Deliverable DJ5.1.5,1: Roaming cookbook (1st version)

Deliverable 5.1.5,1

Contractual Date: 31/05/06
Actual Date: ?
Contract Number: 511082
Instrument type: Integrated Infrastructure Initiative (I3)
Activity: JRA5
Work Item: 1 - Roaming
Nature of Deliverable: R (Report)
Dissemination Level: PU (Public)
Lead Partner: SURFnet
Document Code: GN2-06-105v2
Authors: K. Wierenga (SURFnet, main author), P. Dekkers (SURFnet), L. Guido (FCCN), T. Kersting (DFN), S. Papageorgiou (NTUA/GRNET), Janos Mohacsi (NIIF/HUNGARNET), R. Papez (ARNES), M. Milinovic (CARNet/Srce), D. Penezic (CARNet/Srce), J. Rauschenbach (DFN), Ch. Schwartz (RESTENA), S. Winter (RESTENA), T. Wolniewicz (Nicolaus Copernicus University, Torun), JRA5 group

Abstract
This Deliverable provides a collection of guidelines and installation instructions ("cookbook") for implementing the eduroam service.
Inter-NREN roaming: Cookbook

Document Revision History

<This page to be deleted before submission to the EC>

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description of change</th>
<th>Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>06-07-06</td>
<td>First draft issued</td>
<td>K. Wierenga</td>
</tr>
<tr>
<td>2</td>
<td>10-07-06</td>
<td>Configuration examples added</td>
<td>P. Dekkers, L. Guido, S. Winter, S. Papageorgiou</td>
</tr>
<tr>
<td>3</td>
<td>02-08-06</td>
<td>Adaptation to JRA5</td>
<td>K. Wierenga</td>
</tr>
<tr>
<td>4</td>
<td>24-08-06</td>
<td>Edited reference setup</td>
<td>K. Wierenga</td>
</tr>
<tr>
<td>5</td>
<td>01-11-06</td>
<td>Integration of new contributions and edit</td>
<td>T. Kersting</td>
</tr>
<tr>
<td>6</td>
<td>09-11-06</td>
<td>Edit</td>
<td>K. Wierenga Review</td>
</tr>
<tr>
<td>7</td>
<td>15-03-07</td>
<td>Updated freeradius configuration</td>
<td>R. Papež</td>
</tr>
</tbody>
</table>

REVIEW

<table>
<thead>
<tr>
<th>Summary of suggested changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main reviewer</td>
</tr>
</tbody>
</table>

Recommendation

1) Major revision

Re-submitted for review - if 1) DD/MM/YY

Final comments

Approved:

1 Deliverable must be changed and reviewed again before submission to the EC can be considered
2 Deliverable may be submitted to the EC after the author has made changes to take into account reviewers' comments as appropriate
3 For submission to EC
# Table of Contents

0 Executive Summary ........................................................................................................... v

1 Introduction ....................................................................................................................... vi

2 eduroam in a nutshell ........................................................................................................ 2
   2.1 General overview ........................................................................................................ 2
   2.2 Elements of the eduroam infrastructure .................................................................. 3
       International top-level RADIUS-server (TLR) ....................................................... 3
       National TLR ............................................................................................................... 3
       Institutional RADIUS ............................................................................................... 3
       Identity Management System .................................................................................... 4
       Switches .................................................................................................................... 4
       Access Points ............................................................................................................ 5
       Supplicants ............................................................................................................... 5

3 Example eduroam Setup .................................................................................................. 5
   3.1 Reference National RADIUS Proxy ......................................................................... 5
       3.1.1 Common configuration ..................................................................................... 5
       3.1.2 Client definition ............................................................................................... 6
       3.1.3 Handling unknown realms under the own TLD .............................................. 7
       3.1.4 Handling empty realms ................................................................................... 7
       3.1.5 Proxying to an organisation with one RADIUS Server .................................. 7
       3.1.6 Proxying to multiple RADIUS servers ......................................................... 8
   3.2 Reference Campus Setup .......................................................................................... 10
       3.2.1 Introduction ...................................................................................................... 10
       3.2.2 Configuring the switch for eduroam ............................................................... 12
       3.2.3 Setting up the RADIUS-server ....................................................................... 13
       3.2.4 Configuring the Access Point for eduroam ................................................. 18
       3.2.5 Supplicant ....................................................................................................... 21
# Table of Figures

Figure 1: Layers of the eduroam RADIUS hierarchy ........................................... 3

Figure 2: Message flow in RADIUS-server ......................................................... 4

Figure 3: Network Topology ........................................................................ 11
Executive Summary

There are plenty of options to build an eduroam infrastructure, especially at the campus level. Devices and software solutions both of commercial vendors as well as open source products are available. This deliverable presents the various elements needed by a typical institution to participate in the eduroam network and provides hands-on guidelines to configure those in a nutshell. It is intended as a technical guideline “cookbook” to help administrators of institutions willing to join the eduroam network. It is not meant to describe eduroam and the underlying architecture in detail, these descriptions can be found in the deliverable DJ5.1.4 (“Inter-NREN Roaming Architecture: Description and Development Items”).

The deliverable describes the main functional principle and introduces the components needed in an eduroam infrastructure in the general case. A reference campus set-up based on products frequently used in the research and education area is provided as an example with detailed configuration and installation instructions. However, this is not a recommendation which technical equipment to buy or to use. Other solutions might provide the same or at least the same level of functionality. The bigger part of the document is therefore dedicated to the configuration of alternative components, for a better readability these instructions are provided as an appendix to the deliverable. It must be added that this again, is not an exhaustive list. The products described are used by one or more JRAS5 participants to ensure the availability of a certain level of practical experience.

This is the first version of the “cookbook”. A second version will follow and provide an option to integrate new developments both in the eduroam architecture and in the spectrum of products and solutions considered.
1 Introduction

The current eduroam architecture was described in more detail in the deliverable "Inter-NREN Roaming Architecture: Description and Development Items" [DJ5.1.4]. It did not address the question how this more general depicted architecture including the components could be implemented in a real environment. This "cookbook" document is addressing exactly this question. The purpose is, however, not to promote any product, but to select popular solutions with a broader usage rate in the research and education area as examples to describe the installation procedures. The idea is to enable the campus administrator or the eduroam operator to adapt these configuration principles for the devices operated in the home environment.

The first section (chapter 2) provides an overview of the necessary, basic elements needed as well as their specific functions within the eduroam network, whereas the second section (chapter 3) shows the configuration of a typical example setup ranging from national RADIUS servers down to user supplicants. Configuration examples for multiple variations of the elements described are shown in the Appendices A to C, covering RADIUS servers, switches, access points and supplicants of different vendors or available implementations as open source solutions.

The list of products for which guidelines are provided is not exhaustive and might also depend on the version of the software used. Changes in the context of technical progress are unavoidable, so that the deliverable has an inherent dynamic aspect and the reader should be aware of this. A second version of the "cookbook" is part of the project plan and will provide an option to reflect new developments and necessary corrections to the material provided in version 1 as well.
2 eduroam in a nutshell

2.1 General overview

Please refer to deliverable DJ5.1.4 “Inter-NREN Roaming Architecture: Description and Development Items” for an in-depth description of eduroam and the underlying architecture.

Eduroam stands for EDUcation ROAMing, it enables users from one academic institution participating in eduroam to securely connect to the Internet at another participant.

The eduroam architecture making this possible is based on a number of technologies and agreements which together provide the eduroam user experience: open your laptop and be online.

The crucial agreement laying the foundation of eduroam is that the authentication of a user is done at his home institution using their specific authentication method, whereas the authorisation decision allowing access to the network resources upon proper authentication is done by the owner of the visited network.

In order to transport the authentication information of a user from the visited institution to his home institution (and the authentication response back) a hierarchical system of RADIUS-servers is created. Typically every institution deploys a RADIUS-server which is connected to a local user database, this RADIUS-server is connected to a central national RADIUS-server which in turn is connected to a European (or global) RADIUS-server. Because users are using usernames of the format ‘user@realm, where realm is of the form institution.tld (tld=country code top-level domain) the RADIUS-servers can use this information to route the request to the appropriate next hop in the hierarchy, until the home institution is reached. An example of the RADIUS hierarchy is shown in figure 1.

To transfer the users authentication information securely across the RADIUS-infrastructure to his home institution and to prevent other users from hijacking the connection after successful authentication, the access points or switches are using the IEEE 802.1X standard which encompasses the use of EAP, the Extensible Authentication Protocol. Using the right EAP-method either a secure tunnel will be set up from the users computer to his home institution through which the actual authentication information (username/password etc.) will be carried (EAP-TTLS or PEAP) or mutual authentication by public X.509 certificates, otherwise not vulnerable to eavesdropping, will be used (EAP-TLS).

After successful authentication by the home institution and authorisation by the visiting institution, this visited institution grants the user network access, possibly by placing the user in a specific VLAN intended for guests.

In the next chapter the various elements of this architecture and their function will be shortly described.
2.2 Elements of the eduroam infrastructure

International top-level RADIUS-server (TLR)

The international top-level RADIUS-servers, at the time of writing located in the Netherlands and Denmark for the European confederation and Australia and Hongkong for the Asian and Pacific region each have a list of connected country domains (.nl, .dk, .au, .cn etc.) serving the appropriate NRENs. They accept requests for country domains for which they are authoritative and subsequently forward them to the associated RADIUS-server for that country (and transport the result of the authentication request back). Requests for country domains they are not authoritative for are forwarded to the proper international TLR.

