```
server custom {
######################################################################
#
#	As of 2.0.0, FreeRADIUS supports virtual hosts using the
#	"server" section, and configuration directives.
#
#	Virtual hosts should be put into the "sites-available"
#	directory.  Soft links should be created in the "sites-enabled"
#	directory to these files.  This is done in a normal installation.
#
#	$Id$
#
######################################################################
#
#	Read "man radiusd" before editing this file.  See the section
#	titled DEBUGGING.  It outlines a method where you can quickly
#	obtain the configuration you want, without running into
#	trouble.  See also "man unlang", which documents the format
#	of this file.
#
#	This configuration is designed to work in the widest possible
#	set of circumstances, with the widest possible number of
#	authentication methods.  This means that in general, you should
#	need to make very few changes to this file.
#
#	The best way to configure the server for your local system
#	is to CAREFULLY edit this file.  Most attempts to make large
#	edits to this file will BREAK THE SERVER.  Any edits should
#	be small, and tested by running the server with "radiusd -X".
#	Once the edits have been verified to work, save a copy of these
#	configuration files somewhere.  (e.g. as a "tar" file).  Then,
#	make more edits, and test, as above.
#
#	There are many "commented out" references to modules such
#	as ldap, sql, etc.  These references serve as place-holders.
#	If you need the functionality of that module, then configure
#	it in radiusd.conf, and un-comment the references to it in
#	this file.  In most cases, those small changes will result
#	in the server being able to connect to the DB, and to
#	authenticate users.
#
######################################################################

#
#	In 1.x, the "authorize", etc. sections were global in
#	radiusd.conf.  As of 2.0, they SHOULD be in a server section.
#
#	The server section with no virtual server name is the "default"
#	section.  It is used when no server name is specified.
#
#	We don't indent the rest of this file, because doing so
#	would make it harder to read.
#
```

```
#  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.
    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

    #
    #  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

    #
    #  The WiMAX specification says that the Calling-Station-Id
    #  is 6 octets of the MAC.  This definition conflicts with
    #  RFC 3580, and all common RADIUS practices.  Un-commenting
    #  the "wimax" module here means that it will fix the
    #  Calling-Station-Id attribute to the normal format as
    #  specified in RFC 3580 Section 3.21
```

```
#	wimax

	#
	#  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
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.
	#
	#  As of 2.0, the EAP module returns "ok" in the authorize stage
	#  for TTLS and PEAP.  In 1.x, it never returned "ok" here, so
	#  this change is compatible with older configurations.
	#
	#  The example below uses module failover to avoid querying all
	#  of the following modules if the EAP module returns "ok".
	#  Therefore, your LDAP and/or SQL servers will not be queried
	#  for the many packets that go back and forth to set up TTLS
	#  or PEAP.  The load on those servers will therefore be reduced.
	#
	eap {
		ok = return
	}

	#
	#  Pull crypt'd passwords from /etc/passwd or /etc/shadow,
	#  using the system API's to get the password.  If you want
	#  to read /etc/passwd or /etc/shadow directly, see the
	#  passwd module in radiusd.conf.
	#
#	unix

	#
	#  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

	expiration
	logintime

	#
	#  If no other module has claimed responsibility for
	#  authentication, then try to use PAP.  This allows the
	#  other modules listed above to add a "known good" password
	#  to the request, and to do nothing else.  The PAP module
	#  will then see that password, and use it to do PAP
	#  authentication.
	#
	#  This module should be listed last, so that the other modules
	#  get a chance to set Auth-Type for themselves.
	#
	pap

	#
	#  If "status_server = yes", then Status-Server messages are passed
	#  through the following section, and ONLY the following section.
	#  This permits you to do DB queries, for example.  If the modules
	#  listed here return "fail", then NO response is sent.
	#
#	Autz-Type Status-Server {
#
#	}
}
```

```
#  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 (Auth-Type := Reject),
#  or to or forcibly accept the user (Auth-Type := Accept).
#
#  Note that Auth-Type := Accept will NOT work with EAP.
#
#  Please do not put "unlang" configurations into the "authenticate"
#  section.  Put them in the "post-auth" section instead.  That's what
#  the post-auth section is for.
#
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

	#
	#  The older configurations sent a number of attributes in
	#  Access-Challenge packets, which wasn't strictly correct.
	#  If you want to filter out these attributes, uncomment
	#  the following lines.
	#
#	Auth-Type eap {
#		eap {
#			handled = 1
#		}
#		if (handled && (Response-Packet-Type == Access-Challenge)) {
#			attr_filter.access_challenge.post-auth
#			handled  # override the "updated" code from attr_filter
#		}
#	}
}


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

```
    #
    #  Session start times are *implied* in RADIUS.
    #  The NAS never sends a "start time".  Instead, it sends
    #  a start packet, *possibly* with an Acct-Delay-Time.
    #  The server is supposed to conclude that the start time
    #  was "Acct-Delay-Time" seconds in the past.
    #
    #  The code below creates an explicit start time, which can
    #  then be used in other modules.
    #
    #  The start time is: NOW - delay - session_length
    #

#     update request {
#       FreeRADIUS-Acct-Session-Start-Time = "%{expr: %l -
%{%{Acct-Session-Time}:-0} - %{%{Acct-Delay-Time}:-0}}"
#   }



    #
    #  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
    # Setting suffix as realm default MSCHAP needs ntdomain
ntdomain

