#
#  If these are commented out, the server will run as the user/group
#  that started it.  In order to change to a different user/group, you
#  MUST be root ( or have root privleges ) to start the server.
#
#  We STRONGLY recommend that you run the server with as few permissions
#  as possible.  That is, if you're not using shadow passwords, the
#  user and group items below should be set to 'nobody'.
#
#   On SCO (ODT 3) use "user = nouser" and "group = nogroup".
#
#  NOTE that some kernels refuse to setgid(group) when the value of
#  (unsigned)group is above 60000; don't use group nobody on these systems!
#
```

```
#  On systems with shadow passwords, you might have to set 'group = shadow'
#  for the server to be able to read the shadow password file.  If you can
#  authenticate users while in debug mode, but not in daemon mode, it may be
#  that the debugging mode server is running as a user that can read the
#  shadow info, and the user listed below can not.
#
user = radiusd
group = radiusd

#  max_request_time: The maximum time (in seconds) to handle a request.
#
#  Requests which take more time than this to process may be killed, and
#  a REJECT message is returned.
#
#  WARNING: If you notice that requests take a long time to be handled,
#  then this MAY INDICATE a bug in the server, in one of the modules
#  used to handle a request, OR in your local configuration.
#
#  This problem is most often seen when using an SQL database.  If it takes
#  more than a second or two to receive an answer from the SQL database,
#  then it probably means that you haven't indexed the database.  See your
#  SQL server documentation for more information.
#
#  Useful range of values: 5 to 120
#
max_request_time = 30

#  delete_blocked_requests: If the request takes MORE THAN 'max_request_time'
#  to be handled, then maybe the server should delete it.
#
#  If you're running in threaded, or thread pool mode, this setting
#  should probably be 'no'.  Setting it to 'yes' when using a threaded
#  server MAY cause the server to crash!
#
delete_blocked_requests = no

#  cleanup_delay: The time to wait (in seconds) before cleaning up
#  a reply which was sent to the NAS.
#
#  The RADIUS request is normally cached internally for a short period
#  of time, after the reply is sent to the NAS.  The reply packet may be
#  lost in the network, and the NAS will not see it.  The NAS will then
#  re-send the request, and the server will respond quickly with the
#  cached reply.
#
#  If this value is set too low, then duplicate requests from the NAS
```

```
#  MAY NOT be detected, and will instead be handled as seperate requests.
#
#  If this value is set too high, then the server will cache too many
#  requests, and some new requests may get blocked.  (See 'max_requests'.)
#
#  Useful range of values: 2 to 10
#
cleanup_delay = 5

#  max_requests: The maximum number of requests which the server keeps
#  track of.  This should be 256 multiplied by the number of clients.
#  e.g. With 4 clients, this number should be 1024.
#
#  If this number is too low, then when the server becomes busy,
#  it will not respond to any new requests, until the 'cleanup_delay'
#  time has passed, and it has removed the old requests.
#
#  If this number is set too high, then the server will use a bit more
#  memory for no real benefit.
#
#  If you aren't sure what it should be set to, it's better to set it
#  too high than too low.  Setting it to 1000 per client is probably
#  the highest it should be.
#
#  Useful range of values: 256 to infinity
#
max_requests = 1024

#  bind_address:  Make the server listen on a particular IP address, and
#  send replies out from that address.  This directive is most useful
#  for machines with multiple IP addresses on one interface.
#
#  It can either contain "*", or an IP address, or a fully qualified
#  Internet domain name.  The default is "*"
#
#  As of 1.0, you can also use the "listen" directive.  See below for
#  more information.
#
bind_address = 10.10.251.23

#  port: Allows you to bind FreeRADIUS to a specific port.
#
#  The default port that most NAS boxes use is 1645, which is historical.
#  RFC 2138 defines 1812 to be the new port.  Many new servers and
#  NAS boxes use 1812, which can create interoperability problems.
#
```

```
#  The port is defined here to be 0 so that the server will pick up
#  the machine's local configuration for the radius port, as defined
#  in /etc/services.
#
#  If you want to use the default RADIUS port as defined on your server,
#  (usually through 'grep radius /etc/services') set this to 0 (zero).
#
#  A port given on the command-line via '-p' over-rides this one.
#
#  As of 1.0, you can also use the "listen" directive.  See below for
#  more information.
#
port = 0

#
#  By default, the server uses "bind_address" to listen to all IP's
#  on a machine, or just one IP.  The "port" configuration is used
#  to select the authentication port used when listening on those
#  addresses.
#
#  If you want the server to listen on additional addresses, you can
#  use the "listen" section.  A sample section (commented out) is included
#  below.  This "listen" section duplicates the functionality of the
#  "bind_address" and "port" configuration entries, but it only listens
#  for authentication packets.
#
#  If you comment out the "bind_address" and "port" configuration entries,
#  then it becomes possible to make the server accept only accounting,
#  or authentication packets.  Previously, it always listened for both
#  types of packets, and it was impossible to make it listen for only
#  one type of packet.
#
#listen {
        #  IP address on which to listen.
        #  Allowed values are:
        #        dotted quad (1.2.3.4)
        #      hostname    (radius.example.com)
        #      wildcard    (*)
#       ipaddr = *

        #  Port on which to listen.
        #  Allowed values are:
        #        integer port number (1812)
        #        0 means "use /etc/services for the proper port"
#       port = 0
```

```
        #  Type of packets to listen for.
        #  Allowed values are:
        #       auth    listen for authentication packets
        #       acct    listen for accounting packets
        #
#       type = auth
#}

listen {
        ipaddr = 10.10.251.23
        port = 1646
        type = acct
}


#  hostname_lookups: Log the names of clients or just their IP addresses
#  e.g., www.freeradius.org (on) or 206.47.27.232 (off).
#
#  The default is 'off' because it would be overall better for the net
#  if people had to knowingly turn this feature on, since enabling it
#  means that each client request will result in AT LEAST one lookup
#  request to the nameserver.   Enabling hostname_lookups will also
#  mean that your server may stop randomly for 30 seconds from time
#  to time, if the DNS requests take too long.
#
#  Turning hostname lookups off also means that the server won't block
#  for 30 seconds, if it sees an IP address which has no name associated
#  with it.
#
#  allowed values: {no, yes}
#
hostname_lookups = no

#  Core dumps are a bad thing.  This should only be set to 'yes'
#  if you're debugging a problem with the server.
#
#  allowed values: {no, yes}
#
allow_core_dumps = no

#  Regular expressions
#
#  These items are set at configure time.  If they're set to "yes",
#  then setting them to "no" turns off regular expression support.
#
#  If they're set to "no" at configure time, then setting them to "yes"
```

```
#  WILL NOT WORK.  It will give you an error.
#
regular_expressions    = yes
extended_expressions = yes

#  Log the full User-Name attribute, as it was found in the request.
#
# allowed values: {no, yes}
#
log_stripped_names = no

#  Log authentication requests to the log file.
#
#  allowed values: {no, yes}
#
log_auth = no

#  Log passwords with the authentication requests.
#  log_auth_badpass  - logs password if it's rejected
#  log_auth_goodpass - logs password if it's correct
#
#  allowed values: {no, yes}
#
log_auth_badpass = no
log_auth_goodpass = no

# usercollide:  Turn "username collision" code on and off.  See the
# "doc/duplicate-users" file
#
#  WARNING
#  !!!!!!!  Setting this to "yes" may result in the server behaving
#  !!!!!!!  strangely.  The "username collision" code will ONLY work
#  !!!!!!!  with clear-text passwords.  Even then, it may not do what
#  !!!!!!!  you want, or what you expect.
#  !!!!!!!
#  !!!!!!!  We STRONGLY RECOMMEND that you do not use this feature,
#  !!!!!!!  and that you find another way of acheiving the same goal.
#  !!!!!!!
#  !!!!!!!  e,g. module fail-over.  See 'doc/configurable_failover'
#  WARNING
#
usercollide = no

# lower_user / lower_pass:
# Lower case the username/password "before" or "after"
# attempting to authenticate.
```

```
#
#  If "before", the server will first modify the request and then try
#  to auth the user.  If "after", the server will first auth using the
#  values provided by the user.  If that fails it will reprocess the
#  request after modifying it as you specify below.
#
#  This is as close as we can get to case insensitivity.  It is the
#  admin's job to ensure that the username on the auth db side is
#  *also* lowercase to make this work
#
# Default is 'no' (don't lowercase values)
# Valid values = "before" / "after" / "no"
#
lower_user = no
lower_pass = no

# nospace_user / nospace_pass:
#
#  Some users like to enter spaces in their username or password
#  incorrectly.  To save yourself the tech support call, you can
#  eliminate those spaces here:
#
# Default is 'no' (don't remove spaces)
# Valid values = "before" / "after" / "no" (explanation above)
#
nospace_user = no
nospace_pass = no

#  The program to execute to do concurrency checks.
checkrad = ${sbindir}/checkrad

# SECURITY CONFIGURATION
#
#  There may be multiple methods of attacking on the server.  This
#  section holds the configuration items which minimize the impact
#  of those attacks
#
security {
        #
        #  max_attributes: The maximum number of attributes
        #  permitted in a RADIUS packet.  Packets which have MORE
        #  than this number of attributes in them will be dropped.
        #
        #  If this number is set too low, then no RADIUS packets
        #  will be accepted.
        #
```