National TLR

A national RADIUS-server has a list of connected institutional servers and the associated realm. It receives requests from the international servers and institutions it is connected to and forwards them to the proper institution or in case of a request for an international destination to an international server.

Institutional RADIUS

The Institutional RADIUS-server is responsible for authenticating its own users (at home or visiting another institution) by checking the credentials against a local identity management system and for forwarding requests from visiting users to the national RADIUS-server. Upon proper authentication of a user the institutional RADIUS-server may assign a VLAN to the user.
Note that the institutional RADIUS-server is the most complicated one, whereas the other RADIUS-servers merely proxy requests, the institutional server also needs to terminate the requests, and therefore needs to be able to terminate EAP requests and perform identity management system lookups.

The message flow inside an institutional RADIUS-server can be depicted as follows:

![Message flow in RADIUS-server](image)

### Identity Management System

The Identity Management System contains the information of the endusers, for instance usernames and passwords. They must be kept up-to-date by the responsible institution.

### Switches
Switches need to be able to forward access requests coming from a supplicant (see below) to the institutional RADIUS-server, to grant network access upon proper authentication and to possibly assign users to specific VLANs based on information received from the RADIUS-server.

**Access Points**

Access Points need to be 802.1X capable and able to forward access requests coming from a supplicant (see below) to the institutional RADIUS-server, to give network access upon proper authentication and to possibly assign users to specific VLANs based on information received from the RADIUS-server. Furthermore Access Points exchange keying material with client systems to prevent session hijacking.

**Supplicants**

A supplicant is a piece of software (built-in or separate) that ‘talks’ 802.1X and that sends authentication information using EAP.

---

### 3 Example eduroam Setup

The following sections provide an example setup of the eduroam network from top to bottom. At first the configuration for a national RADIUS Server is presented, followed by the various components needed at an institutional level right down to the supplicants of the end-users.

#### 3.1 Reference National RADIUS Proxy


Examples in this chapter can be used as a repository of configuration snippets for building complex proxy servers, or when used as is, for simple proxy relations between one organization and two top-level servers.

Radiator expects the configuration in file `/etc/radiator/radius.cfg`

#### 3.1.1 Common configuration

- `LogDir` /var/log/radiator
- `DbDir` /usr/share/radiator
LogDir defines the directory in which start-up logs and PID file reside, DbDir defines the path to Radiator's data files such as dictionaries.

Trace 3 logs will be sent to the system syslog, and Trace 4 logs will be stored in the directory /var/log/arch/radiator/

The log files will be split on a daily basis.

<Log SYSLOG>
  Facility local7
  LogIdent log-syslog
  Trace 3
</Log>

<Log FILE>
  Filename /var/log/arch/radiator/radiator.%Y_%m_%d.log
  LogIdent log-file
  Trace 4
</Log>

Optionally: SNMP allows remote monitoring of activity on a RADIUS server with tools such as RADAR from OSC (http://www.open.com.au/radar/index.html) or drawing simple graphs of activity by rgraph from CESNET (http://www.eduroam.cz/rgraph/).

<SNMPAgent>
  ROCommunity xxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  Managers localhost 127.0.0.1
</SNMPAgent>

The ports on which Radiator will listen for Authentication and Accounting requests need to be defined. The port numbers 1812 and 1813 were assigned to the RADIUS protocol by IANA, unfortunately some RADIUS servers (such as CiscoACS 3.x) are still using old the numbers 1645, 1646. For this reason it is suggested to use both numbers.

AuthPort  1645,1812
AcctPort   1646,1813

3.1.2 Client definition

In the client section all possible peers (the "institutional" and the confederation RADIUS servers) have to be listed and a secret has to be assigned to them. As this secret is the only thing which protects the communication between the RADIUS servers from eavesdropping, it has to be cryptographically strong and well protected.

<Client localhost>
  Secret    mysecret
  DupInterval 0
</Client>

<Client radius.orgA.tld>
  Secret    xxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  Identifier radius.orgA.tld
</Client>
<Client etlr1.eduroam.org>
  Secret  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  Identifier etlr1.eduroam.org
</Client>

<Client etlr2.eduroam.org>
  Secret  xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  Identifier etlr2.eduroam.org
</Client>

It is necessary to mark each host by a unique identifier, it is later used to prevent loops in the hierarchy. Etlr1 and etlr2 are the European toplevel RADIUS servers.

3.1.3 Handling unknown realms under the own TLD

Known realms under the TLD for which the RADIUS server is responsible are explicitly listed, so if the RADIUS server cannot match an Access-Request under his TLD to an entry in the Handler section, the realm is considered unknown. Any Access-Request with unknown realm under the TLD for which this RADIUS server is responsible is being rejected with an Access-Reject message. Possible Accounting requests are acknowledged and then dropped.

<Handler Realm=/>.tld$/i>
  AccountingHandled
</Handler>

3.1.4 Handling empty realms

Theoretically empty realms should not reach a National RADIUS Proxy, but if they do, this code will prevent the National RADIUS Proxy from sending these empty realms to the international top-level proxy servers.

<Handler Realm=/>.$/>
  AccountingHandled
</Handler>

3.1.5 Proxying to an organisation with one RADIUS Server

In the case in which an organisation has only one server, configuration is relatively easy:

<Handler Realm=/^orgA.tld$/i,
  Client-Identifier=/^(?!radius.orgA.tld$)/>
  <AuthBy RADIUS>
    RetryTimeout 3
    Retries 1
    FailureBackoffTime 0
    UseExtendedIds
  </AuthBy>
  <Host radius.orgA.tld>
    AuthPort 1812
  </Host>
Cookbook roaming v1

Realms should be pattern matched case-insensitive, because users sometimes type their realms using upper case letters.

The configuration directive “Client-Identifier=/^(?!radius.orgA.tld)/” is blocking RADIUS packets from radius.orgA.tld with realm orgA.tld from entering this Handler.

If such a packet will be received it will be rejected by the handler matching realm=/>.*/.tld$/i. This prevents loops between the national proxy and the server of orgA.tld.

Servers to proxy to should never be marked as dead, because in doing so, radiator will not try to communicate with them until BackoffTime expires, even if that server is already up again.

### 3.1.6 Proxying to multiple RADIUS servers

Top-level servers are duplicated for better reliability of the eduroam service as required in the policy document [DJ5.1.3,2], also some organisations have multiple RADIUS servers for the same reason. To use multiple servers in eduroam, a special setup has to be used. Standard detection of dead hosts based on timeout will not work here, timeouts might be caused by the infrastructure behind the direct peer of RADIUS server – in this case the server cannot assume that his peer is dead. But in the case it is really dead, a switch to the backup server must be done. Unfortunately the RADIUS protocol does not provide a clear way to do that.

The problem can be solved by using a special configuration developed at CESNET and described in the article “Dead-realm marking feature for Radiator RADIUS servers” ([http://www.eduroam.cz/dead-realm/docs/dead-realm.html](http://www.eduroam.cz/dead-realm/docs/dead-realm.html)).

The idea behind the dead-realm marking is quite simple. Instead of marking the server dead for all realms (as dead-host marking is doing) just the particular realm on the host is marked dead and requests are routed to another configured host.

The variable dr_timeout controls for how long a realm will be marked dead on a particular host. 3600 seconds is the suggested value. Much lower values will cause the system to check the dead primary server too often. Higher values can be considered if the backup server is equally powerful as the primary server.

```
DefineFormattedGlobalVar dr_timeout 3600
```

Following the two AuthBy sections are definitions of the servers which will be used as a pool for destinations defined later. NoReplyHook and ReplyHook are necessary for the dead-realm marking feature, a Perl implementation is available online at [http://www.eduroam.cz/dead-realm/](http://www.eduroam.cz/dead-realm/).

NoReplyHook is called when no reply is received from the RADIUS server after 2 attempts within 6 seconds. In this case the processed packet is ignored, the realm is marked as dead on this server and when a subsequent packet with the same realm is received it will be forwarded to the other defined server.

ReplyHook is called when a response from the server is received, in case the realm is marked dead the realm will be unmarked dead.

```
<AuthBy RADIUS>
   Identifier etlr1.eduroam.org
   RetryTimeout 3
</AuthBy>
```
Retries 1
FailureBackoffTime 0

UseExtendedIds

نكوت etlr1.eduroam.org>:
AuthPort 1812
AcctPort 1813
Secret xxxxxxxxxxxxxxxxxxxxxxxxxx
</Host>

NoReplyHook file:="/etc/radiator/dr_no-reply-hook.pl"
ReplyHook file:="/etc/radiator/dr_reply-hook.pl"
</AuthBy>

<AuthBy RADIUS>
Identifier etlr2.eduroam.org

RetryTimeout 3
Retries 1
FailureBackoffTime 0

UseExtendedIds

نكوت etlr2.eduroam.org>:
AuthPort 1812
AcctPort 1813
Secret xxxxxxxxxxxxxxxxxxxxxxxxxx
</Host>

NoReplyHook file:="/etc/radiator/dr_no-reply-hook.pl"
ReplyHook file:="/etc/radiator/dr_reply-hook.pl"
</AuthBy>

The following directive (DefineFormattedGlobalVar) defines the variable dr_TOLEVEL_server_list and assigns a list of servers responsible for the handler marked as TOLEVEL to this variable.

