# Server configuration

Updated: 4/06/2021

The key component in 802.1X is the RADIUS server which is capable of AAA. There are several widely-deployed commercial RADIUS servers available: Microsoft® IAS, Cisco® ACS, Funk Steel-Belted Radius. There are also two widely deployed Open Source RADIUS servers available, FreeRADIUS and RADIATOR. FreeRADIUS is non-commercial GPL software, RADIATOR is commercial.

All servers have their strengths and weaknesses: Microsoft® IAS limits the type of authentications (EAP-PEAP and EAP-TLS) and FreeRADIUS has third party commercial support. However, all of them perform the core functionality required for central 802.1X RADIUS authentication. As FreeRADIUS supports many types of authentication and authorisation and is free, this Technical Guide documents the use of FreeRADIUS.

## FreeRADIUS

FreeRADIUS is a popular Open Source RADIUS server which can operate in enterprise environments handling tens of thousands of authentications a day.

This Technical Guide refers to FreeRADIUS version 2.0 for several reasons: firstly, when installing FreeRADIUS 2.0, the EAP section is automatically populated with a working certificate system ideal for testing (you should replace the certificates with appropriate versions before going into service); the EAP engine is vastly improved over version 1.1.x (an EAP authentication now only needs to pass through the LDAP or SQL section once, thus vastly reducing load on such servers); and finally the server is now equipped with 'unlang' – a pseudo code which enables very flexible checking parameters and testing loops in the main configuration files.

## Installing FreeRADIUS

Prerequisites: a Linux® server running a recent distribution (with kernel 2.6 or above), at least OpenSSL 0.9.8b, development tools (GCC compiler) and a network interface able to communicate with the NAS (e.g. switch, wireless access point or WiFi central controller). A system with at least two NICs is recommended – one on a managed network and one on the management network, with at least 100Mbit/s ports configured for full-duplex operation to reduce latency.

Download FreeRADIUS from: http://www.freeradius.org/ [1]

There may be a version available for your distribution; however, check that it is a recent version.

***The current version of FreeRADIUS is 3.0.22 - the information below relating to version 1.1 is historical.***

Whilst 1.1.x does not have the same flexibility if you are going to create or import your own certificates, if you are familiar with the system then this guide is largely suitable for version 1.1.x too. 1.1.x specific issues may be highlighted.

After downloading, extract the .tar.gz archive and run the configure, make and make install steps as per the INSTALL document within the archive. More advanced users might want to run ./configure --help to check what special options might be appropriate. Please ensure that you read the output of the ./configure stage. Check for any lines with ‘WARNING’ – you may need to install several development/support packages to get FreeRADIUS to work with your required environment.

After compilation and installing (you will have noted the generation of SSL certificates if this is a first-time 2.0 install) FreeRADIUS is ready for configuring.

Whilst most of the configuration files will be very much site specific (sql.conf, proxy.conf, LDAP/SQL authentication) a core set of configuration files are common to all systems for 802.1X authentication. After an initial editing of radiusd.conf to handle site-specific requirements – the listen section, the user/group that FreeRADIUS runs as (must be able to read  configuration files and write log files) – the main configuration file is eap.conf. This file is fairly well self-documented but we will now visit each stanza to guide through a configuration.

## eap.conf

The file starts with an ‘eap’ statement, so the server activates the eap module. Several core EAP selections are then made: default_eap_type (set this to your default expected incoming type, e.g. peap), timer_expire, ignore_unknown_eap_types and cisco_accounting_username_bug. These are all straightforward and only need to be changed if you feel it correct for your environment. Several EAP types are then configured. If you are not using such types, then comment them out.

Note: If you are using PEAP or TTLS then you will need to keep the EAP-TLS section active.

## EAP-TLS section

This is the most important section in the 802.1X configuration. Within this section the SSL private key, the Certificate Authority (CA) and the diffie-hellman and random generator are defined, as are critical values such as fragment size or EAP packets, Certificate Revocation List checking, ciphers allowed for encryption and EAP packet options such as including length.