```
        #  If this number is set too high, then an attacker may be
        #  able to send a small number of packets which will cause
        #  the server to use all available memory on the machine.
        #
        #  Setting this number to 0 means "allow any number of attributes"
        max_attributes = 200

        #
        #  reject_delay: When sending an Access-Reject, it can be
        #  delayed for a few seconds.  This may help slow down a DoS
        #  attack.  It also helps to slow down people trying to brute-force
        #  crack a users password.
        #
        #  Setting this number to 0 means "send rejects immediately"
        #
        #  If this number is set higher than 'cleanup_delay', then the
        #  rejects will be sent at 'cleanup_delay' time, when the request
        #  is deleted from the internal cache of requests.
        #
        #  Useful ranges: 1 to 5
        reject_delay = 1

        #
        #  status_server: Whether or not the server will respond
        #  to Status-Server requests.
        #
        #  Normally this should be set to "no", because they're useless.
        #  See: http://www.freeradius.org/rfc/rfc2865.html#Keep-Alives
        #
        #  However, certain NAS boxes may require them.
        #
        #  When sent a Status-Server message, the server responds with
        #  an Access-Accept packet, containing a Reply-Message attribute,
        #  which is a string describing how long the server has been
        #  running.
        #
        status_server = no
}

# PROXY CONFIGURATION
#
#  proxy_requests: Turns proxying of RADIUS requests on or off.
#
#  The server has proxying turned on by default.  If your system is NOT
#  set up to proxy requests to another server, then you can turn proxying
#  off here.  This will save a small amount of resources on the server.
```

```
#
#  If you have proxying turned off, and your configuration files say
#  to proxy a request, then an error message will be logged.
#
#  To disable proxying, change the "yes" to "no", and comment the
#  $INCLUDE line.
#
#  allowed values: {no, yes}
#
proxy_requests  = yes
$INCLUDE  ${confdir}/proxy.conf


# CLIENTS CONFIGURATION
#
#  Client configuration is defined in "clients.conf".
#

#  The 'clients.conf' file contains all of the information from the old
#  'clients' and 'naslist' configuration files.  We recommend that you
#  do NOT use 'client's or 'naslist', although they are still
#  supported.
#
#  Anything listed in 'clients.conf' will take precedence over the
#  information from the old-style configuration files.
#
$INCLUDE  ${confdir}/clients.conf


# SNMP CONFIGURATION
#
#  Snmp configuration is only valid if SNMP support was enabled
#  at compile time.
#
#  To enable SNMP querying of the server, set the value of the
#  'snmp' attribute to 'yes'
#
snmp    = no
$INCLUDE  ${confdir}/snmp.conf


# THREAD POOL CONFIGURATION
#
#  The thread pool is a long-lived group of threads which
#  take turns (round-robin) handling any incoming requests.
#
```

```
#  You probably want to have a few spare threads around,
#  so that high-load situations can be handled immediately.  If you
#  don't have any spare threads, then the request handling will
#  be delayed while a new thread is created, and added to the pool.
#
#  You probably don't want too many spare threads around,
#  otherwise they'll be sitting there taking up resources, and
#  not doing anything productive.
#
#  The numbers given below should be adequate for most situations.
#
thread pool {
        #  Number of servers to start initially --- should be a reasonable
        #  ballpark figure.
        start_servers = 5

        #  Limit on the total number of servers running.
        #
        #  If this limit is ever reached, clients will be LOCKED OUT, so it
        #  should NOT BE SET TOO LOW.  It is intended mainly as a brake to
        #  keep a runaway server from taking the system with it as it spirals
        #  down...
        #
        #  You may find that the server is regularly reaching the
        #  'max_servers' number of threads, and that increasing
        #  'max_servers' doesn't seem to make much difference.
        #
        #  If this is the case, then the problem is MOST LIKELY that
        #  your back-end databases are taking too long to respond, and
        #  are preventing the server from responding in a timely manner.
        #
        #  The solution is NOT do keep increasing the 'max_servers'
        #  value, but instead to fix the underlying cause of the
        #  problem: slow database, or 'hostname_lookups=yes'.
        #
        #  For more information, see 'max_request_time', above.
        #
        max_servers = 32

        #  Server-pool size regulation.  Rather than making you guess
        #  how many servers you need, FreeRADIUS dynamically adapts to
        #  the load it sees, that is, it tries to maintain enough
        #  servers to handle the current load, plus a few spare
        #  servers to handle transient load spikes.
        #
        #  It does this by periodically checking how many servers are
```

```
        #  waiting for a request.  If there are fewer than
        #  min_spare_servers, it creates a new spare.  If there are
        #  more than max_spare_servers, some of the spares die off.
        #  The default values are probably OK for most sites.
        #
        min_spare_servers = 3
        max_spare_servers = 10

        #  There may be memory leaks or resource allocation problems with
        #  the server.  If so, set this value to 300 or so, so that the
        #  resources will be cleaned up periodically.
        #
        #  This should only be necessary if there are serious bugs in the
        #  server which have not yet been fixed.
        #
        #  '0' is a special value meaning 'infinity', or 'the servers never
        #  exit'
        max_requests_per_server = 0
}

# MODULE CONFIGURATION
#
#  The names and configuration of each module is located in this section.
#
#  After the modules are defined here, they may be referred to by name,
#  in other sections of this configuration file.
#
modules {
        #
        #  Each module has a configuration as follows:
        #
        #       name [ instance ] {
        #               config_item = value
        #               ...
        #       }
        #
        #  The 'name' is used to load the 'rlm_name' library
        #  which implements the functionality of the module.
        #
        #  The 'instance' is optional.  To have two different instances
        #  of a module, it first must be referred to by 'name'.
        #  The different copies of the module are then created by
        #  inventing two 'instance' names, e.g. 'instance1' and 'instance2'
        #
        #  The instance names can then be used in later configuration
        #  INSTEAD of the original 'name'.  See the 'radutmp' configuration
```

```
#  below for an example.
#

# PAP module to authenticate users based on their stored password
#
#  As of 1.1.4, the "encryption_scheme" configuration should
#  no longer be used.  For backwards compatibility, it will still
#  work as before, but we recommend that it is not used.
#
#  The replacement is "auto_header = yes".
#  For backwards compatibility, the default is "auto_header = no",
#  but we recommend reviewing your use of the PAP module, based
#  on the documentation in "man rlm_pap".
#
#  The new capability in this module makes it MUCH easier to
#  configure the server for multiple crypt/hash schemes, AND
#  it supports more methods than before.  Please read "man rlm_pap"
#  for more detailed documentation.
#
pap {
        auto_header = yes
}

# CHAP module
#
#  To authenticate requests containing a CHAP-Password attribute.
#
chap {
        authtype = CHAP
}

# Pluggable Authentication Modules
#
#  For Linux, see:
#       http://www.kernel.org/pub/linux/libs/pam/index.html
#
#  WARNING: On many systems, the system PAM libraries have
#         memory leaks!  We STRONGLY SUGGEST that you do not
#           use PAM for authentication, due to those memory leaks.
#
pam {
        #
        #  The name to use for PAM authentication.
        #  PAM looks in /etc/pam.d/${pam_auth_name}
        #  for it's configuration.  See 'redhat/radiusd-pam'
        #  for a sample PAM configuration file.
```

```
        #
        #  Note that any Pam-Auth attribute set in the 'authorize'
        #  section will over-ride this one.
        #
        pam_auth = radiusd
}

# Unix /etc/passwd style authentication
#
unix {
        #
        #  Cache /etc/passwd, /etc/shadow, and /etc/group
        #
        #  The default is to NOT cache them.
        #
        #  For FreeBSD and NetBSD, you do NOT want to enable
        #  the cache, as it's password lookups are done via a
        #  database, so set this value to 'no'.
        #
        #  Some systems (e.g. RedHat Linux with pam_pwbd) can
        #  take *seconds* to check a password, when th passwd
        #  file containing 1000's of entries.  For those systems,
        #  you should set the cache value to 'yes', and set
        #  the locations of the 'passwd', 'shadow', and 'group'
        #  files, below.
        #
        # allowed values: {no, yes}
        cache = no

        # Reload the cache every 600 seconds (10mins). 0 to disable.
        cache_reload = 600

        #
        #  Define the locations of the normal passwd, shadow, and
        #  group files.
        #
        #  'shadow' is commented out by default, because not all
        #  systems have shadow passwords.
        #
        #  To force the module to use the system password functions,
        #  instead of reading the files, leave the following entries
        #  commented out.
        #
        #  This is required for some systems, like FreeBSD,
        #  and Mac OSX.
        #
```

```
                #       passwd = /etc/passwd
                #       shadow = /etc/shadow
                #       group = /etc/group

                #
                #  The location of the "wtmp" file.
                #  This should be moved to it's own module soon.
                #
                #  The only use for 'radlast'.  If you don't use
                #  'radlast', then you can comment out this item.
                #
                radwtmp = ${logdir}/radwtmp
        }

        #  Extensible Authentication Protocol
        #
        #  For all EAP related authentications.
        #  Now in another file, because it is very large.
        #
$INCLUDE ${confdir}/eap.conf