DefineFormattedGlobalVar dr_TOLEVEL_server_list etlr1.eduroam.org,etlr2.eduroam.org

The handler with the filter Realm="/^.+$/ will receive any request which will not be caught by more specific filters before (Realm="/^orgA.tld$/i, Realm="/.*\tld$/", Realm="/^$/"). In this example this will be all requests which belong to other TLDs than the ones defined on this server – requests which have to be forwarded to top-level servers.

RequestHook will use the value of Identifier (= TOLEVEL) defined in this handler to search the variable dr_TOLEVEL_server_list holding the list of servers. It will forward the request to a server not marked dead or if all are marked dead the one with the oldest dead-realm mark.

<Handler Realm="/^.+$/">
  Client-Identifier="/^(?!etlr1.eduroam.org$)/">
  Client-Identifier="/^(?!etlr2.eduroam.org$)/">

  Identifier TOLEVEL

</AuthBy INTERNAL>
RequestHook file="/etc/radiator/dr_choose-server.pl"
3.2 Reference Campus Setup

3.2.1 Introduction

Campus networks vary widely, both in topology and used equipment, software etc. In order to assist a campus administrator in setting up eduroam on their campus we present the implementation of a documented typical setup. It is hoped that this will allow even users of different topologies and/or equipment to understand the necessary steps to take. Furthermore, in the appendices the same setup will be provided integrally, as well as worked out for a number of other common types of equipment and software. Lastly, it is the intention to provide these and future example configurations at the website http://www.eduroam.org.

For the reference network we use a typical set of network equipment consisting of:

- a Cisco Catalyst 3550 (or similar) switch,
- a Cisco Aironet AP-1200 Access Point,
- a laptop with Windows XP and
- a Radiator RADIUS-server

The network topology is as follows:
In this setup wireless users are separated in different VLANs: VLAN906 for ‘administrative’ users and VLAN909 for normal eduroam users. The next table describes each VLAN used in this document:

<table>
<thead>
<tr>
<th>VLAN ID</th>
<th>Propose</th>
</tr>
</thead>
<tbody>
<tr>
<td>901</td>
<td>VLAN for internet access – access to core routers</td>
</tr>
<tr>
<td>902</td>
<td>The Administrative VLAN of the hotspot (AP’s; RADIUS; etc.)</td>
</tr>
<tr>
<td>903</td>
<td>VLAN with open SSID for giving information about the institute</td>
</tr>
<tr>
<td>906</td>
<td>VLAN reserved for administrative users</td>
</tr>
<tr>
<td>909</td>
<td>VLAN reserved for ‘normal’ eduroam users</td>
</tr>
</tbody>
</table>

Table 1: VLAN description
The next table describes the IP configuration for the router sub-interfaces and what networks are configured for each VLAN:

<table>
<thead>
<tr>
<th>Interface</th>
<th>802.1Q Tag</th>
<th>Interface IP Address</th>
<th>DHCP Pool</th>
<th>What is accessible in this network</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE0.901</td>
<td>901</td>
<td>Some public IP address</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>FE0.902</td>
<td>902</td>
<td>192.168.10.254</td>
<td>N/A</td>
<td>AP’s; RADIUS Server</td>
</tr>
<tr>
<td>FE0.906</td>
<td>906</td>
<td>10.9.6.254</td>
<td>10.9.6.0/24</td>
<td>administrators</td>
</tr>
<tr>
<td>FE0.909</td>
<td>909</td>
<td>10.9.9.254</td>
<td>10.9.9.0/24</td>
<td>eduroam clients</td>
</tr>
</tbody>
</table>

Table 2: Router Configuration

### 3.2.2 Configuring the switch for eduroam

In order to gain access to the Internet the configuration of the switch needs to be changed. You have to create a VLAN in which the Access Points will be placed, and provide it with the correct IP-address and gateway information. This can be done with the commands described below.

The next table describes the VLAN associated at each Port of the switch and what equipment will be connected to that specific port.
Table 3: Ethernet Switch Configuration

<table>
<thead>
<tr>
<th>Port</th>
<th>VLAN configuration (T – Tagged; U – Untagged)</th>
<th>What is connected to it</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>U (902)</td>
<td>RADIUS Server</td>
</tr>
<tr>
<td>2-47</td>
<td>U (902) T (909)</td>
<td>Access Points</td>
</tr>
<tr>
<td>48</td>
<td>U (901) T (902; 909)</td>
<td>Central Switch</td>
</tr>
</tbody>
</table>

First we will configure the port where the RADIUS Server will be connected and put it on the Administrative VLAN:

```plaintext
switch(config)#interface range fastethernet0/1
switch(config-if)#description RADIUS Server
switch(config-if)#switchport mode access
switch(config-if)#switchport access vlan 902
switch(config-if)#spanning-tree portfast
```

Then configure all switch-ports that will connect Access Points for the VLAN's that users and Access Points can have access to (in trunk mode) – at least the administrative VLAN and the VLAN where authenticated users will be placed:

```plaintext
switch(config)#interface fastethernet0/2 - 47
switch(config-if)#description eduroam Access Points
switch(config-if)#switchport trunk encapsulation dot1q
switch(config-if)#switchport trunk native vlan 902
switch(config-if)#switchport trunk allowed vlan 902, 909
switch(config-if)#switchport mode trunk
```

The uplink can be defined with:

```plaintext
switch(config)#interface fastethernet0/48
switch(config-if)#switchport trunk encapsulation dot1q
switch(config-if)#switchport trunk native vlan 901
switch(config-if)#switchport trunk allowed vlan 901, 902, 909
switch(config-if)#switchport mode trunk
```

### 3.2.3 Setting up the RADIUS-server

Now the RADIUS server will be configured.

Because of the EAP authentication within RADIUS, a (small) PKI is required. If there is no PKI available one could create the required key and certificate with for instance TinyCA. TinyCA (http://tinyca.sm-zone.net/) is a simple graphical interface on top of OpenSSL. It is possible to use OpenSSL directly (but instructions to do so are outside the scope of this document).
There is also a bootable CD available based on Knoppix that runs TinyCA, the roCA (read-only CA) that can be found at http://www.intrusion-lab.net/roca/

Depending on the EAP-type used, client certificates may also be needed.

Within the Radiator distribution there are also simple scripts available to create certificates for testing purposes.

The Radiator RADIUS-server needs the configuration file /etc/radiator/radius.cfg. This configuration file can be created with the editor of choice, e.g.

\texttt{vi /etc/radiator/radius.cfg}
\texttt{or}
\texttt{pico /etc/radiator/radius.cfg}

In the following examples there are two kinds of EAP which are configured at `institution'. Firstly, EAP-TLS based on client-certificates, secondly, EAP-TTLS and EAP-PEAP that do not require client certificates but traditional mechanisms such as username/password authentication instead.

\textit{Clients}

RADIUS is based on a client-server model. The NAS-devices (Access Points, switches etc.) forward credentials to a RADIUS server but can also act as a client, and therefore need to be defined on the RADIUS server. Other RADIUS servers can act as a client as well, so every kind of RADIUS-request can be forwarded to another server.

The clients are configured within Radiator using the \texttt{<Client>-clause}.

\begin{verbatim}
<Client 192.168.10.200>
  Secret 6.6obaFkm&RNs666
  Identifier ACCESSPOINT1
  IdenticalClients 192.168.10.201
</Client>
\end{verbatim}

In this example there is a client definition for 192.168.10.200, an Access-Point. The secret is a series of (at best 16) characters that are used to encrypt the credentials sent in the RADIUS-request.

It is of course recommended to create a secret that can not be easily guessed since otherwise the RADIUS-message can be decrypted. This is not a big problem with EAP-authentication using 802.1X since the credentials are also transmitted over a SSL-encrypted tunnel between the client and the final authentication server, but especially with regular credentials like the ones used with Web-based redirection authentication this is sensitive information that might be captured, therefore a reasonably complex secret and an SSL tunnel is recommended.

The Identifier in the Client-definition can be used further on in the Radiator-configuration to filter on a specific request.

If there is more then one Client that should use this same secret and identifier definition, the IdenticalClients statement can be used. If there are really many clients with different IP-addresses, there is also the possibility for a “catch-all” client. This is the default-client that is used after all other client-definitions didn’t match. Define this client as:

\begin{verbatim}
<Client DEFAULT>
\end{verbatim}

If this kind of configuration is used, it is worth filtering with firewall-rules on RADIUS packets. There are only a few places where a RADIUS-request should come from: the management VLAN with the NAS-devices (switches and access-points), and the RADIUS infrastructure where unknown requests can be sent to.
Realms and VLAN assignment

The processing of authentication of accounting requests is done by linear processing of the present <Realm>- or <Handler>-clauses in the Radiator configuration file. Handler-clauses are more potent than Realm clauses in terms of filtering things besides realms and are therefore the preferred method. A realm is the part behind a username to indicate the origin of a user. With RADIUS, the username is usually separated from the realm with a “@” so the complete username looks like a regular e-mail address.

A <Handler>-clause is terminated with a </Handler>.

Within a Handler many mechanisms can be configured that define what to do with the RADIUS request.

PROXY example

The most simple Handler is the definition for proxy-ing the request to another server using the “AuthBy RADIUS” definition within this Handler.