In a vanilla installation, all of these items are auto-configured for production environments: however, the CA-File, Private Certificate and private key will need to be corrected.

The next two sections are for EAP-TTLS and PEAP. Both of these enable the default inner type to be defined (MSCHAPv2 is the default), the request copied to the tunnel (allows all of the outer values to be passed through to the inner tunnel engine) and the tunnelled reply to be used, which needs to be completed to return RADIUS attributes to the switch or access point – such as VLAN return attributes.

There is a strong warning regarding PEAP and Windows® clients. One must ensure when generating certificates (or getting them generated by a third party) that they contain the required certificate extensions – as per the default configuration documentation or URLs:

- http://support.microsoft.com/kb/814394/en-us [2]
- http://support.microsoft.com/kb/885453/en-us [3]

Once the basic configuration has been completed, the FreeRADIUS daemon is ready to be started in debug mode radiusd -X to confirm the system is operational. If the daemon does not start correctly, check the output very carefully to see why it is failing. At this point, with no AAA traffic, the output is short and concise; therefore it is easy to spot an issue such as unreadable configuration file or non-existing certificate.

To complete configuring FreeRADIUS as a RADIUS server, there are several actions. This includes editing the sql.conf file if you are to use MySQL, Oracle or PostgreSQL for authorization, authentication or accounting. Sites-enabled/default will need to be edited to remove any methods that you do not use, for example: CHAP, LDAP and Digest.

The users file will need to be edited to add a test account, or to set DEFAULT values (e.g. a default VLAN return attribute). There is a good starting guide available for EAP-PEAP against an Active Directory:
http://wiki.freeradius.org/FreeRADIUS_Active_Directory_Integration_HOWTO [4].

To use LDAP, the radiusd.conf file and  sites-enabled/default will need to be edited accordingly: for example, to use TLS for the communication, or to set the access user id.

In addition, clients.conf contains a list of all the NAS devices that can communicate with FreeRADIUS. If the device is not listed then it cannot communicate with the FreeRADIUS daemon on UDP ports 1812,1813 and 1814. These will also need to be opened on the server's local firewall, if there is one running, and the naslist can also be supplied as an SQL table.

proxy.conf is applicable to sending RADIUS packets to another RADIUS server and the significance of this is covered further in Section 4.6.

Whilst man pages, the inline configuration documents and various other files in the FreeRADIUS archive cover the basics, there is an online guide available at:
http://wiki.freeradius.org/Main_Page [5].

For further help and guidance, the freeradius users mailing list is hosted from: http://www.freeradius.org/list/ [6]. As with such lists, the more information you provide, the better the help you will receive. radius -X output is always required for ‘it does not work’ style cases. However, ensure you have sanitised the output before sending it, for example obfuscate IP addresses, change passwords and user names.

## Example configuration segments

Whilst each site will have local authentication and authorisation requirements, there are some key parts that are common: for example, returning a specific VLAN for a certain user. This is a key feature of 802.1X and allows functions such as ‘user is quarantined’, ‘user is a guest on guest network’, ‘staff on different network to students’ etc.

```
DEFAULT Tunnel-Medium-Type = IEEE-802,

Tunnel-Private-Group-Id = 666,

Tunnel-Type = VLAN
```

This segment is the bare code required to send back the message ‘move this device to VLAN 666’. It could be contained in the users file (usually with a matching condition after the DEFAULT):

```
DEFAULT NAS-IP-Address == "192.168.1.5"

Tunnel-Medium-Type = IEEE-802,

Tunnel-Private-Group-Id = 666,

Tunnel-Type = VLAN
```

With this statement, any device accessing the network from the NAS with address 192.168.1.6 gets put onto VLAN 666.