        # Microsoft CHAP authentication
        #
        #  This module supports MS-CHAP and MS-CHAPv2 authentication.
        #  It also enforces the SMB-Account-Ctrl attribute.
        #
        mschap {
                #
                #  As of 0.9, the mschap module does NOT support
                #  reading from /etc/smbpasswd.
                #
                #  If you are using /etc/smbpasswd, see the 'passwd'
                #  module for an example of how to use /etc/smbpasswd

                # if use_mppe is not set to no mschap will
                # add MS-CHAP-MPPE-Keys for MS-CHAPv1 and
                # MS-MPPE-Recv-Key/MS-MPPE-Send-Key for MS-CHAPv2
                #
                #use_mppe = no

                # if mppe is enabled require_encryption makes
                # encryption moderate
                #
                #require_encryption = yes

                # require_strong always requires 128 bit key
```

```
                # encryption
                #
                #require_strong = yes

                # Windows sends us a username in the form of
                # DOMAIN\user, but sends the challenge response
                # based on only the user portion.  This hack
                # corrects for that incorrect behavior.
                #
                #with_ntdomain_hack = no

                # The module can perform authentication itself, OR
                # use a Windows Domain Controller.  This configuration
                # directive tells the module to call the ntlm_auth
                # program, which will do the authentication, and return
                # the NT-Key.  Note that you MUST have "winbindd" and
                # "nmbd" running on the local machine for ntlm_auth
                # to work.  See the ntlm_auth program documentation
                # for details.
                #
                # Be VERY careful when editing the following line!
                #
                #ntlm_auth = "/path/to/ntlm_auth --request-nt-key --username=%{Stripped-User-
Name:-%{User-Name:-None}} --challenge=%{mschap:Challenge:-00} --nt-
response=%{mschap:NT-Response:-00}"
        }

        # Lightweight Directory Access Protocol (LDAP)
        #
        #  This module definition allows you to use LDAP for
        #  authorization and authentication.
        #
        #  See doc/rlm_ldap for description of configuration options
        #  and sample authorize{} and authenticate{} blocks
        #
        #  However, LDAP can be used for authentication ONLY when the
        #  Access-Request packet contains a clear-text User-Password
        #  attribute.  LDAP authentication will NOT work for any other
        #  authentication method.
        #
        #  This means that LDAP servers don't understand EAP.  If you
        #  force "Auth-Type = LDAP", and then send the server a
        #  request containing EAP authentication, then authentication
        #  WILL NOT WORK.
        #
        #  The solution is to use the default configuration, which does
```

```
#  work.
#
#  Setting "Auth-Type = LDAP" is ALMOST ALWAYS WRONG.  We
#  really can't emphasize this enough.
#
ldap {
        server = "10.10.251.23"
        port=636

        identity = "cn=radiusadmin,ou=ldap-users,ou=svcs,o=abc"
        password = rkfKSTgm
        basedn = "o=abc"
        filter = "(uid=%{Stripped-User-Name:-%{User-Name}})"

        # base_filter = "(objectclass=radiusprofile)"

        # set this to 'yes' to use TLS encrypted connections
        # to the LDAP database by using the StartTLS extended
        # operation.
        # The StartTLS operation is supposed to be used with normal
        # ldap connections instead of using ldaps (port 689) connections
        start_tls = no
        tls_mode=yes

        # tls_cacertfile= /path/to/cacert.pem
        # tls_cacertdir         = /path/to/ca/dir/
        # tls_certfile          = /path/to/radius.crt
        # tls_keyfile           = /path/to/radius.key
        # tls_randfile          = /path/to/rnd
        # tls_require_cert      = "demand"
        tls_cacertfile          = /etc/raddb/certs/rootder.b64

        # Mapping of RADIUS dictionary attributes to LDAP
        # directory attributes.
        dictionary_mapping = ${raddbdir}/ldap.attrmap

        ldap_connections_number = 5

        password_attribute = nspmPassword

        # Un-comment the following to disable Novell eDirectory account
        # policy check and intruder detection. This will work *only if*
        # FreeRADIUS is configured to build with --with-edir option.
        #
        #edir_account_policy_check=no
        edir_account_policy_check=yes
```

```
                #
                groupname_attribute = cn
                #groupmembership_filter = "(member=%{Ldap-UserDn})"
                groupmembership_filter = "(&(objectClass=GroupOfNames)(member=%{Ldap-
UserDn}))"
                groupmembership_attribute = VPN-RAS-access-group
                timeout = 4
                timelimit = 3
                net_timeout = 1
                # compare_check_items = yes
                # do_xlat = yes
                # access_attr_used_for_allow = yes
                access_attr_used_for_allow = yes

                #
                #  By default, if the packet contains a User-Password,
                #  and no other module is configured to handle the
                #  authentication, the LDAP module sets itself to do
                #  LDAP bind for authentication.
                #
                #  You can disable this behavior by setting the following
                #  configuration entry to "no".
                #
                #  allowed values: {no, yes}
                set_auth_type = yes
                # set_auth_type = no
        }

        # passwd module allows to do authorization via any passwd-like
        # file and to extract any attributes from these modules
        #
        # parameters are:
        #   filename - path to filename
        #   format - format for filename record. This parameters
        #            correlates record in the passwd file and RADIUS
        #            attributes.
        #
        #            Field marked as '*' is key field. That is, the parameter
        #            with this name from the request is used to search for
        #            the record from passwd file
        #            Attribute marked as '=' is added to reply_itmes instead
        #            of default configure_itmes
        #              Attribute marked as '~' is added to request_items
        #
        #            Field marked as ',' may contain a comma separated list
```

```
        #           of attributes.
        #   authtype - if record found this Auth-Type is used to authenticate
        #           user
        #   hashsize - hashtable size. If 0 or not specified records are not
        #           stored in memory and file is red on every request.
        #   allowmultiplekeys - if few records for every key are allowed
        #   ignorenislike - ignore NIS-related records
        #   delimiter - symbol to use as a field separator in passwd file,
        #           for format ':' symbol is always used. '\0', '\n' are
      #     not allowed
        #

        #  An example configuration for using /etc/smbpasswd.
        #
        #passwd etc_smbpasswd {
        #       filename = /etc/smbpasswd
        #       format = "*User-Name::LM-Password:NT-Password:SMB-Account-CTRL-
TEXT::"
        #       authtype = MS-CHAP
        #       hashsize = 100
        #       ignorenislike = no
        #       allowmultiplekeys = no
        #}

        #  Similar configuration, for the /etc/group file. Adds a Group-Name
        #  attribute for every group that the user is member of.
        #
        #passwd etc_group {
        #       filename = /etc/group
        #       format = "=Group-Name:::*,User-Name"
        #       hashsize = 50
        #       ignorenislike = yes
        #       allowmultiplekeys = yes
        #       delimiter = ":"
        #}

        # Realm module, for proxying.
        #
        #  You can have multiple instances of the realm module to
        #  support multiple realm syntaxs at the same time.  The
        #  search order is defined by the order in the authorize and
        #  preacct sections.
        #
        #  Four config options:
        #       format        -  must be 'prefix' or 'suffix'
        #       delimiter     -  must be a single character
```

```
#         ignore_default -  set to 'yes' or 'no'
#       ignore_null    -  set to 'yes' or 'no'
#
#  ignore_default and ignore_null can be set to 'yes' to prevent
#  the module from matching against DEFAULT or NULL realms.  This
#  may be useful if you have have multiple instances of the
#  realm module.
#
#  They both default to 'no'.
#

#  'realm/username'
#
#  Using this entry, IPASS users have their realm set to "IPASS".
realm IPASS {
        format = prefix
        delimiter = "/"
        ignore_default = no
        ignore_null = no
}

#  'username@realm'
#
realm suffix {
        format = suffix
        delimiter = "@"
        ignore_default = no
        ignore_null = no
}

#  'username%realm'
#
realm realmpercent {
        format = suffix
        delimiter = "%"
        ignore_default = no
        ignore_null = no
}

#
#  'domain\user'
#
realm ntdomain {
        format = prefix
        delimiter = "\\"
        ignore_default = no
```

```
        ignore_null = no
}

#  A simple value checking module
#
#  It can be used to check if an attribute value in the request
#  matches a (possibly multi valued) attribute in the check
#  items This can be used for example for caller-id
#  authentication.  For the module to run, both the request
#  attribute and the check items attribute must exist
#
#  i.e.
#  A user has an ldap entry with 2 radiusCallingStationId
#  attributes with values "12345678" and "12345679".  If we
#  enable rlm_checkval, then any request which contains a
#  Calling-Station-Id with one of those two values will be
#  accepted.  Requests with other values for
#  Calling-Station-Id will be rejected.
#
#  Regular expressions in the check attribute value are allowed
#  as long as the operator is '=~'
#
checkval {
        # The attribute to look for in the request
        item-name = Calling-Station-Id

        # The attribute to look for in check items. Can be multi valued
        check-name = Calling-Station-Id

        # The data type. Can be
        # string,integer,ipaddr,date,abinary,octets
        data-type = string

        # If set to yes and we dont find the item-name attribute in the
        # request then we send back a reject
        # DEFAULT is no
        #notfound-reject = no
}

#  rewrite arbitrary packets.  Useful in accounting and authorization.
#
#
#  The module can also use the Rewrite-Rule attribute. If it
#  is set and matches the name of the module instance, then
#  that module instance will be the only one which runs.
#
```