In this example a proxy-configuration is shown. First we have a Handler that matches on any request, as long as it does not come from the client with the identifier “Proxy-Identifier”. This is to prevent a proxy loop. When a request comes from a proxy-server, we should never forward it back to that proxy-server.

Another important part is the hostname to forward the request to. Multiple hostnames can be defined here for redundancy reasons: if the first host does not respond within 3 seconds, the second one is tried instead. The UDP-ports to forward the RADIUS-request to can be defined in this “AuthBy RADIUS” mechanism as well.

```
<Handler Client-Identifier=/^(?!Proxy-Identifier$)/>
<AuthBy RADIUS>
    Host 192.87.36.3
    Secret super_secret!
    AuthPort 1812
    AcctPort 1813
    StripFromReply Tunnel-Type, Tunnel-Medium-Type, \
         Tunnel-Private-Group-ID
    AddToReply Tunnel-Type=1:VLAN, Tunnel-Medium-Type=1:Ether_802, \
         Tunnel-Private-Group-ID=1:909
</AuthBy>
</Handler>
```

It is good to know that for a ‘Host’ both the IP-address and FQDN can be used. It’s more or less a personal preference of the RADIUS administrator with one remark that the hostnames are only looked up once at the Radiator (re)start. If the lookup fails, the Host cannot be used until the next restart. This can be tricky with a power outage, where for instance the DNS server is not yet available when Radiator already is.

While using hostnames one benefits from the administrative ease when an IP-address is changed. In that case, you only (or still) need a restart of the RADIUS server.

The last part in this <AuthBy RADIUS>-definition shows the addition of RADIUS-attributes to the RADIUS-response. These attributes can be used to define a VLAN that should be assigned to users that authenticate using this Handler. With StripFromReply the attributes that came from the proxy-server are stripped first to prevent malicious VLAN-assignments, afterwards the attributes are added with the proper values for the local network design. In this case, VLAN 909 is used for guests.
Secure authentication with EAP-TLS

If it is an option to issue end user certificates the EAP-TLS mechanism can be used.

In this example the AuthBy-definition is outside the Handler, and is referred to using the Identifier. (This is useful if the AuthBy-definition is reused in another Handler for instance.)

<AuthBy FILE>
  Identifier ID4-TLS
  Filename %D/TLS-users
  EAPType TLS
  EAPTLS_CAFile %D/cert/institution-ca-chain.pem
  EAPTLS_CertificateFile %D/cert/radius-server-cert.pem
  EAPTLS_CertificateType PEM
  EAPTLS_PrivateKeyFile %D/cert/radius-server-key.pem
  EAPTLS_PrivateKeyPassword (the secret that secures the private-key)
  EAPTLS_MaxFragmentSize 1024
  AutoMPPEKeys
  SSLeayTrace 1
  StripFromReply Tunnel-Type,Tunnel-Medium-Type,Tunnel-Private-Group-ID
  AddToReply Tunnel-Type=1:VLAN,Tunnel-Medium-Type=1:Ether_802,Tunnel-Private-Group-ID=1:909,User-Name=%u
</AuthBy>

In this AuthBy-clause there is a file defined (TLS-users) in which every employee is listed. This way, we control the users that can access the infrastructure using EAP-TLS.

The next definitions define what to do with the EAP-request. First the “EAPType TLS” limits the use of this AuthBy-definition for TLS-only. We do not want regular password authentication here, just certificates. Next, the certificate files are configured and the secret that secures the private-key file can be provided. If there is no secret for the private key, this can be omitted.

The next part defines in what chunks the EAP-messages should be fragmented. Since some parts of the EAP-TLS challenge (which is just a normal Challenge like the ones used for securing a website) are to big to fit in a RADIUS request the packets should be fragmented.

The MPPE-keys (Microsoft Point to Point Encryption, the protocol for encrypting the data across links) portion is important for wireless access. With 802.1x, encryption is done at the edge of the network, between the Access-Point and the client. To provide this secure encryption, a unique key is created and encrypted using the MPPE-keys that are derived from the SSL-challenge. This is something that can both be done at the Access-Point and the Client end so that there is no need to transfer the WEP-key in plain text over the air. This, plus the fact that the key can be rotated within a period defined by either the Access-Point or the RADIUS server provides 802.1x users with a good level of security.

The last part of the AuthBy-definition shows again how to assign a proper VLAN.

<Handler Realm=instituition.cc, EAP-Message=/.+/>  
  AuthBy ID4-TLS
</Handler>

The Handler above shows the referral to the AuthBy-definition and some filtering mechanisms to filter out the proper requests. If more things need to be filtered on, they can be added to this handler, as follows:

NAS-Port-Type=/^Wireless-IEEE-802-11|Ethernet$/

In this way, only requests with the proper NAS-Port-Types are allowed.
For Accounting purposes, a new handler should be defined in this case, that filters on
“Request-Type=Accounting-Request”
since the request does not match the Handler that filters on the EAP-Message.

**EAP-TTLS or EAP-PEAP**

When issuing end user certificates is not an option, the EAP-mechanisms PEAP and TTLS can be used.

These two mechanisms look the same, they both set up an SSL tunnel on which the credentials can be transported. In the case of PEAP this is another EAP-method in an encrypted tunnel, while when using TTLS another RADIUS type is used that is more flexible because this enables the use of PAP-authentication based on a plain-text username and password, transported over a SSL-tunnel. PAP credentials can be verified with almost any backend. EAP-PEAP however uses EAP-MSCHAP for tunneled authentication, and therefore requires a specific backend, like Active Directory.

When there is just a file with plaintext usernames and passwords (for guests for instance) both PEAP and TTLS can be used with the same realm.

Instead of a flat file a more flexible backend for user accounts is a database like MySQL, or LDAP.

```xml
<Handler TunnelledByPEAP=1, Realm=tunnelled.institution.cc>
  <AuthBy FILE>
    Filename %D/peap-users
    EAPType MSCHAP-V2
  </AuthBy>
</Handler>

<Handler TunnelledByTTLS=1, Realm=tunneled.institution.cc>
  <AuthBy FILE>
    Filename %D/ttls-users
  </AuthBy>
</Handler>
```

In these Handlers with the filtering option “TunneledByPEAP” and “TunneledByTTLS” define that the tunneled authentication (with the username and password in it) is handled here.

The “outer authentication”, where the SSL tunnel is set up, looks like the TLS handler.

```xml
<Handler Realm=group_1>
  <AuthBy FILE>
    # the %D/users file can be empty, it’s there for normal PAP
    # authentication. This can however be used for the WEB captive
    # portals.
    Filename %D/users
    EAPType TTLS, PEAP
    EAPTLS_CAFile %D/root.pem
    EAPTLS_CertificateFile %D/server.pem
    EAPTLS_CertificateType PEM
    EAPTLS_PrivateKeyFile %D/server.pem
    EAPTLS_PrivateKeyPassword serverkey
    EAPTLS_MaxFragmentSize 1024
    EAPAnonymous anonymous@group1
    AutoMPPEKeys
  </AuthBy>
</Handler>
```
**General options**

Before all clients and handler definitions there is some space for general configuration parameters for Radiator.

**Trace 4**

LogDir /var/log/radius  
DbDir /etc/radiator  
AuthPort 1812  
AcctPort 1813

Here the location of logfiles (referred in the configuration as %L) and databases (%D) can be defined as well as the amount of log output is given (with Trace).

Also, there is an important section with “AuthPort” and “AcctPort”. These parameters define the ports used by the RADIUS server. The most common ports are 1812 for authentication and 1813 for accounting, however Radiator still uses the ports 1645 and 1646 as default.

### 3.2.4 Configuring the Access Point for eduroam

**Cisco AP 1200 Series (802.11g Radio)**

The configuration examples used in this document were tested and made on a Cisco Series 1200 with a 802.11g Radio Module and with the following Cisco software:

**IOS Version:**
Cisco IOS Software, C1200 Software (C1200-K9W7-M), Version 12.3(8)JA2, RELEASE SOFTWARE (fc1)

**Bootloader:**
C1200 Boot Loader (C1200-BOOT-M) Version 12.2(8)JA, EARLY DEPLOYMENT RELEASE SOFTWARE (fc1)

**The multiple (dynamic) VLAN assignment**

If multiple VLANs are configured on the Cisco AP, it is mandatory to associate each SSID to one VLAN otherwise the Access Point will not activate the SSID’s without VLAN. It is possible however, to put different users which are connected to the same SSID e.g. eduroam - on different VLANs based for instance on the user profile. To activate this feature it is necessary to enter ‘aaa authorization network default group radiusgroup’ in the Access Point's configuration. The AP will then give priority to the VLANs returned by RADIUS over the ones statically associated with the SSID. This feature is often called dynamic VLAN assignment.

Cisco's Access Points require that 2 virtual interfaces for each VLAN (a radio and an Ethernet port interface) are configured. If e.g. 4 VLANs are to be used for eduroam users (for students, admin staff, teachers and visiting eduroam users from other institutions for example) then it is necessary to define one Dot11Radio0.vlanID, and one FastEthernet0.vlanID, and ensure that both have the same encapsulation dot1Q vlanID and the same bridge-group for each VLAN.
Two commands that are also needed are: ‘encryption vlan vlanID mode ciphers aes-ccm tkip wep128’ and ‘broadcast-key vlan vlanID change 600 membership-termination capability-change’ otherwise the access point will not change the user to the received VLAN.

In the example configuration below will be no dynamic VLAN assignment.