The configuration could be placed into return attributes in an SQL table – which can then be triggered for certain IDs or NAS addresses, or it could be embedded in a PERL or Python script which is triggered in the authorization or post-authorization section.

```
sites-enabled/default

post-auth {

perl

}

experimental.conf
```

```
perl {
  module = /path/to/yourcode.pm
func_post_auth = post_auth
max_clones = 64
start_clones = 64
min_spare_clones = 0
max_spare_clones = 64
cleanup_delay = 2
max_request_per_clone = 0
# ensure that the above values match the main radius server!!!
}
/path/to/yourcode.pm
# Example code!
# Date:   20/11/2007
#
use strict;
# very important ! The script needs the values from radiusd.
use vars qw(%RAD_REQUEST %RAD_REPLY %RAD_CHECK);
sub post_auth {
$RAD_REPLY{'Tunnel-Medium-Type'} = "IEEE-802";
$RAD_REPLY{'Tunnel-Type'} = "VLAN";
$RAD_REPLY{'Tunnel-Private-Group-Id'} = "666"; # return  vlan
$RAD_REPLY{'Session-Timeout'} = "14400"; # timeout
return RLM_MODULE_UPDATED;
}
```

## eduroam(UK) National RADIUS Proxy Servers (NRPS)

A key element of the eduroam service provided by Jisc is the National RADIUS Proxy Service.

The National RADIUS Proxy Servers (NRPS) are operated by the eduroam(UK) team, part of the Network Access Group in the Jisc e-infrastructure directorate. The NRPSs, together with members' Organisation RADIUS Proxy Servers (ORPSs) form a hierarchical network of RADIUS servers which allow authentication requests to be forwarded between organisations which are members of the eduroam(UK) federation.

Before an organisation's RADIUS server can become part of the eduroam RADIUS hierarchy, the organisation must join the eduroam(UK) federation. A brief form must be completed and submitted by the network management contact at the organisation seeking to join. Before submitting this application, it is essential that the eduroam(UK) Technical Specification and eduroam(UK) Policy are read to ensure they contain no insurmountable obstacles to participation.

How to join is described on: https://community.jisc.ac.uk/library/janet-services-documentation/how-does-organisation-join-service [7] Applications for membership result in a ticket being generated in the Jisc Service Desk system and are processed by the support team. Member organisation details are automatically entered into the eduroam(UK) support database to enable registration and peering of ORPSs and for further diagnostic functions to be supported.

Following application, the services of Jisc eduroam(UK) Technical Support will become available to assist with the implementation process. On completion of the required infrastructure eduroam(UK) Technical Support will seek confirmation of compliance with the service requirements and then complete the peering process between the new ORPS and the national infrastructure. Final confirmation tests can then be undertaken using the automated tools on the support website.

This section describes the key changes to RADIUS configuration required to convert a simple RADIUS AAA server into a platform for delivery of JRS Tier 2 (or Tier 3 if you have WPA2, IPv6 and no NAT end points).

Once you have 802.1X authentication operating there are only a couple of changes to make to enable the authentication of unknown users via their home RADIUS servers, to proxy 802.1X AAA. There are several architectural AAA decisions to make, such as: do you have a set of front end RADIUS servers that take all AAA requests and then proxy everything – your home users to backend AAA systems and unknown users to the NRPS? Or do you have a single farm of RADIUS servers that perform all local authentication themselves and anything else gets handed off to the NRPS? Usually these decisions are decided by: the security of the main AAA servers, load balancing requirements, local network architecture and IT skills of the site staff. However, it is recommended that only systems that should talk to the RADIUS servers can talk to the RADIUS servers. This can be achieved by local firewalling on the required ports. In the case of FreeRADIUS, this is UDP ports 1812 – 1814.

This section, like the main 802.1X configuration section, uses FreeRADIUS for the examples. Both version 1 and version 2 examples are provided as the proxy code has been massively changed between the two versions, unlike the EAP configuration. The other RADIUS servers use the same general terms and methods used within these examples.