```
#  Also if new_attribute is set to yes then a new attribute
#  will be created containing the value replacewith and it
#  will be added to searchin (packet, reply, proxy, proxy_reply or config).
# searchfor,ignore_case and max_matches will be ignored in that case.
#
# Backreferences are supported: %{0} will contain the string the whole match
# and %{1} to %{8} will contain the contents of the 1st to the 8th parentheses
#
# If max_matches is greater than one the backreferences will correspond to the
# first match

#
#attr_rewrite sanecallerid {
#       attribute = Called-Station-Id
        # may be "packet", "reply", "proxy", "proxy_reply" or "config"
#       searchin = packet
#       searchfor = "[+ ]"
#       replacewith = ""
#       ignore_case = no
#       new_attribute = no
#       max_matches = 10
#       ## If set to yes then the replace string will be appended to the original string
#       append = no
#}

# Preprocess the incoming RADIUS request, before handing it off
# to other modules.
#
#  This module processes the 'huntgroups' and 'hints' files.
#  In addition, it re-writes some weird attributes created
#  by some NASes, and converts the attributes into a form which
#  is a little more standard.
#
preprocess {
        huntgroups = ${confdir}/huntgroups
        hints = ${confdir}/hints

        # This hack changes Ascend's wierd port numberings
        # to standard 0-??? port numbers so that the "+" works
        # for IP address assignments.
        with_ascend_hack = no
        ascend_channels_per_line = 23

        # Windows NT machines often authenticate themselves as
        # NT_DOMAIN\username
        #
```

```
        # If this is set to 'yes', then the NT_DOMAIN portion
        # of the user-name is silently discarded.
        #
        # This configuration entry SHOULD NOT be used.
        # See the "realms" module for a better way to handle
        # NT domains.
        with_ntdomain_hack = no

        # Specialix Jetstream 8500 24 port access server.
        #
        # If the user name is 10 characters or longer, a "/"
        # and the excess characters after the 10th are
        # appended to the user name.
        #
        # If you're not running that NAS, you don't need
        # this hack.
        with_specialix_jetstream_hack = no

        # Cisco (and Quintum in Cisco mode) sends it's VSA attributes
        # with the attribute name *again* in the string, like:
        #
        #   H323-Attribute = "h323-attribute=value".
        #
        # If this configuration item is set to 'yes', then
        # the redundant data in the the attribute text is stripped
        # out.  The result is:
        #
        #  H323-Attribute = "value"
        #
        # If you're not running a Cisco or Quintum NAS, you don't
        # need this hack.
        with_cisco_vsa_hack = no
}

# Livingston-style 'users' file
#
files {
        usersfile = ${confdir}/users
        acctusersfile = ${confdir}/acct_users
        preproxy_usersfile = ${confdir}/preproxy_users

        #  If you want to use the old Cistron 'users' file
        #  with FreeRADIUS, you should change the next line
        #  to 'compat = cistron'.  You can the copy your 'users'
        #  file from Cistron.
        compat = no
```

```
}

# Write a detailed log of all accounting records received.
#
detail {
        #  Note that we do NOT use NAS-IP-Address here, as
        #  that attribute MAY BE from the originating NAS, and
        #  NOT from the proxy which actually sent us the
        #  request.  The Client-IP-Address attribute is ALWAYS
        #  the address of the client which sent us the
        #  request.
        #
        #  The following line creates a new detail file for
        #  every radius client (by IP address or hostname).
        #  In addition, a new detail file is created every
        #  day, so that the detail file doesn't have to go
        #  through a 'log rotation'
        #
        #  If your detail files are large, you may also want
        #  to add a ':%H' (see doc/variables.txt) to the end
        #  of it, to create a new detail file every hour, e.g.:
        #
        #   ..../detail-%Y%m%d:%H
        #
        #  This will create a new detail file for every hour.
        #
        detailfile = ${radacctdir}/%{Client-IP-Address}/detail-%Y%m%d

        #
        #  The Unix-style permissions on the 'detail' file.
        #
        #  The detail file often contains secret or private
        #  information about users.  So by keeping the file
        #  permissions restrictive, we can prevent unwanted
        #  people from seeing that information.
        detailperm = 0600

        #
        # Certain attributes such as User-Password may be
        # "sensitive", so they should not be printed in the
        # detail file.  This section lists the attributes
        # that should be suppressed.
        #
        # The attributes should be listed one to a line.
        #
        #suppress {
```

```
                # User-Password
        #}
}

#
#  Many people want to log authentication requests.
#  Rather than modifying the server core to print out more
#  messages, we can use a different instance of the 'detail'
#  module, to log the authentication requests to a file.
#
#  You will also need to un-comment the 'auth_log' line
#  in the 'authorize' section, below.
#
# detail auth_log {
        # detailfile = ${radacctdir}/%{Client-IP-Address}/auth-detail-%Y%m%d

        #
        #  This MUST be 0600, otherwise anyone can read
        #  the users passwords!
        # detailperm = 0600
# }

#
#  This module logs authentication reply packets sent
#  to a NAS.  Both Access-Accept and Access-Reject packets
#  are logged.
#
#  You will also need to un-comment the 'reply_log' line
#  in the 'post-auth' section, below.
#
# detail reply_log {
        # detailfile = ${radacctdir}/%{Client-IP-Address}/reply-detail-%Y%m%d

        #
        #  This MUST be 0600, otherwise anyone can read
        #  the users passwords!
        # detailperm = 0600
# }

#
#  This module logs packets proxied to a home server.
#
#  You will also need to un-comment the 'pre_proxy_log' line
#  in the 'pre-proxy' section, below.
#
# detail pre_proxy_log {
```

```
                # detailfile = ${radacctdir}/%{Client-IP-Address}/pre-proxy-detail-%Y%m%d

                #
                #  This MUST be 0600, otherwise anyone can read
                #  the users passwords!
                # detailperm = 0600
        # }

        #
        #  This module logs response packets from a home server.
        #
        #  You will also need to un-comment the 'post_proxy_log' line
        #  in the 'post-proxy' section, below.
        #
        # detail post_proxy_log {
                # detailfile = ${radacctdir}/%{Client-IP-Address}/post-proxy-detail-%Y%m%d

                #
                #  This MUST be 0600, otherwise anyone can read
                #  the users passwords!
                # detailperm = 0600
        # }

        #
        #  The rlm_sql_log module appends the SQL queries in a log
        #  file which is read later by the radsqlrelay program.
        #
        #  This module only performs the dynamic expansion of the
        #  variables found in the SQL statements. No operation is
        #  executed on the database server. (this could be done
        #  later by an external program) That means the module is
        #  useful only with non-"SELECT" statements.
        #
        #  See rlm_sql_log(5) manpage.
        #
#       sql_log {
#               path = ${radacctdir}/sql-relay
#               acct_table = "radacct"
#               postauth_table = "radpostauth"
#
#               Start = "INSERT INTO ${acct_table} (AcctSessionId, UserName, \
#                NASIPAddress, FramedIPAddress, AcctStartTime, AcctStopTime, \
#                AcctSessionTime, AcctTerminateCause) VALUES             \
#               ('%{Acct-Session-Id}', '%{User-Name}', '%{NAS-IP-Address}', \
#                '%{Framed-IP-Address}', '%S', '0', '0', '');"
#               Stop = "INSERT INTO ${acct_table} (AcctSessionId, UserName,  \
```

```
#                NASIPAddress, FramedIPAddress, AcctStartTime, AcctStopTime, \
#                AcctSessionTime, AcctTerminateCause) VALUES             \
#                ('%{Acct-Session-Id}', '%{User-Name}', '%{NAS-IP-Address}', \
#                '%{Framed-IP-Address}', '0', '%S', '%{Acct-Session-Time}',  \
#                '%{Acct-Terminate-Cause}');"
#               Alive = "INSERT INTO ${acct_table} (AcctSessionId, UserName, \
#                NASIPAddress, FramedIPAddress, AcctStartTime, AcctStopTime, \
#                AcctSessionTime, AcctTerminateCause) VALUES             \
#                ('%{Acct-Session-Id}', '%{User-Name}', '%{NAS-IP-Address}', \
#                '%{Framed-IP-Address}', '0', '0', '%{Acct-Session-Time}','');"
#
#               Post-Auth = "INSERT INTO ${postauth_table}              \
#                (user, pass, reply, date) VALUES                        \
#                ('%{User-Name}', '%{User-Password:-Chap-Password}',      \
#                '%{reply:Packet-Type}', '%S');"
#       }

        #
        #  Create a unique accounting session Id.  Many NASes re-use
        #  or repeat values for Acct-Session-Id, causing no end of
        #  confusion.
        #
        #  This module will add a (probably) unique session id
        #  to an accounting packet based on the attributes listed
        #  below found in the packet.  See doc/rlm_acct_unique for
        #  more information.
        #
        acct_unique {
                key = "User-Name, Acct-Session-Id, NAS-IP-Address, Client-IP-Address, NAS-
Port"
        }


        #  Include another file that has the SQL-related configuration.
        #  This is another file only because it tends to be big.
        #
        #  The following configuration file is for use with MySQL.
        #
        #  For Postgresql, use:                 ${confdir}/postgresql.conf
        #  For MS-SQL, use:             ${confdir}/mssql.conf
        #  For Oracle, use:             ${confdir}/oraclesql.conf
        #
        $INCLUDE  ${confdir}/sql.conf