The eduroam SSID – logical setup

In the following example the eduroam SSID is configured to accept all cipher modes used today on eduroam enabled institutions: 802.1X/WEP; WPA/TKIP; WPA2/AES.

Cisco Aironet 1200 Series basic configuration

First some basic eduroam configuration parameters for the Cisco AiroNet Series 1200 (similar for other Cisco Access Points Series).

Setting the Name and IP address

First an IP address to the BVI interface (the IP address that this Access Point will have for accessing resources like the RADIUS server) needs to be configured. Also a unique name for this Access Point (ap1200) will be configured.

```
ap#configure terminal
ap1200(config)#hostname ap1200
ap1200(config)#interface BVI 1
ap1200(config-if)# ip address 192.168.10.200 255.255.255.0
```

RADIUS/AAA section

In the authentication, authorisation and accounting configuration parameters (AAA) at least one group radsrv needs to be defined (radsrv), that will be assigned later for the several AAA operations. More groups can be defined if needed used for various purposes – one for authentication, another for accounting, etc. In this example the RADIUS server has the IP address 192.168.10.253.

```
ap1200(config)#aaa new-model
ap1200(config)#radius-server host 192.168.10.253 auth-port 1812 acct-port 1813 key <secret>
ap1200(config)#aaa group server radius radsrv
ap1200(config-ag-radius)#server 192.168.10.253 auth-port 1812 acct-port 1813
ap1200(config-ag-radius)#!
ap1200(config-ag-radius)#aaa authentication login eap_methods group radsrv
ap1200(config)#aaa authorization network default group radsrv
ap1200(config)#aaa accounting session-duration ntp-adjusted
ap1200(config)#aaa accounting update newinfo periodic 15
ap1200(config)#aaa accounting network default start-stop group radsrv
ap1200(config)#aaa accounting network acct_methods start-stop group radsrv
```

Configuring the SSID’s

For each SSID one dot11 ssid <SSID NAME> must be configured. In this section the default VLAN for the SSID will be configured as well as the authentication framework, the accounting and, if desired, the SSID to be broadcasted (guest-mode).
ap1200(config)#dot11 ssid eduroam
ap1200(config-ssid)#vlan 909
ap1200(config-ssid)#authentication open eap eap_methods
ap1200(config-ssid)#authentication network-eap eap_methods
ap1200(config-ssid)#authentication key-management wpa optional
ap1200(config-ssid)#accounting acct_methods
ap1200(config-ssid)#guest-mode

More SSID's can be configured. An open SSID for giving information about the institution and/or how to connect to the eduroam SSID

ap1200(config)#dot11 ssid guest
ap1200(config-ssid)#vlan 903
ap1200(config-ssid)#authentication open
ap1200(config-ssid)#accounting acct_methods

**The Radio Interface**

Now the configured SSID's will be mapped to the radio interface and it will be specified what ciphers will be used/allowed on each VLAN. If dynamic VLANs are planned the ciphers for those VLANs must also be configured even if there's no direct mapping on any SSID (this example shows the usage of the VLANs 906 and 909 for eduroam users)

ap1200(config)#interface Dot11Radio 0
ap1200(config-if)# encryption vlan 906 mode ciphers aes-ccm tkip wep128
ap1200(config-if)# encryption vlan 909 mode ciphers aes-ccm tkip wep128
ap1200(config-if)#ssid eduroam

To bind extra SSID's the previous command, for each SSID to be bound, needs to be repeated.

The following command sets the maximum time (e.g. 300 seconds, which is recommended) for rekeying/reauthentication:

dot1x reauth-period 300

**VLAN interfaces**

For each VLAN to be used for wireless clients 2 virtual interfaces need to be defined: one on “the air” (DotRadio) and another on the “wire” (FastEthernet) then they need to be bridged together with the same bridge group. This VLANs are always tagged with the proper VLAN identifier.

An administrative VLAN needs to be configured as well (for maintenance/managing and authentication/accounting traffic). This VLAN is usually untagged (the command defining the VLAN has to be suffixed with the keyword “native”) and belongs to bridge-group 1. The Radio virtual interface for this VLAN does not need to be defined since default will keep the physical interface (Dot Radio 0) in bridge-group 1.

Since VLANs can be from 1 to 4094 and bridge-groups from 1 to 255, it is not necessary to have the same bridge-group id as the vlan id.

ap1200(config)#interface fastEthernet 0.902
ap1200(config-subif)#encapsulation dot1Q 902 native
ap1200(config-subif)#bridge-group 1

ap1200(config)#interface dot11Radio 0.903
ap1200(config-subif)#encapsulation dot1Q 903
ap1200(config-subif)#bridge-group 3

ap1200(config)#interface fastEthernet 0.903
ap1200(config-subif)#encapsulation dot1Q 903
3.2.5 **Supplicant**

**Supporting a large user community for EAP-TTLS authentication with preconfigured SecureW2.**

The usage of EAP-TTLS has been considered the easiest way to implement eduroam in a large (especially student) community. MS Windows has no built-in support for EAP-TTLS, but it can be added by installing SecureW2. Installing SecureW2 can pose some security issues, which can be addressed by using a preconfigured distribution.

The main security problems are:

1. Allowing users to set up new connections – this is one of the advanced options of SecureW2: when disabled it will prevent users from carelessly accepting a rogue RADIUS server, and thus will prevent the users from sending their credentials to a fake server; however running SecureW2 this way requires the necessary certificates for the home RADIUS server to be preinstalled;

2. Installation of certificates – SecureW2 requires the certificates necessary to confirm server authenticity to be installed in a specific certificate store; which is different from the default certificate store used by Windows. The result is that even though the user has the main certificate of his/her home institution installed, SecureW2 will not be able to use it properly, thus requiring the possibility to set up new connections.

The proposed procedure is completely secure (under the assumption that users have previously installed their home institution certificate in the standard Windows certificate store). It also vastly simplifies the configuration of SecureW2.

Note: Installation instructions for a single SecureW2 client can be found in the appendices.

In order to prepare the preconfigured exe file, the administrator has to take the following steps:

1. Prepare the SecureW2.INF file
2. Prepare the NSIS configuration file
3. Create the exe file with NSIS
4. Digitally sign the exe file.

The steps for the user are:

```plaintext
ap1200(config-subif)#bridge-group 3
ap1200(config)#interface dot11Radio 0.906
ap1200(config-subif)#encapsulation dot1Q 906
ap1200(config-subif)#bridge-group 6

ap1200(config)#interface fastEthernet 0.906
ap1200(config-subif)#encapsulation dot1Q 906
ap1200(config-subif)#bridge-group 6

ap1200(config)#interface dot11Radio 0.909
ap1200(config-subif)#encapsulation dot1Q 909
ap1200(config-subif)#bridge-group 9

ap1200(config)#interface fastEthernet 0.909
ap1200(config-subif)#encapsulation dot1Q 909
ap1200(config-subif)#bridge-group 9
```
1. Download the preconfigured SecureW2 exe file
2. Confirm the signature of the exe file
3. Start the exe file and enter his/her credentials into the program prompt window
4. Reboot the computer
5. Choose SecureW2 as the authentication method for the eduroam network
6. Connect to eduroam

Detailed instructions of these administrative steps as well as an illustration of the installation process from a user perspective are provided in the following sections.

Preparation the SecureW2 installer

Prerequisites

1. Nullsoft Scriptable Install System (NSIS) – Download NSIS from http://nsis.sourceforge.net/ and install it;
2. Code signing
   a) Download the .NET Framework SDK from Microsoft and install it;
   b) Prepare a code-signing key using openssl
      • Generate the key
        openssl genrsa -des3 -out test_sign.key 1024
      • and certification request
        openssl req -new -key test_sign.key -out test_sign.csr
   c) The CA needs to certify the key for code signing. Using OpenSSL this is done by
      openssl ca -out test_sign.crt -in test_sign.csr -extensions id_kp_codeSigning -extfile codesigning
      • where the codesigning file contains:
        [ id_kp_codeSigning ]
        extendedKeyUsage = 1.3.6.1.5.5.7.3.3
   d) Generate the PFX file
      openssl pkcs12 -export -out test_sign.pfx -in test_sign.crt -inkey test_sign.key -clcerts
   e) Install the PFX in Windows by double-clicking and accepting the default values except for the key protection
      where strong protection should be used

Step 1. Prepare the SecureW2.INF file containing:

```
[Version]
Signature = "$Windows NT$"
Provider = "Alfa & Ariss"
Config = 7

[Certificates]
Certificate.1 = your_ca_cert.der

[WZCSVC]
Enable = AUTO

[SSID.1]
Name = "eduroam"
Profile = "DEFAULT"
[Profile.1]
AuthenticationMethod = PAP
EAPType = 0
Name = "DEFAULT"
Description = "Enter your login credentials:"
UseAlternateIdentity = FALSE
VerifyServerCertificate = TRUE
```

---

Project: GN2
Deliverable Number: DJX.0.1
Date of Issue: 20/03/07
EC Contract No.: 511082
Document Code: GN2-06-105v2

---
PromptUserForCredentials = FALSE
TrustedRootCA.0 = your_ca_cert_fingerprint
UserName = PROMPTUSER

Notes:
1. Under the Certificates section you need to provide the entire certification path starting from the CA directly certifying your RADIUS server and ending with the root CA. Numbering goes from 1 up. As Certificate.0 you may (but do not have to) install your certificate of the RADIUS server.
2. As TrustedRootCA.0 you should provide a fingerprint of one of the CAs in the chain (the most natural choice will be the fingerprint of your organisation's CA)
3. Section WZCSVC causes the automatic start of Window Zero-Configuration service which is needed for SecureW2
4. There are many other options that you may put into the INI file (see SecureW2 administrators guide), but the provided example is sufficient for our configuration. One feature that one might want to add is the use of anonymous outer identity. The configuration above makes the outer identity equal to the real username.