**Clients**

The first step is to define the NRPS as clients within your FreeRADIUS configuration. This allows requests from other sites that are being proxied by the NRPS systems to be accepted at your ORPS.

In FreeRADIUS, either add each NRPS address to clients.conf or the SQL-based naslist table. If using naslist in SQL the required secrets for each NRPS are supplied for each ORPS as they are registered in the Janet eduroam support server system.

**Proxy**

Now that the NRPS can send packets to your ORPS you should be able to run a ‘remote test’ via the Janet eduroam support site to check that remote sites can authenticate your visiting users. The next step is to reciprocate this by allowing their users to authenticate whilst on your network. Here is where we configure the proxy settings to enable their AAA requests to be passed back to the remote ORPS that will check and test their credentials.

In FreeRADIUS 1.x insert the following into proxy.conf – supplying the appropriate secret and authhost/accthost DNS entry:

```
realm DEFAULT {

type   = radius

authhost = roamingX.ja.net:1812

accthost = roamingX.ja.net:1813

secret   = secret

nostrip

}
```

For FreeRADIUS 2.x we define a pool for an external realm:

```
realm jrs {

pool = jrs

}

jrs_config {

server0 = roaming0.ja.net

server1 = roaming1.ja.net

server2 = roaming2.ja.net

secret0 = secret
```

```
secret1 = secret

secret2 = secret

}

server_pool jrs {

   home_server = jrs0

home_server = jrs1

home_server = jrs2

}

home_server jrs0 {

ipaddr = ${jrs_config.server0}

secret = ${jrs_config.secret0}

port = 1812

type = auth+acct

nostrip

}

home_server jrs1 {

ipaddr = ${jrs_config.server1}

secret = ${jrs_config.secret1}

port = 1812

type = auth+acct

nostrip

}

home_server jrs2 {

ipaddr = ${jrs_config.server2}

secret = ${jrs_config.secret2}
```

```
port = 1812

type = auth+acct

nostrip

}
```

In FreeRADIUS 1.x the realm engine will tag a request as being NULL, LOCAL or DEFAULT depending on the string value of the realm. For example: username = NULL, username@site.ac.uk [8] REALM is site.ac.uk which, if not defined in proxy.conf will be treated as DEFAULT rather than LOCAL.

In FreeRADIUS 2.x it is best to use ‘unlang’ to configure the realm. In the sites-enabled/default main config within the authorize stanza, straight after the preprocess/suffix section and before the authorization methods insert the following realm creation engine.

```
if"%{User-Name}" =~ /\\\\?([^@\\\\]+)@?([-[:alnum:]._]*)?$/) {

if(!"%{2}" || ("%{2}" == "yourrealm.ac.uk") || ("%{2}" == "yourotherrealm.ac.uk") ){

  update request {

   #Stripped-User-Name := "%{1}"  # you may need this line if handling names
elsewhere

Realm := "local"

  }

}

else{

  update request {

   #Stripped-User-Name := "%{1}@%{2}" # you may need this line if handling
names elsewhere

Realm := "%{2}"

   }

}

}

# Username in invalid format (this may just be someone doing eap-tls)

# set the realm to local, else this will end up going to the default home server.
```

```
else{
update request {
  Stripped-User-Name := "anonymous"
Realm := "local"
}
}
# PROXYING LOGIC
# Eventually if we ever need to proxy to multiple locations we
# can do checks here, but for now assume all non local realms go through JRS
switch "%{Realm}" {
case "local" {
  # Don't do any proxy stuff here, request will be handled later.
}
case "yourrealm.ac.uk" {
case {
  update control {
   Proxy-To-Realm := "jrs"
  }
  update request {
   Realm := "jrs"
  }
}
}
```

The ‘unlang’ feature is far more flexible for the addition of further proxy/realm rules and handling packets in special ways.

Now, a visitor using a proper full outer id username@theirrealm.ac.uk [9] will have their AAA 802.1X proxied through the NRPS which will then proxy it through to their ORPS for handling.