        #  For Cisco VoIP specific accounting with Postgresql,
```

```
#  use:            ${confdir}/pgsql-voip.conf
#
#  You will also need the sql schema from:
#          src/billing/cisco_h323_db_schema-postgres.sql
#  Note: This config can be use AS WELL AS the standard sql
#  config if you need SQL based Auth


#  Write a 'utmp' style file, of which users are currently
#  logged in, and where they've logged in from.
#
#  This file is used mainly for Simultaneous-Use checking,
#  and also 'radwho', to see who's currently logged in.
#
radutmp {
        #  Where the file is stored.  It's not a log file,
        #  so it doesn't need rotating.
        #
        filename = ${logdir}/radutmp

        #  The field in the packet to key on for the
        #  'user' name,  If you have other fields which you want
        #  to use to key on to control Simultaneous-Use,
        #  then you can use them here.
        #
        #  Note, however, that the size of the field in the
        #  'utmp' data structure is small, around 32
        #  characters, so that will limit the possible choices
        #  of keys.
        #
        #  You may want instead: %{Stripped-User-Name:-%{User-Name}}
        username = %{User-Name}


        #  Whether or not we want to treat "user" the same
        #  as "USER", or "User".  Some systems have problems
        #  with case sensitivity, so this should be set to
        #  'no' to enable the comparisons of the key attribute
        #  to be case insensitive.
        #
        case_sensitive = yes

        #  Accounting information may be lost, so the user MAY
        #  have logged off of the NAS, but we haven't noticed.
        #  If so, we can verify this information with the NAS,
        #
```

```
        #  If we want to believe the 'utmp' file, then this
        #  configuration entry can be set to 'no'.
        #
        check_with_nas = yes

        # Set the file permissions, as the contents of this file
        # are usually private.
        perm = 0600

        callerid = "yes"
}

# "Safe" radutmp - does not contain caller ID, so it can be
# world-readable, and radwho can work for normal users, without
# exposing any information that isn't already exposed by who(1).
#
# This is another 'instance' of the radutmp module, but it is given
# then name "sradutmp" to identify it later in the "accounting"
# section.
radutmp sradutmp {
        filename = ${logdir}/sradutmp
        perm = 0644
        callerid = "no"
}

# attr_filter - filters the attributes received in replies from
# proxied servers, to make sure we send back to our RADIUS client
# only allowed attributes.
attr_filter {
        attrsfile = ${confdir}/attrs
}

#  counter module:
#  This module takes an attribute (count-attribute).
#  It also takes a key, and creates a counter for each unique
#  key.  The count is incremented when accounting packets are
#  received by the server.  The value of the increment depends
#  on the attribute type.
#  If the attribute is Acct-Session-Time or of an integer type we add the
#  value of the attribute. If it is anything else we increase the
#  counter by one.
#
#  The 'reset' parameter defines when the counters are all reset to
#  zero.  It can be hourly, daily, weekly, monthly or never.
#
#  hourly: Reset on 00:00 of every hour
```

```
#  daily: Reset on 00:00:00 every day
#  weekly: Reset on 00:00:00 on sunday
#  monthly: Reset on 00:00:00 of the first day of each month
#
#  It can also be user defined. It should be of the form:
#  num[hdwm] where:
#  h: hours, d: days, w: weeks, m: months
#  If the letter is ommited days will be assumed. In example:
#  reset = 10h (reset every 10 hours)
#  reset = 12  (reset every 12 days)
#
#
#  The check-name attribute defines an attribute which will be
#  registered by the counter module and can be used to set the
#  maximum allowed value for the counter after which the user
#  is rejected.
#  Something like:
#
#  DEFAULT Max-Daily-Session := 36000
#         Fall-Through = 1
#
#  You should add the counter module in the instantiate
#  section so that it registers check-name before the files
#  module reads the users file.
#
#  If check-name is set and the user is to be rejected then we
#  send back a Reply-Message and we log a Failure-Message in
#  the radius.log
#  If the count attribute is Acct-Session-Time then on each login
#  we send back the remaining online time as a Session-Timeout attribute
#
#  The counter-name can also be used instead of using the check-name
#  like below:
#
#  DEFAULT  Daily-Session-Time > 3600, Auth-Type = Reject
#     Reply-Message = "You've used up more than one hour today"
#
#  The allowed-servicetype attribute can be used to only take
#  into account specific sessions. For example if a user first
#  logs in through a login menu and then selects ppp there will
#  be two sessions. One for Login-User and one for Framed-User
#  service type. We only need to take into account the second one.
#
#  The module should be added in the instantiate, authorize and
#  accounting sections.  Make sure that in the authorize
#  section it comes after any module which sets the
```

```
#  'check-name' attribute.
#
counter daily {
        filename = ${raddbdir}/db.daily
        key = User-Name
        count-attribute = Acct-Session-Time
        reset = daily
        counter-name = Daily-Session-Time
        check-name = Max-Daily-Session
        allowed-servicetype = Framed-User
        cache-size = 5000
}

#
#  This module is an SQL enabled version of the counter module.
#
#  Rather than maintaining seperate (GDBM) databases of
#  accounting info for each counter, this module uses the data
#  stored in the raddacct table by the sql modules. This
#  module NEVER does any database INSERTs or UPDATEs.  It is
#  totally dependent on the SQL module to process Accounting
#  packets.
#
#  The 'sqlmod_inst' parameter holds the instance of the sql
#  module to use when querying the SQL database. Normally it
#  is just "sql".  If you define more and one SQL module
#  instance (usually for failover situations), you can
#  specify which module has access to the Accounting Data
#  (radacct table).
#
#  The 'reset' parameter defines when the counters are all
#  reset to zero.  It can be hourly, daily, weekly, monthly or
#  never.  It can also be user defined. It should be of the
#  form:
#        num[hdwm] where:
#        h: hours, d: days, w: weeks, m: months
#        If the letter is ommited days will be assumed. In example:
#        reset = 10h (reset every 10 hours)
#        reset = 12  (reset every 12 days)
#
#  The 'key' parameter specifies the unique identifier for the
#  counter records (usually 'User-Name').
#
#  The 'query' parameter specifies the SQL query used to get
#  the current Counter value from the database. There are 3
#  parameters that can be used in the query:
```

```
        #               %k      'key' parameter
        #               %b      unix time value of beginning of reset period
        #               %e      unix time value of end of reset period
        #
        #  The 'check-name' parameter is the name of the 'check'
        #  attribute to use to access the counter in the 'users' file
        #  or SQL radcheck or radcheckgroup tables.
        #
        #  The 'reply-name' parameter is the name the the attribute
        #  which holds the time remaining for the user.  This is normally
        #  Session-Timeout, which makes the NAS disconnect the user
        #  once the session time is up.
        #
        #  DEFAULT  Max-Daily-Session > 3600, Auth-Type = Reject
        #     Reply-Message = "You've used up more than one hour today"
        #
        sqlcounter dailycounter {
                counter-name = Daily-Session-Time
                check-name = Max-Daily-Session
                reply-name = Session-Timeout
                sqlmod-inst = sql
                key = User-Name
                reset = daily

                # This query properly handles calls that span from the
                # previous reset period into the current period but
                # involves more work for the SQL server than those
                # below
                # For mysql:
                query = "SELECT SUM(AcctSessionTime - \
            GREATEST((%b - UNIX_TIMESTAMP(AcctStartTime)), 0)) \
            FROM radacct WHERE UserName='%{%k}' AND \
            UNIX_TIMESTAMP(AcctStartTime) + AcctSessionTime > '%b'"

                # For postgresql:
#               query = "SELECT SUM(AcctSessionTime - \
#           GREATER((%b - AcctStartTime::ABSTIME::INT4), 0)) \
#           FROM radacct WHERE UserName='%{%k}' AND \
#           AcctStartTime::ABSTIME::INT4 + AcctSessionTime > '%b'"

                # This query ignores calls that started in a previous
                # reset period and continue into into this one. But it
                # is a little easier on the SQL server
                # For mysql:
#               query = "SELECT SUM(AcctSessionTime) FROM radacct WHERE \
#           UserName='%{%k}' AND AcctStartTime > FROM_UNIXTIME('%b')"
```

```
                # For postgresql:
#               query = "SELECT SUM(AcctSessionTime) FROM radacct WHERE \
#           UserName='%{%k}' AND AND AcctStartTime::ABSTIME::INT4 > '%b'"

                # This query is the same as above, but demonstrates an
                # additional counter parameter '%e' which is the
                # timestamp for the end of the period
                # For mysql:
#               query = "SELECT SUM(AcctSessionTime) FROM radacct \
#           WHERE UserName='%{%k}' AND AcctStartTime BETWEEN \
#           FROM_UNIXTIME('%b') AND FROM_UNIXTIME('%e')"

                # For postgresql:
#               query = "SELECT SUM(AcctSessionTime) FROM radacct \
#           WHERE UserName='%{%k}' AND AcctStartTime::ABSTIME::INT4 \
#           BETWEEN '%b' AND '%e'"
        }

        sqlcounter monthlycounter {
                counter-name = Monthly-Session-Time
                check-name = Max-Monthly-Session
                reply-name = Session-Timeout
                sqlmod-inst = sql
                key = User-Name
                reset = monthly