Step 2. Prepare the NSIS configuration directory containing:
1. The configuration file – SecureW2.NSI:

```nsi
!define APPLICATION "SecureW2 installer"
!define VERSION "1.0.0"
!include "MUI.nsh"
;General
;Name and file
Name "${APPLICATION} ${VERSION}"
OutFile "SecureW2_312_Test.exe"
;Interface Settings
!define MUI_ICON "your_icon.ico"
!define MUI_UNICON "your_icon.ico"
!define MUI_ABORTWARNING
Section "${APPLICATION}" SecInstall
SectionIn RO
; Extract all file to the temp dir
SetOutPath $TEMPDIR
; Define all the files required for the installation here:
File "SecureW2_312.exe"
File "SecureW2.INF"
File "your_ca_cert.der"
ExecWait "SecureW2_312.exe"
; If an error occurs then goto Error label else goto Continue label
IfErrors Error
Goto Continue
; Error Label, show error box and then quit
Error:
MessageBox MB_OK|MB_ICONEXCLAMATION "SecureW2 installation problem, please report to ..."
; Continue Label
Continue:
; Remove temporary files
Delete "$TEMPDIR\SecureW2_312.exe"
Delete "$TEMPDIR\SecureW2.INF"
Delete "$TEMPDIR\your_ca_cert.der"
Quit
SectionEnd
```

Note: you should customise the OutFile name, the text in the MessageBox, the icon filename and the certificate filename(s)
2. The original SecureW2 exe file – SecureW2_312.exe
3. The SecureW2.INF file
4. The icon file (your_icon.ico) – it is advisable to have your own Windows icon to mark your customised installer;
5. The certificate file "your_ca_cert.der" (if your certificate chain consists of more files then you need to provide all files listed in the SecureW2.INF file and list them in the SecureW2.NSI file, remember to add them to the Delete section as well);

**Step 3. Prepare your customised installer**
Right click on SecureW2.NSI and chose “Compile NSI script”. If all went well you should now have your exe file. If there were some errors, NSI will report them.

**Step 4. Sign your customised installer**

```plaintext
C:\Program Files\Microsoft.NET\SDK\v2.0\Bin\signtool sign /a your_installer.exe
```

you can also use the GUI to signtool:

```plaintext
C:\Program Files\Microsoft.NET\SDK\v2.0\Bin\signtool signwizard
```

**Installing SecureW2**
After downloading the installer, the user should right-click the filename and check the properties for the correct digital signature. After confirming that the file is genuine, the user should start the installer which immediately starts the SecureW2 installer and the user has to go through a few self-explanatory steps:
Completing the SecureW2 Client 3.1.2 Setup Wizard

Your computer must be restarted in order to complete the installation of SecureW2 Client 3.1.2. Do you want to reboot now?

- [ ] Reboot now
- [ ] I want to manually reboot later
Conclusions

This document aims at giving administrators sufficient information to make it relatively straightforward to implement eduroam. It is obvious that campus architectures vary widely and it is far from us to suggest one specific vendor. However, the document has aimed to provide both a cookbook for a typical setup with equipment that is found at many campuses in the research and education networking community in Europe as well as giving sufficient information for those using different architectures or vendors to understand the necessary steps to become ‘eduroam-enabled’.

To further assist these administrators the appendices contain example configurations for many products as assembled by those in the NREN community in Europe. Furthermore, this document is meant to live on as a living document that will be constantly adapted and augmented.

References


[RFC2865] URL
6  Acronyms

[ARP] [The Address Resolution Protocol (ARP) is the method for finding a host’s hardware address when only its network layer address is known. Due to the overwhelming prevalence of IPv4 and Ethernet, ARP is primarily used to translate IP addresses to Ethernet MAC addresses. It is also used for IP over other LAN technologies, such as Token Ring, FDDI, or IEEE 802.11, and for IP over ATM]

[BVI] [A BVI (Bridge Virtual Interface) is a virtual interface within the campus switch router that acts like a normal routed interface. A BVI does not support bridging, but it actually represents the corresponding bridge group to routed interfaces within the switch router. The interface number is the link between the BVI and the bridge group.]

[Credentials] [A credential is a proof of qualification, competence, or clearance that is attached to a person, and often considered an attribute of that person. In this document it refers to username/password pairs.]

[FQDN] [A fully qualified domain name (or FQDN) is an unambiguous domain name that specifies the node’s position in the DNS tree hierarchy absolutely. To distinguish an FQDN from a regular domain name, a trailing period is added. Ex: somehost.example.com. An FQDN differs from a regular domain name by its absoluteness; a suffix will not be added]

[GUI] [A graphical user interface (or GUI), is a particular case of user interface for interacting with a computer which employs graphical images and widgets in addition to text to represent the information and actions to the user. Usually the actions are performed through direct manipulation of the graphical elements.]

[IANA] [Internet Assigned Numbers Authority. IANA is broadly responsible for the allocation of globally-unique names and numbers that are used in Internet protocols that are published as RFC documents. It maintains a close liaison with the IETF and RFC Editor in fulfilling this function.]

[MPPE] [Microsoft Point-to-Point Encryption (MPPE) is a protocol for encrypting data across Point-to-Point Protocol and Virtual Private Network links. It uses the RSA RC4 encryption algorithm. MPPE supports 40-bit, 56-bit and 128-bit session keys, which are changed frequently to improve security. The exact frequency that the keys are changes is negotiated, but may as frequent as every packet. MPPE alone does not compress or expand data, but the protocol is often used in conjunction with Microsoft Point-to-Point Compression which compresses data across PPP or VPN links.]

[MS-IAS] [Internet Authentication Service (IAS) is the Microsoft Implementation of a Remote Authentication Dial-in User Service (RADIUS) server and proxy. As a RADIUS server, IAS performs centralized connection authentication, authorization, and accounting for many types of network access, including wireless and virtual private network (VPN) connections. As a RADIUS proxy, IAS forwards authentication and accounting messages to other RADIUS servers.]

[.NET] [Microsoft .NET is an umbrella term that applies to a collection of products and technologies from Microsoft. Most have in common a dependence on the Microsoft .NET Framework, a component of the Windows operating system]

[PFX] [pfx (Personal Information eXchange) is a common file extension for X.509 certificates. A .pfx File may contain certificate(s) (public) and private keys (password protected)]

[PID] [The process identifier (normally referred to as the process ID or just PID) is a number used by some operating system kernels (such as that of UNIX or Windows NT) to uniquely identify a process. Under Unix, the PID of a newly created child process is returned by the fork() system call to the parent. The PID can be passed to process control functions like waitpid() or kill() to perform actions on the given process, and if the operating system as procs support the files in /proc/pid/ can be queried for information about the process.]

[SDK] [A software development kit (SDK or “devkit”) is typically a set of development tools that allows a software engineer to create applications for a certain software package, software framework, hardware platform, operating system or similar. It may be something as simple as an application programming interface in the form of some files to interface to a particular programming language, or include sophisticated hardware to communicate with a certain embedded system. Common tools include debugging aids such as an IDE and other utilities. SDKs also frequently include sample code and supporting technical notes or other supporting documentation to help clarify points from the primary reference material.]

[SNMP] [The simple network management protocol (SNMP) forms part of the Internet protocol suite as defined by the Internet Engineering Task Force (IETF). More specifically, it is a Layer 7 or Application Layer protocol that is used by network management systems]
for monitoring network-attached devices for conditions that warrant administrative attention}
Appendix A  **RADIUS-servers**

### A.1 Radiator institutional server

**Note:** more explanation is given in the main document

Trace 4

LogDir /var/log/radius
DbDir /etc/radiator

AuthPort 1812
AcctPort 1813

<Client 192.168.10.200>
  Secret 6.6obaFkm&RNs666
  Identifier ACCESSPOINT1
  IdenticalClients 192.168.10.201
</Client>

<Handler Client-Identifier="/^(?!Proxy-Identifier$)/">
  <AuthBy RADIUS>
    Host 192.87.36.3
    Secret super_secret!
    AuthPort 1812
    AcctPort 1813
    StripFromReply Tunnel-Type, Tunnel-Medium-Type, \ Tunnel-Private-Group-ID
    AddToReply Tunnel-Type=1:VLAN,Tunnel-Medium-Type=1:Ether_802, \ Tunnel-Private-Group-ID=1:909
  </AuthBy>
</Handler>

<AuthBy FILE>
  Identifier ID4-TLS
  Filename %D/TLS-users
  EAPType TLS
  EAPTLS_CAFile %D/cert/institution-ca-chain.pem
  EAPTLS_CertificateFile %D/cert/radius-server-cert.pem
  EAPTLS_CertificateType PEM
  EAPTLS_PrivateKeyFile %D/cert/radius-server-key.pem
  EAPTLS_PrivateKeyPassword (the secret that secures the private-key)
  EAPTLS_MaxFragmentSize 1024
  AutoMPPEKeys
FreeRADIUS (http://www.freeradius.org) is a very powerful, configurable, fast and freely available open-source RADIUS server. The example configurations are based on FreeRADIUS 1.1.4 and require the regex patch for easy filtering of realms using regular expressions.