Note: A visitor will not need to have your RADIUS certificate installed onto their system, as their main EAP is passed all the way back to their ORPS, the endpoint of the conversation being their ORPS. Likewise, your users will not need to install the certificates of the remote site before they can use the remote EAP network; they simply need their home site RADIUS certificates and CA signing certificate. Also note that EAP packets can be large, and can often be fragmented so any firewall must allow UDP fragments between the ORPS and NRPS. This is especially true for EAP-TLS authentication.

## Filtering of Attributes

Once you proxy an 802.1X AAA method, the remote RADIUS server is likely to send back more information than you desire, such as VLAN return attributes; therefore you should always ensure that you filter the return packets. In FreeRADIUS this is achieved by using the attr_filter option. This should not be run on pre-proxy packets; however it should be run in the post_proxy section.

There are a minimal list of attributes that must not be filtered (otherwise EAP breaks as proxy-state, EAP flags and MPPE flags must all be preserved in the conversation); an up to date list is maintained on the Janet eduroam support server in the FAQ section.

In FreeRADIUS 1.x add attr_filter to the authorize and post-proxy stanzas in radiusd.conf. In FreeRADIUS 2.x, add the attr_filter.post-proxy to the post-proxy section in your active servers in sites-enabled.

attrs file in both cases should contain:

```
    DEFAULT

    Service-Type == Framed-User,

    Service-Type == Login-User,

    Login-Service == Telnet,

    Login-Service == Rlogin,

    Login-Service == TCP-Clear,

    Login-TCP-Port <= 65536,

    Framed-IP-Address == 255.255.255.254,

    Framed-IP-Netmask == 255.255.255.255,

    Framed-Protocol == PPP,

    Framed-Protocol == SLIP,

    Framed-Compression == Van-Jacobson-TCP-IP,
    Framed-MTU >= 576,
    Framed-Filter-ID =* ANY,
    Reply-Message =* ANY,
    Proxy-State =* ANY,
    EAP-Message =* ANY,
    MS-MPPE-Recv-Key =* ANY,
    MS-MPPE-Send-Key =* ANY,
    MS-CHAP-MPPE-Keys =* ANY,
    State =* ANY,
    Message-Authenticator =* ANY,
    Session-Timeout <= 28800,
    Idle-Timeout <= 600,
    Port-Limit <= 2
```

It is left to individual site policy what other attributes are filtered. Sites may require other REALM identities to be added to this file and allow such realms to be able to send specific attributes through the proxy. For example Tunnel-Medium-Type or Tunnel-Private-Group-Id filtering can be undertaken to ensure that the values are only what you expect: e.g. TunnelPrivate-Group-Id == 666.

### Failover and Load Balancing

Whilst the load-balancing behaviour of FreeRADIUS is improving, you cannot load balance EAP such that parts of the conversation go through different NRPS. The conversation must pass through a single NRPS; it is therefore recommended to configure the NRPS as fail_over.

In FreeRADIUS 1.x this is using the ldflag; in FreeRADIUS 2.x you define type = fail_over in the pool section. See the config inline documentation in both cases.

RADIATOR has some load-balancing methods which are able to keep each EAP conversation to a single NRPS. If a site is using RADIATOR it is worth investigating this method further.

---

**Source URL:** https://community.jisc.ac.uk/library/advisory-services/server-configuration

**Links**
[1] http://www.freeradius.org/
[2] http://support.microsoft.com/kb/814394/en-us
[3] http://support.microsoft.com/kb/885453/en-us
[4] http://wiki.freeradius.org/FreeRADIUS_Active_Directory_Integration_HOWTO
[5] http://wiki.freeradius.org/Main_Page

[6] http://www.freeradius.org/list/
[7] https://community.jisc.ac.uk/library/janet-services-documentation/how-does-organisation-join-service
[8] mailto:username@site.ac.uk
[9] mailto:username@theirrealm.ac.uk