                # This query properly handles calls that span from the
                # previous reset period into the current period but
                # involves more work for the SQL server than those
                # below
                # The same notes above about the differences between mysql
                # versus postgres queries apply here.
                query = "SELECT SUM(AcctSessionTime - \
           GREATEST((%b - UNIX_TIMESTAMP(AcctStartTime)), 0)) \
           FROM radacct WHERE UserName='%{%k}' AND \
           UNIX_TIMESTAMP(AcctStartTime) + AcctSessionTime > '%b'"

                # This query ignores calls that started in a previous
                # reset period and continue into into this one. But it
                # is a little easier on the SQL server
#               query = "SELECT SUM(AcctSessionTime) FROM radacct WHERE \
#           UserName='%{%k}' AND AcctStartTime > FROM_UNIXTIME('%b')"

                # This query is the same as above, but demonstrates an
                # additional counter parameter '%e' which is the
```

```
                # timestamp for the end of the period
#               query = "SELECT SUM(AcctSessionTime) FROM radacct \
#           WHERE UserName='%{%k}' AND AcctStartTime BETWEEN \
#           FROM_UNIXTIME('%b') AND FROM_UNIXTIME('%e')"
        }

        #
        # The "always" module is here for debugging purposes. Each
        # instance simply returns the same result, always, without
        # doing anything.
        always fail {
                rcode = fail
        }
        always reject {
                rcode = reject
        }
        always ok {
                rcode = ok
                simulcount = 0
                mpp = no
        }

        #
        #  The 'expression' module currently has no configuration.
        #
        #  This module is useful only for 'xlat'.  To use it,
        #  put 'exec' into the 'instantiate' section.  You can then
        #  do dynamic translation of attributes like:
        #
        #  Attribute-Name = `%{expr:2 + 3 + %{exec: uid -u}}`
        #
        #  The value of the attribute will be replaced with the output
        #  of the program which is executed.  Due to RADIUS protocol
        #  limitations, any output over 253 bytes will be ignored.
        expr {
        }

        #
        #  The 'digest' module currently has no configuration.
        #
        #  "Digest" authentication against a Cisco SIP server.
        #  See 'doc/rfc/draft-sterman-aaa-sip-00.txt' for details
        #  on performing digest authentication for Cisco SIP servers.
        #
        digest {
        }
```

```
#
#  Execute external programs
#
#  This module is useful only for 'xlat'.  To use it,
#  put 'exec' into the 'instantiate' section.  You can then
#  do dynamic translation of attributes like:
#
#  Attribute-Name = `%{exec:/path/to/program args}`
#
#  The value of the attribute will be replaced with the output
#  of the program which is executed.  Due to RADIUS protocol
#  limitations, any output over 253 bytes will be ignored.
#
#  The RADIUS attributes from the user request will be placed
#  into environment variables of the executed program, as
#  described in 'doc/variables.txt'
#
exec {
        wait = yes
        input_pairs = request
}

#
#  This is a more general example of the execute module.
#
#  This one is called "echo".
#
#  Attribute-Name = `%{echo:/path/to/program args}`
#
#  If you wish to execute an external program in more than
#  one section (e.g. 'authorize', 'pre_proxy', etc), then it
#  is probably best to define a different instance of the
#  'exec' module for every section.
#
exec echo {
        #
        #  Wait for the program to finish.
        #
        #  If we do NOT wait, then the program is "fire and
        #  forget", and any output attributes from it are ignored.
        #
        #  If we are looking for the program to output
        #  attributes, and want to add those attributes to the
        #  request, then we MUST wait for the program to
        #  finish, and therefore set 'wait=yes'
```

```
#
# allowed values: {no, yes}
wait = yes

#
#  The name of the program to execute, and it's
#  arguments.  Dynamic translation is done on this
#  field, so things like the following example will
#  work.
#
program = "/bin/echo %{User-Name}"

#
#  The attributes which are placed into the
#  environment variables for the program.
#
#  Allowed values are:
#
#       request         attributes from the request
#       config          attributes from the configuration items list
#       reply           attributes from the reply
#       proxy-request   attributes from the proxy request
#       proxy-reply     attributes from the proxy reply
#
#  Note that some attributes may not exist at some
#  stages.  e.g. There may be no proxy-reply
#  attributes if this module is used in the
#  'authorize' section.
#
input_pairs = request

#
#  Where to place the output attributes (if any) from
#  the executed program.  The values allowed, and the
#  restrictions as to availability, are the same as
#  for the input_pairs.
#
output_pairs = reply

#
#  When to execute the program.  If the packet
#  type does NOT match what's listed here, then
#  the module does NOT execute the program.
#
#  For a list of allowed packet types, see
#  the 'dictionary' file, and look for VALUEs
```

```
        #  of the Packet-Type attribute.
        #
        #  By default, the module executes on ANY packet.
        #  Un-comment out the following line to tell the
        #  module to execute only if an Access-Accept is
        #  being sent to the NAS.
        #
        #packet_type = Access-Accept
}

#  Do server side ip pool management. Should be added in post-auth and
#  accounting sections.
#
#  The module also requires the existance of the Pool-Name
#  attribute. That way the administrator can add the Pool-Name
#  attribute in the user profiles and use different pools
#  for different users. The Pool-Name attribute is a *check* item not
#  a reply item.
#
# Example:
# radiusd.conf: ippool students { [...] }
# users file  : DEFAULT Group == students, Pool-Name := "students"
#
# ********* IF YOU CHANGE THE RANGE PARAMETERS YOU MUST *********
# ********* THEN ERASE THE DB FILES                    *********
#
ippool main_pool {

        #  range-start,range-stop: The start and end ip
        #  addresses for the ip pool
        range-start = 192.168.1.1
        range-stop = 192.168.3.254

        #  netmask: The network mask used for the ip's
        netmask = 255.255.255.0

        #  cache-size: The gdbm cache size for the db
        #  files. Should be equal to the number of ip's
        #  available in the ip pool
        cache-size = 800

        # session-db: The main db file used to allocate ip's to clients
        session-db = ${raddbdir}/db.ippool

        # ip-index: Helper db index file used in multilink
        ip-index = ${raddbdir}/db.ipindex
```

```
                # override: Will this ippool override a Framed-IP-Address already set
                override = no

                # maximum-timeout: If not zero specifies the maximum time in seconds an
                # entry may be active. Default: 0
                maximum-timeout = 0
        }

        # $INCLUDE  ${confdir}/sqlippool.conf
        # $INCLUDE  ${confdir}/postgresqlippool.conf

        # OTP token support.  Not included by default.
        # $INCLUDE  ${confdir}/otp.conf

}

# Instantiation
#
#  This section orders the loading of the modules.  Modules
#  listed here will get loaded BEFORE the later sections like
#  authorize, authenticate, etc. get examined.
#
#  This section is not strictly needed.  When a section like
#  authorize refers to a module, it's automatically loaded and
#  initialized.  However, some modules may not be listed in any
#  of the following sections, so they can be listed here.
#
#  Also, listing modules here ensures that you have control over
#  the order in which they are initalized.  If one module needs
#  something defined by another module, you can list them in order
#  here, and ensure that the configuration will be OK.
#
instantiate {
        #
        #  Allows the execution of external scripts.
        #  The entire command line (and output) must fit into 253 bytes.
        #
        #  e.g. Framed-Pool = `%{exec:/bin/echo foo}`
        exec

        #
        #  The expression module doesn't do authorization,
        #  authentication, or accounting.  It only does dynamic
        #  translation, of the form:
        #
```

```
        #       Session-Timeout = `%{expr:2 + 3}`
        #
        #  So the module needs to be instantiated, but CANNOT be
        #  listed in any other section.  See 'doc/rlm_expr' for
        #  more information.
        #
        expr

        #
        # We add the counter module here so that it registers
        # the check-name attribute before any module which sets
        # it
#       daily
        ldap
}

#  Authorization. First preprocess (hints and huntgroups files),
#  then realms, and finally look in the "users" file.
#
#  The order of the realm modules will determine the order that
#  we try to find a matching realm.
#
#  Make *sure* that 'preprocess' comes before any realm if you
#  need to setup hints for the remote radius server
authorize {
        #
        #  The preprocess module takes care of sanitizing some bizarre
        #  attributes in the request, and turning them into attributes
        #  which are more standard.
        #
        #  It takes care of processing the 'raddb/hints' and the
        #  'raddb/huntgroups' files.
        #
        #  It also adds the %{Client-IP-Address} attribute to the request.
        preprocess

        #
        #  If you want to have a log of authentication requests,
        #  un-comment the following line, and the 'detail auth_log'
        #  section, above.
#       auth_log

#       attr_filter

        #
        #  The chap module will set 'Auth-Type := CHAP' if we are
```

```
        #  handling a CHAP request and Auth-Type has not already been set
        chap

        #
        #  If the users are logging in with an MS-CHAP-Challenge
        #  attribute for authentication, the mschap module will find
        #  the MS-CHAP-Challenge attribute, and add 'Auth-Type := MS-CHAP'
        #  to the request, which will cause the server to then use
        #  the mschap module for authentication.
        mschap

        #
        #  If you have a Cisco SIP server authenticating against
        #  FreeRADIUS, uncomment the following line, and the 'digest'
        #  line in the 'authenticate' section.
#       digest