    #
    #  Read the 'acct_users' file
#   files
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

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

	#
	#  If you receive stop packets with zero session length,
	#  they will NOT be logged in the database.  The SQL module
	#  will print a message (only in debugging mode), and will
	#  return "noop".
	#
	#  You can ignore these packets by uncommenting the following
	#  three lines.  Otherwise, the server will not respond to the
	#  accounting request, and the NAS will retransmit.
	#
#	if (noop) {
#	    ok
#	}

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

	#  Cisco VoIP specific bulk accounting
#	pgsql-voip

	#  Filter attributes from the accounting response.
	attr_filter.accounting_response

	#
	#  See "Autz-Type Status-Server" for how this works.
```

```
    #
#   Acct-Type Status-Server {
#
#   }
}


#  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 Queries" 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

    #
    #  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

#   exec
```

```
    #
    #  Calculate the various WiMAX keys.  In order for this to work,
    #  you will need to define the WiMAX NAI, usually via
    #
    #   update request {
    #          WiMAX-MN-NAI = "%{User-Name}"
    #   }
    #
    #  If you want various keys to be calculated, you will need to
    #  update the reply with "template" values.  The module will see
    #  this, and replace the template values with the correct ones
    #  taken from the cryptographic calculations.  e.g.
    #
    #   update reply {
    #       WiMAX-FA-RK-Key = 0x00
    #       WiMAX-MSK = "%{EAP-MSK}"
    #   }
    #
    #  You may want to delete the MS-MPPE-*-Keys from the reply,
    #  as some WiMAX clients behave badly when those attributes
    #  are included.  See "raddb/modules/wimax", configuration
    #  entry "delete_mppe_keys" for more information.
    #
#   wimax

    #  If the WiMAX module did it's work, you may want to do more
    #  things here, like delete the MS-MPPE-*-Key attributes.
    #
    #   if (updated) {
    #       update reply {
    #           MS-MPPE-Recv-Key !* 0x00
    #           MS-MPPE-Send-Key !* 0x00
    #       }
    #   }

    #
    #  Access-Reject packets are sent through the REJECT sub-section of the
    #  post-auth section.
    #
    #  Add the ldap module name (or instance) if you have set
    #  'edir_account_policy_check = yes' in the ldap module configuration
    #
    #Post-Auth-Type REJECT {
    #   attr_filter.access_reject
    #}
    # Load module: ldap
    ldap
    if ("%{request:User-Name}" =~ /^host\/(.*).domain$/) {
        update request {
            User-Name := "%{1}$"
        }
    }
    # Gemeinsame SSID
    # Anforderung 1. Schuleigene Geräte (iPads, Windows Rechner etc.) sollen ein
eigenes WLAN-Netz erhalten.
    if
("%{ldap:ldap:///dc=schulen,dc=landkreis,dc=univentiontest?cn?sub?(&(memberUid=%{U
ser-Name})(cn=schuleigene-rechner))}") {
        update reply {
            Reply-Message := "DEBUG: Schuleigene Rechner"
            Tunnel-Type := VLAN
            Tunnel-Medium-Type := IEEE-802
            Tunnel-Private-Group-Id := "1032"
        }
    }
    # Anforderung 2. BYOD der Lehrerschaft sollen ein eigenes WLAN-Netz
erhalten.
    elsif
("%{ldap:ldap:///dc=schulen,dc=landkreis,dc=univentiontest?cn?sub?(&(memberUid=%{U
ser-Name})(cn=lehrer-011))}") {
        update reply {
            Reply-Message := "DEBUG: Lehrer Schule BYOD"
            Tunnel-Type := VLAN
            Tunnel-Medium-Type := IEEE-802
            Tunnel-Private-Group-Id := "1064"
        }
    }
    # Anforderung 3. BYOD der Schülerschaft sollen ein eigenes WLAN-Netz
erhalten.
    elsif
("%{ldap:ldap:///dc=schulen,dc=landkreis,dc=univentiontest?cn?sub?(&(memberUid=%{U
ser-Name})(cn=schueler-011))}") {
        update reply {
            Reply-Message := "DEBUG: Schüler Schule BYOD"
            Tunnel-Type := VLAN
            Tunnel-Medium-Type := IEEE-802
            Tunnel-Private-Group-Id := "1064"
        }
    }
    else {
        update reply {
            Reply-Message := "DEBUG: Not found, reject"
        }
        reject
    }
}

#
#  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

	#  Uncomment the following line if you want to filter requests
	#  sent to remote servers based on the rules defined in the
	#  'attrs.pre-proxy' file.
#	attr_filter.pre-proxy

	#  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.post-proxy

	#
	#  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

	#
	#  If the server tries to proxy a request and fails, then the
	#  request is processed through the modules in this section.
	#
	#  The main use of this section is to permit robust proxying
	#  of accounting packets.  The server can be configured to
	#  proxy accounting packets as part of normal processing.
	#  Then, if the home server goes down, accounting packets can
	#  be logged to a local "detail" file, for processing with
	#  radrelay.  When the home server comes back up, radrelay
	#  will read the detail file, and send the packets to the
	#  home server.
	#
	#  With this configuration, the server always responds to
	#  Accounting-Requests from the NAS, but only writes
	#  accounting packets to disk if the home server is down.
	#
#	Post-Proxy-Type Fail {
#			detail
#	}

}
}
```