### A.2 FreeRADIUS institutional server

#### A.2.1 Setting up FreeRADIUS

FreeRADIUS is a very powerful, configurable, fast and freely available open-source RADIUS server. The example configurations are based on FreeRADIUS 1.1.4 and require the regex patch for easy filtering of realms using regular expressions.
expressions. The RPM installation packages for Fedora Core Linux and a standalone regex patch are available for download from the 
http://www.pingo.org/eduroam.
All the configuration files for FreeRADIUS are stored in a /etc/raddb directory. Configuration is broken down into several files by default.

Server is configured as follows:
• EAP-TTLS + PAP protocol for authenticating the clients
  o Outer identity is anonymous@domain.tld
• A few test user accounts are statically entered in a file
• Regular users are authenticated from an LDAP directory, using the eduPerson schema. Usernames are stored in 
eduPersonPrincipalName attribute and the plaintext password in userPassword attribute.
• Accounting is stored in a MySQL database
• Optionally a script can be set-up to update accounting SQL table with information of assigned IP addresses.

A.2.2 Defining clients - Access Points and RADIUS servers]

Access points, RADIUS servers and other RADIUS clients (NAS devices, RADIUS test scripts, ...) are defined in the file /etc/raddb/clients.conf. This file lists devices that may send requests to the server. Comment out or delete the existing entries and add the access points:

```
# APs and other devices
client ap<name 1> {
  secret = <shared secret 1>
  shortname = ap<name 1>
  nastype = cisco
}

client ap<name N> {
  secret = <shared secret N>
  shortname = ap<name N>
  nastype = cisco
}
```

For Cisco APs the nastype is set to cisco, for other NAS devices the value should be set differently. It is recommended to use a different shared secret for every access point. You can use mkpasswd to randomly generate shared secrets. Since a linked RADIUS server is viewed as a RADIUS client device, they also have to be added here:

```
# TLD radius - 1
client 10.1.2.3 {
  secret = <TLD shared secret 1>
  shortname = tld1
  nastype = other
}

# TLD radius - 2
client 10.1.2.4 {
  secret = <TLD shared secret 2>
  shortname = tld2
  nastype = other
}
```

A loopback client is useful for running testing scripts, so we make sure it is set correctly:
client 127.0.0.1 {
    secret = <shared secret for Loopback>
    shortname = loopback
    nastype = other
}

A.2.3 Configure realm handling and proxying

RADIUS requests from local users must be handled locally, while requests from roaming users must be proxied to the national TLD RADIUS server. If organisation has a domain domain.tld all requests for *.domain.tld are forwarded from the national RADIUS server to their organisational RADIUS server and it is their responsibility to filter out invalid domains via the blackhole rule. Proxying is configured in the file /etc/raddb/proxy.conf, the first rule that matches is used. The default configuration file could be cleared or the entries could be modified to match the provided samples. Instead of the “domain.tld” the administrator must use their delegated domain name.

# Proxy servers configuration
proxy server {
    synchronous = no
    retry_delay = 5
    retry_count = 2
    dead_time = 0
    default_fallback = no
    post_proxy_authorize = no
}

# We want to hadle the requests for domain.tld and the student.domain.tld with
# this server. If we wanted to handle more domains we can add as many as we want
# to.
realm domain.tld {
}
realm student.domain.tld {
}

# blackhole and NULL are also handled locally but we deny requests for them later
# in users file. NULL is an invalid realm since users aren't able to roam
# without the realm part. The blackhole realm is a regex with a catch-all to
# deny the requests for realms we are authoritative for, but aren't valid.
realm blackhole.domain.tld {
    regex = "^.*\domain\.tld$"
}
realm NULL {
}

# Unknown realms, we are not authoritative for are forwarded to TLD1
# or TLD2, round robin scheduling
realm DEFAULT {
    type = radius
    authhost = 10.1.2.3:1812
    accthost = 10.1.2.3:1813
    secret = <TLD shared secret 1>
    ldflag = round_robin
    nostrip
}
realm DEFAULT {
    type = radius
}
authhost = 10.1.2.4:1812
caccthost = 10.1.2.4:1813
secret = <TLD shared secret 2>
ldflag = round_robin
nostrip
}

Value ldflag is set to round_robin in order that requests to TLD servers are divided equally. The nostrip command specifies that no domain stripping is allowed (username must always be in the NetID form short_username@domain.tld, for both local and roaming users.

### A.2.4 Users authentication and realm handling

The configuration file /etc/raddb/users is very important. It contains statically configured users, specifies how request are handled and how different modules should authenticate and authorize the users. A \\ char shows an unintentional line-wrap. In the actual configuration file, the character must be removed and lines concatenated together. Administrator should clear or comment out the existing entries and add the following:

```
########################################################
# Users with a blackholed or NULL realm should be rejected
#
# domain.tld must be replaced with delegated domain name
########################################################
DEFAULT Realm == NULL, Auth-Type := Reject
DEFAULT Realm == blackhole.domain.tld, Auth-Type := Reject

########################################################
# User anonymous@realm must be allowed, activate EAP
# (Caution! Watch the line wrap)
########################################################
DEFAULT User-Name =~ "^[Aa][Nn][Oo][Nn][Yy][Mm][Oo][Uu][Ss]@.*$",\n Auth-Type := EAP

########################################################
# Accounting fix for AP;
# We send a real user-name in the outer Access-Accept
#
# domain.tld must be replaced with delegated domain name
########################################################
DEFAULT Realm == student.domain.tld, Freeradius-Proxied-To == 127.0.0.1
 User-Name = `{User-Name}`,
 Fall-Through = yes
DEFAULT Realm == domain.tld, Freeradius-Proxied-To == 127.0.0.1
 User-Name = `{User-Name}`,
 Fall-Through = yes

All NAS's used with this RADIUS configuration are required to follow RFC2865 recommendation for using the User-Name in Access-Accept for accounting information. With this records the outer anonymous username is replaced by the real users username so NAS can send correct accounting for the correct user.

########################################################
# Static test account for the NREN
#
# domain.tld must be replaced with delegated domain name
########################################################
nren_radius_test Realm == domain.tld, User-Password == "<test password>"
```

RADIUS links are establish with UDP and are connectionless. Every organisation is required to provide a test username to the NREN.
Inter-NREN Roaming: Cookbook

(more exactly: TLD RADIUS maintainers) for testing of the RADIUS link. This way we can also verify all RADIUS links when the TLD RADIUS is reconfigured. Activate LDAP authorization for the students:

```
########################################################
# Activate LDAP for the students, ID: LDAP2
# Linewrap caution! This MUST also be a single line
# domain.tld must be replaced with delegated domain name
########################################################
DEFAULT Realm == student.domain.tld, Autz-Type := LDAP2, Auth-Type := PAP
   Freradius-Proxied-To == 127.0.0.1
```

Activate LDAP authorization for the staff and assign them dynamically to a VLAN. We used 313 as an example, the actual VLAN number must be matched with the Access Point configuration. Use of dynamic VLAN mechanism is optional and is not mandated for eduroam networks.

```
########################################################
# Activate LDAP for the staff, ID: LDAP1
# Linewrap caution! Only Tunnel-* attributes are on new lines
# domain.tld must be replaced with delegated domain name
########################################################
DEFAULT Realm == domain.tld, Autz-Type := LDAP1, Auth-Type := PAP
   Freradius-Proxied-To == 127.0.0.1
   Tunnel-Type = VLAN,
   Tunnel-Medium-Type = IEEE-802,
   Tunnel-Private-Group-Id = 313
```

### A.2.5 Setting up accounting in the SQL database

All accounting information is logged into the MySQL database. The configuration is in the file `/etc/raddb/sql.conf`. Existing contents should be replaced with the following:

```
############################
# MySQL settings for accounting
############################
sql {
   driver = "rlm_sql_mysql"
   server = "localhost"
   login = "<user_for_mysql>"
   password = "<password_for_mysql>"
   radius_db = "radius"
   accounting_start_query = "INSERT into ACCOUNTING SET\n      `User-Name` = '%{User-Name}',\n      `Calling-Station-Id` = '%{Calling-Station-Id}',\n      `Called-Station-Id` = '%{Called-Station-Id}',\n      `NAS-IP-Address` = '%{NAS-IP-Address}',\n      `NAS-Port` = %{NAS-Port},\n      `Timestamp Start` = NOW(),\n      `Acct-Unique-Session-Id` = '%{Acct-Unique-Session-Id}'\n   "
   accounting_update_query = "UPDATE ACCOUNTING SET\n      `Acct-Session-Time` = '%{Acct-Session-Time}',\n   "
```