        #
        #  Look for IPASS style 'realm/', and if not found, look for
        #  '@realm', and decide whether or not to proxy, based on
        #  that.
#       IPASS

        #
        #  If you are using multiple kinds of realms, you probably
        #  want to set "ignore_null = yes" for all of them.
        #  Otherwise, when the first style of realm doesn't match,
        #  the other styles won't be checked.
        #
        suffix
#       ntdomain

        #
        #  This module takes care of EAP-MD5, EAP-TLS, and EAP-LEAP
        #  authentication.
        #
        #  It also sets the EAP-Type attribute in the request
        #  attribute list to the EAP type from the packet.
        eap

        #
        #  Read the 'users' file
        files

        #
        #  Look in an SQL database.  The schema of the database
```

```
        #  is meant to mirror the "users" file.
        #
        #  See "Authorization Queries" in sql.conf
#       sql

        #
        #  If you are using /etc/smbpasswd, and are also doing
        #  mschap authentication, the un-comment this line, and
        #  configure the 'etc_smbpasswd' module, above.
#       etc_smbpasswd

        #
        #  The ldap module will set Auth-Type to LDAP if it has not
        #  already been set
        ldap

        #
        #  Enforce daily limits on time spent logged in.
#       daily

        #
        # Use the checkval module
#       checkval

        #
        # As of 1.1.4, you should list "pap" last in this section.
        # See "man rlm_pap" for more information.
        pap
}


#  Authentication.
#
#
#  This section lists which modules are available for authentication.
#  Note that it does NOT mean 'try each module in order'.  It means
#  that a module from the 'authorize' section adds a configuration
#  attribute 'Auth-Type := FOO'.  That authentication type is then
#  used to pick the apropriate module from the list below.
#

#  In general, you SHOULD NOT set the Auth-Type attribute.  The server
#  will figure it out on its own, and will do the right thing.  The
#  most common side effect of erroneously setting the Auth-Type
#  attribute is that one authentication method will work, but the
#  others will not.
```

```
#
#  The common reasons to set the Auth-Type attribute by hand
#  is to either forcibly reject the user, or forcibly accept him.
#
authenticate {
        #
        #  PAP authentication, when a back-end database listed
        #  in the 'authorize' section supplies a password.  The
        #  password can be clear-text, or encrypted.
        Auth-Type PAP {
                pap
        }

        #
        #  Most people want CHAP authentication
        #  A back-end database listed in the 'authorize' section
        #  MUST supply a CLEAR TEXT password.  Encrypted passwords
        #  won't work.
        Auth-Type CHAP {
                chap
        }

        #
        #  MSCHAP authentication.
#       Auth-Type MS-CHAP {
#               mschap
#       }

        #
        #  If you have a Cisco SIP server authenticating against
        #  FreeRADIUS, uncomment the following line, and the 'digest'
        #  line in the 'authorize' section.
#       digest

        #
        #  Pluggable Authentication Modules.
#       pam

        #
        #  See 'man getpwent' for information on how the 'unix'
        #  module checks the users password.  Note that packets
        #  containing CHAP-Password attributes CANNOT be authenticated
        #  against /etc/passwd!  See the FAQ for details.
        #
#       unix
```

```
        # Uncomment it if you want to use ldap for authentication
        #
        # Note that this means "check plain-text password against
        # the ldap database", which means that EAP won't work,
        # as it does not supply a plain-text password.
        Auth-Type LDAP {
                ldap
        }

        #
        #  Allow EAP authentication.
        # eap
}


#
#  Pre-accounting.  Decide which accounting type to use.
#
preacct {
        preprocess

        #
        #  Ensure that we have a semi-unique identifier for every
        #  request, and many NAS boxes are broken.
        acct_unique

        #
        #  Look for IPASS-style 'realm/', and if not found, look for
        #  '@realm', and decide whether or not to proxy, based on
        #  that.
        #
        #  Accounting requests are generally proxied to the same
        #  home server as authentication requests.
#       IPASS
        suffix
#       ntdomain

        #
        #  Read the 'acct_users' file
        files
}

#
#  Accounting.  Log the accounting data.
#
accounting {
```

```
        #
        #  Create a 'detail'ed log of the packets.
        #  Note that accounting requests which are proxied
        #  are also logged in the detail file.
        detail
#       daily

        #  Update the wtmp file
        #
        #  If you don't use "radlast", you can delete this line.
        unix

        #
        #  For Simultaneous-Use tracking.
        #
        #  Due to packet losses in the network, the data here
        #  may be incorrect.  There is little we can do about it.
        radutmp
#       sradutmp

        #  Return an address to the IP Pool when we see a stop record.
#       main_pool
#       sqlippool

        #
        #  Log traffic to an SQL database.
        #
        #  See "Accounting queries" in sql.conf
#       sql

        #
        #  Instead of sending the query to the SQL server,
        #  write it into a log file.
        #
#       sql_log

        #  Cisco VoIP specific bulk accounting
#       pgsql-voip

}


#  Session database, used for checking Simultaneous-Use. Either the radutmp
#  or rlm_sql module can handle this.
#  The rlm_sql module is *much* faster
session {
```

```
        radutmp

        #
        #  See "Simultaneous Use Checking Querie" in sql.conf
#       sql
}


#  Post-Authentication
#  Once we KNOW that the user has been authenticated, there are
#  additional steps we can take.
post-auth {
        #  Get an address from the IP Pool.
#       main_pool
#       sqlippool

        #
        #  If you want to have a log of authentication replies,
        #  un-comment the following line, and the 'detail reply_log'
        #  section, above.
#       reply_log

        #
        #  After authenticating the user, do another SQL query.
        #
        #  See "Authentication Logging Queries" in sql.conf
#       sql

        #
        #  Instead of sending the query to the SQL server,
        #  write it into a log file.
        #
#       sql_log

        #
        #  Un-comment the following if you have set
        #  'edir_account_policy_check = yes' in the ldap module sub-section of
        #  the 'modules' section.
        #
        ldap
        #
        #  Access-Reject packets are sent through the REJECT sub-section of the
        #  post-auth section.
        #  Uncomment the following and set the module name to the ldap instance
        #  name if you have set 'edir_account_policy_check = yes' in the ldap
        #  module sub-section of the 'modules' section.
```

```
        #
        Post-Auth-Type REJECT {
                ldap
        }

}

#
#  When the server decides to proxy a request to a home server,
#  the proxied request is first passed through the pre-proxy
#  stage.  This stage can re-write the request, or decide to
#  cancel the proxy.
#
#  Only a few modules currently have this method.
#
pre-proxy {
#       attr_rewrite

        #  Uncomment the following line if you want to change attributes
        #  as defined in the preproxy_users file.
#       files

        #  If you want to have a log of packets proxied to a home
        #  server, un-comment the following line, and the
        #  'detail pre_proxy_log' section, above.
#       pre_proxy_log
}

#
#  When the server receives a reply to a request it proxied
#  to a home server, the request may be massaged here, in the
#  post-proxy stage.
#
post-proxy {

        #  If you want to have a log of replies from a home server,
        #  un-comment the following line, and the 'detail post_proxy_log'
        #  section, above.
#       post_proxy_log

#       attr_rewrite

        #  Uncomment the following line if you want to filter replies from
        #  remote proxies based on the rules defined in the 'attrs' file.

#       attr_filter
```

```
        #
        #  If you are proxying LEAP, you MUST configure the EAP
        #  module, and you MUST list it here, in the post-proxy
        #  stage.
        #
        #  You MUST also use the 'nostrip' option in the 'realm'
        #  configuration.  Otherwise, the User-Name attribute
        #  in the proxied request will not match the user name
        #  hidden inside of the EAP packet, and the end server will
        #  reject the EAP request.
        #
        eap
}
```

Sample clients.conf

```
#
# clients.conf - client configuration directives
#
#########################################################################

#########################################################################
#
#  Definition of a RADIUS client (usually a NAS).
#
#  The information given here over rides anything given in the
#  'clients' file, or in the 'naslist' file.  The configuration here
#  contains all of the information from those two files, and allows
#  for more configuration items.
#
#  The "shortname" is be used for logging.  The "nastype", "login" and
#  "password" fields are mainly used for checkrad and are optional.
#

#
#  Defines a RADIUS client.  The format is 'client [hostname|ip-address]'
#
#  '127.0.0.1' is another name for 'localhost'.  It is enabled by default,
#  to allow testing of the server after an initial installation.  If you
#  are not going to be permitting RADIUS queries from localhost, we suggest
#  that you delete, or comment out, this entry.
#
client 127.0.0.1 {
        #
        #  The shared secret use to "encrypt" and "sign" packets between
        #  the NAS and FreeRADIUS.  You MUST change this secret from the
```

```
        #  default, otherwise it's not a secret any more!
        #
        #  The secret can be any string, up to 31 characters in length.
        #
        secret          = testing123

        #
        #  The short name is used as an alias for the fully qualified
        #  domain name, or the IP address.
        #
        shortname       = localhost

        #
        # the following three fields are optional, but may be used by
        # checkrad.pl for simultaneous use checks
        #

        #
        # The nastype tells 'checkrad.pl' which NAS-specific method to
        #  use to query the NAS for simultaneous use.
        #
        #  Permitted NAS types are:
        #
        #       cisco
        #       computone
        #       livingston
        #       max40xx
        #       multitech
        #       netserver
        #       pathras
        #       patton
        #       portslave
        #       tc
        #       usrhiper
        #       other           # for all other types

        #
        nastype     = other     # localhost isn't usually a NAS...

        #
        #  The following two configurations are for future use.
        #  The 'naspasswd' file is currently used to store the NAS
        #  login name and password, which is used by checkrad.pl
        #  when querying the NAS for simultaneous use.
        #
#       login       = !root
```

```
#       password    = someadminpas
}

client 10.10..114.4 {
        secret          = 4radius2use
        shortname       = Contivity
}

client 10.10..116.11 {
        secret          = 4radius2use
        shortname       = RAS
}

client 10.10..217.217 {
        secret          = 4radius2use
        shortname       = Kevin PC
}


#client some.host.org {
#       secret          = testing123
#       shortname       = localhost
#}

#
#  You can now specify one secret for a network of clients.
#  When a client request comes in, the BEST match is chosen.
#  i.e. The entry from the smallest possible network.
#
#client 192.168.0.0/24 {
#       secret          = testing123-1
#       shortname       = private-network-1
#}
#
#client 192.168.0.0/16 {
#       secret          = testing123-2
#       shortname       = private-network-2
#}


#client 10.10.10.10 {
#       # secret and password are mapped through the "secrets" file.
#       secret      = testing123
#       shortname   = liv1
#      # the following three fields are optional, but may be used by
#      # checkrad.pl for simultaneous usage checks
```

```
#       nastype     = livingston
#       login       = !root
#       password    = someadminpas
#}
```

Sample users file

```
#
#       Please read the documentation file ../doc/processing_users_file,
#       or 'man 5 users' (after installing the server) for more information.
#
#       As of 1.1.4, you SHOULD NOT use Auth-Type.  See "man rlm_pap"
#       for a much better way of dealing with differing passwords.
#       If you set Auth-Type, SOME AUTHENTICATION METHODS WILL NOT WORK.
#       If you don't set Auth-Type, the server will figure out what to do,
#       and will almost always do the right thing.
#
#       This file contains authentication security and configuration
#       information for each user.  Accounting requests are NOT processed
#       through this file.  Instead, see 'acct_users', in this directory.
#
#       The first field is the user's name and can be up to
#       253 characters in length.  This is followed (on the same line) with
#       the list of authentication requirements for that user.  This can
#       include password, comm server name, comm server port number, protocol
#       type (perhaps set by the "hints" file), and huntgroup name (set by
#       the "huntgroups" file).
#
#       Indented (with the tab character) lines following the first
#       line indicate the configuration values to be passed back to
#       the comm server to allow the initiation of a user session.
#       This can include things like the PPP configuration values
#       or the host to log the user onto.
#
#       If you are not sure why a particular reply is being sent by the
#       server, then run the server in debugging mode (radiusd -X), and
#       you will see which entries in this file are matched.
#
#       When an authentication request is received from the comm server,
#       these values are tested. Only the first match is used unless the
#       "Fall-Through" variable is set to "Yes".
#
#       A special user named "DEFAULT" matches on all usernames.
#       You can have several DEFAULT entries. All entries are processed
#       in the order they appear in this file. The first entry that
#       matches the login-request will stop processing unless you use
#       the Fall-Through variable.
```

```
#
#       You can include another `users' file with `$INCLUDE users.other'
#

#
#       For a list of RADIUS attributes, and links to their definitions,
#       see:
#
#       http://www.freeradius.org/rfc/attributes.html
#

#
# Deny access for a specific user.  Note that this entry MUST
# be before any other 'Auth-Type' attribute which results in the user
# being authenticated.
#
# Note that there is NO 'Fall-Through' attribute, so the user will not
# be given any additional resources.
#
#lameuser       Auth-Type := Reject
#               Reply-Message = "Your account has been disabled."

#
# Deny access for a group of users.
#
# Note that there is NO 'Fall-Through' attribute, so the user will not
# be given any additional resources.
#
#DEFAULT    Group == "disabled", Auth-Type := Reject
#               Reply-Message = "Your account has been disabled."
#

DEFAULT Ldap-Group == "cn=VPN-RAS-access-group,o=abc", Auth-Type := LDAP
        Fall-Through = 0

DEFAULT      Auth-Type = REJECT
        Fall-Through = 1
#

#
# This is a complete entry for "steve". Note that there is no Fall-Through
# entry so that no DEFAULT entry will be used, and the user will NOT
# get any attributes in addition to the ones listed here.
#
#steve  Cleartext-Password := "testing"
#       Service-Type = Framed-User,
```

```
#       Framed-Protocol = PPP,
#       Framed-IP-Address = 172.16.3.33,
#       Framed-IP-Netmask = 255.255.255.0,
#       Framed-Routing = Broadcast-Listen,
#       Framed-Filter-Id = "std.ppp",
#       Framed-MTU = 1500,
#       Framed-Compression = Van-Jacobsen-TCP-IP

#
# This is an entry for a user with a space in their name.
# Note the double quotes surrounding the name.
#
#"John Doe"   Cleartext-Password := "hello"
#               Reply-Message = "Hello, %u"

#
# Dial user back and telnet to the default host for that port
#
#Deg   Cleartext-Password := "ge55ged"
#       Service-Type = Callback-Login-User,
#       Login-IP-Host = 0.0.0.0,
#       Callback-Number = "9,5551212",
#       Login-Service = Telnet,
#       Login-TCP-Port = Telnet

#
# Another complete entry. After the user "dialbk" has logged in, the
# connection will be broken and the user will be dialed back after which
# he will get a connection to the host "timeshare1".
#
#dialbkCleartext-Password := "callme"
#       Service-Type = Callback-Login-User,
#       Login-IP-Host = timeshare1,
#       Login-Service = PortMaster,
#       Callback-Number = "9,1-800-555-1212"

#
# user "swilson" will only get a static IP number if he logs in with
# a framed protocol on a terminal server in Alphen (see the huntgroups file).
#
# Note that by setting "Fall-Through", other attributes will be added from
# the following DEFAULT entries
#
#swilson        Service-Type == Framed-User, Huntgroup-Name == "alphen"
#               Framed-IP-Address = 192.168.1.65,
#               Fall-Through = Yes
```

```
#
# If the user logs in as 'username.shell', then authenticate them
# against the system database, give them shell access, and stop processing
# the rest of the file.
#
# Note that authenticating against an /etc/passwd file works ONLY for PAP,
# and not for CHAP, MS-CHAP, or EAP.
#
#DEFAULT    Suffix == ".shell", Auth-Type := System
#                  Service-Type = Login-User,
#                  Login-Service = Telnet,
#                  Login-IP-Host = your.shell.machine


#
# The rest of this file contains the several DEFAULT entries.
# DEFAULT entries match with all login names.
# Note that DEFAULT entries can also Fall-Through (see first entry).
# A name-value pair from a DEFAULT entry will _NEVER_ override
# an already existing name-value pair.
#

#
# First setup all accounts to be checked against the UNIX /etc/passwd.
# (Unless a password was already given earlier in this file).
#
# DEFAULT   Auth-Type = LDAP
#         Fall-Through = 0

# DEFAULT   Auth-Type = Reject
# DEFAULT   Auth-Type = System
#         Fall-Through = 1

#
# Set up different IP address pools for the terminal servers.
# Note that the "+" behind the IP address means that this is the "base"
# IP address. The Port-Id (S0, S1 etc) will be added to it.
#
#DEFAULT    Service-Type == Framed-User, Huntgroup-Name == "alphen"
#                  Framed-IP-Address = 192.168.1.32+,
#                  Fall-Through = Yes

#DEFAULT    Service-Type == Framed-User, Huntgroup-Name == "delft"
#                  Framed-IP-Address = 192.168.2.32+,
#                  Fall-Through = Yes
```

```
#
# Defaults for all framed connections.
#
DEFAULT      Service-Type == Framed-User
        Framed-IP-Address = 255.255.255.254,
        Framed-MTU = 576,
        Service-Type = Framed-User,
        Fall-Through = Yes

#
# Default for PPP: dynamic IP address, PPP mode, VJ-compression.
# NOTE: we do not use Hint = "PPP", since PPP might also be auto-detected
#       by the terminal server in which case there may not be a "P" suffix.
#       The terminal server sends "Framed-Protocol = PPP" for auto PPP.
#
DEFAULT      Framed-Protocol == PPP
        Framed-Protocol = PPP,
        Framed-Compression = Van-Jacobson-TCP-IP

#
# Default for CSLIP: dynamic IP address, SLIP mode, VJ-compression.
#
#DEFAULT    Hint == "CSLIP"
#       Framed-Protocol = SLIP,
#       Framed-Compression = Van-Jacobson-TCP-IP

#
# Default for SLIP: dynamic IP address, SLIP mode.
#
#DEFAULT    Hint == "SLIP"
#       Framed-Protocol = SLIP

#
# Last default: rlogin to our main server.
#
#DEFAULT
#       Service-Type = Login-User,
#       Login-Service = Rlogin,
#       Login-IP-Host = shellbox.ispdomain.com

# #
# # Last default: shell on the local terminal server.
# #
# DEFAULT
#       Service-Type = Shell-User
```

 # On no match, the user is denied access.