---

Project: GN2
Deliverable Number: DJX.0.1
Date of Issue: 20/03/07
EC Contract No.: 511082
Document Code: GN2-06-105v2
The format of the MySQL 'RADIUS' database is:

```sql
CREATE DATABASE radius;
USE radius;
CREATE TABLE ACCOUNTING (
    `User-Name` varchar(100) NOT NULL default '',
    `Calling-Station-Id` varchar(100) NOT NULL default '',
    `Client-IP-Address` varchar(100) NOT NULL default '',
    `Called-Station-Id` varchar(100) NOT NULL default '',
    `NAS-IP-Address` varchar(100) NOT NULL default '',
    `NAS-Port` int(10) unsigned NOT NULL default '0',
    `Timestamp Start` datetime NOT NULL default '1970-01-01 01:00:00',
    `Timestamp Dhcp` datetime NOT NULL default '1970-01-01 01:00:00',
    `Timestamp Stop` datetime NOT NULL default '1970-01-01 01:00:00',
    `Acct-Unique-Session-Id` varchar(100) NOT NULL default '',
    `Acct-Session-Time` int(10) unsigned NOT NULL default '0',
    `Acct-Input-Octets` int(10) unsigned NOT NULL default '0',
    `Acct-Output-Octets` int(10) unsigned NOT NULL default '0',
    `Acct-Input-Packets` int(10) unsigned NOT NULL default '0',
    `Acct-Output-Packets` int(10) unsigned NOT NULL default '0',
    `Acct-Terminate-Cause` varchar(100) NOT NULL default '',
);```

How to create the database:

```sql
CREATE DATABASE radius;
USE radius;
CREATE TABLE ACCOUNTING (``
A.2.6 The master RADIUS configuration

The master FreeRADIUS configuration file `/etc/raddb/radiusd.conf` specifies which modules are to be used, and defines the actual authentication. Default contents supplied with the freeradius tarball usually changes over the stable releases to reflect bigger changes in freeradius server. The entries in modules {} section should be added and/or modified while the authorize{} and the authenticate{} section should be commented out or replaced:

```sql
modules {

# We add the following to the modules{} section. These lines define two
# more ldap modules that are used to access different LDAP DIT trees or
# even two different LDAP servers. You can add as many LDAP modules as needed.
# If LDAP is not on localhost, you must use TLS
ldap ldap_student {
    server = "localhost"
    identity = "cn=RADIUS,dc=domain,dc=tld"
    password = "<secret for identity dn>"
    basedn = "ou=student,dc=domain,dc=tld"
    filter = "(eduPersonPrincipalName=%{User-Name})"
    start_tls = no
}
ldap ldap_staff {
    server = "localhost"
    identity = "cn=RADIUS,dc=domain,dc=tld"
    password = "<secret for identity dn>"
    basedn = "ou=staff,dc=domain,dc=tld"
    filter = "(eduPersonPrincipalName=%{User-Name})"
    start_tls = no
}

# Mind the Linewrap!
# This specifies which fields in accounting packet are used to compute the
# unique accounting ID. This value is as a key to insert record into the
# MySQL database.
# The original entry should be modified to include a few entries that are
# relevant for the standard Wireless AccessPoint fields.
acct_unique {
    key = "User-Name, Acct-Session-Id, Calling-Station-Id,\n         Called-Station-Id, NAS-IP-Address, NAS-Port"
}

# By default SQL isn't used so we must make sure the entry isn't commented out
$INCLUDE ${confdir}/sql.conf
}
```

The section “authorize” specifies how a request gets handled before the credential verification. The packet traverses modules in the following list:
• preprocess – cleans up some attributes from weird NAS's
• auth_log – logs the packet
• suffix – determines the correct realm
• files – rejects requests for invalid realms, activates EAP if needed, authorizes statically entered users or activates proper LDAP module for authorization and credential retrieval
• ldap_staff – authorizes and retrieves credentials for authenticating the staff
• ldap_students – authorizes and retrieves credentials for authenticating the students

The authorize {} section should be commented out or cleared and the following should be set:

```plaintext
authorize {
  preprocess
  auth_log
  suffix
  # files activate EAP for user anonymous@realm
  # If user entry isn't found the Autz-Type is set
  # depending on the user realm.
  files
  Autz-Type LDAP1 {
    ldap_staff
  }
  Autz-Type LDAP2 {
    ldap_student
  }
}
```

Note: A single realm (domain.tld) can be used for users in multiple LDAP backends or LDAP trees. In that case different LDAP modules can be queried one after another until a user is found. Such configuration is beyond the scope of this document but is well documented in FreeRADIUS documentation.

FreeRADIUS by default tries to support as many authentication protocols as possible. However for wireless some authentication protocols are inappropriate (for example EAP-MD5) or not secure enough (for example LEAP). We disable all the automatic detection of authentication protocols and use only the ones that are manually set (via the users file and/or other modules). Only allow EAP-TTLS + PAP:

The authenticate {} section should be commented out or cleared and the following should be set:

```plaintext
authenticate {
  Auth-Type EAP {
    eap
  }
  # password was retrieved during authorization from LDAP or from file
  # pap module just checks that LDAP and user supplied passwords match
  # With freeradius 1.1.4 PAP modul can autodetect if retrieved password
  # is cleartext or hash.
  Auth-Type PAP {
    pap
  }
}
```

Accounting details are logged to the file and then to the sql database. The authorize {} section should be commented out or cleared and the following should be set:

```plaintext
accounting {
  detail
  sql
}
```

4/`usr/share/freeradius-1.1.4/configurable_failover; see the group {} command`
A.2.7 Logging the client IP address ( Optionally! )

Client IP address is logged in the DHCP log file. However this information can also be stored in the ACCOUNTING table with other accounting data and thus provide easy access to that data. A Sysadmin needs to set-up a script to update the SQL database information with the assigned IP address. The client IP address is inserted by tailing the DHCP log file and monitoring all IP assignments. The script also monitors active connections and cleans up accounting (closes accounting) for stale connections. SNMP access to Access Points is required.

The script is available from here:
http://www.eduroam.si/uploads/CN/eC/CNeCC3Uc7XI9Tw_dLtWYZg/eduroam_monitor-20060809.tar.bz2

The access points need to be registered with the script. This is done by entering the needed access point data into the database.

```
USE radius;
cREATE TABLE access_points ( 
  IP address` varchar(100) PRIMARY KEY NOT NULL,
  `snmp secret` varchar(100) NOT NULL default '',
  `radius secret` varchar(100) NOT NULL default '',
  `root username` varchar(100) NOT NULL default '',
  `root password` varchar(100) NOT NULL default''
) TYPE=MyISAM;
```

A.2.8 More information

The original Slovenian Eduroam technical specifications and sample configuration site:
http://www.eduroam.si

ARNES AAI technical support e-mail address: aaa-podpora@arnes.si

FreeRADIUS files:
http://www.pingo.org/eduroam

Eduroam-in-a-box web configuration tool:
http://eduroam.sourceforge.net

A.3 Navis institutional and national server

A.3.1 Institutional Navis RADIUS server ( ver. 4.5.8 )

Set-up of a local RADIUS server.

Following configuration files have to be changed:

- server_properties
In addition the following files have to be replaced:

- acct_methods
- auth_methods

Changes in the `server_properties` file:
- Radius-Acct-Address = "*:1813"
- Radius-Auth-Address = "*:1812"
- Database-Address = "0"
- Radius-CharSet = UTF8
- Delimiter-Precendence = "@"
- Suffix-Delimiters = "@"

Add the following line in the `method_select` file:

```
* ClientClass setWorkingVars doAcct
```

Add the lines with the national proxy server information in the `clients` file:

```
<192.168.1.10 national_server_secret>
<192.168.1.20 national_server_secret>
...
```

Create the file `acct_methods` with the following content:

```
# This file specifies the processing steps (methods) to take for
# each RADIUS accounting request. The initial method is selected
# by rules in the method_select file.
#
# Drop any request Recieved
#
# doAcct Method-Type=WriteDebug Method-Disabled=FALSE
# WriteDebug-Map = "${Request Variable Group}=${request.*};
# ${Packet Variable Group}=${packet.*};
# ${User Variable Group}=${user.*};
# ${Check Variable Group}=${check.*};
# ${Reply Variable Group}=${reply.*};"
```

Create the file `auth_methods` with the following content:

```
setWorkingVars Method-Type=ReadWrite Method-Next=checklocal Method-On-Fail=discardPolicyError Method-On-Error=discardPolicyError
  ReadWrite-Map = "$\{user.nrp1ip\} := \"192.168.10.10\";\n\{user.nrp1port\} := \"1812\";\n\{user.nrp1secret\} := \"some-secret\";\n\{user.nrp1retry\} := \"1\";\n\{user.nrp1timeout\} := \"7000\";\n\{user.nrp2ip\} := \"192.168.10.20\";\n\{user.nrp2port\} := \"1812\";\n\{user.nrp2secret\} := \"some-secret\";\n\{user.nrp2retry\} := \"1\";\n\{user.nrp2timeout\} := \"7000\";\n\{user.localrealm\} := \"lorealm.tld\";"
  ReadWrite-NewUser = "FALSE"
checklocal Method-Type=Compare Method-On-Error=discardPolicyError Method-On-Fail=setProxyVars Method-Next=checkEAP
  Compare-Input1 = "$\{packet.User-Realm[trim,toLower]\}"
  Compare-Input2 = "$\{user.user.localrealm[trim,toLower]\}"
  Compare-Operator = "=="
```
setProxyVars Method-Type=ReadWrite Method-Next=doProxy1 Method-On-Fail=discardPolicyError Method-On-Error=discardPolicyError

ReadWrite-Map = "${user.proxy1ip} := ${user.nrplip};
${user.proxy1port} := ${user.nrplport};
${user.proxy1secret} := ${user.nrplsecret};
${user.proxy1retry} := ${user.nrplretry};
${user.proxy1timeout} := ${user.nrpltimeout};
${user.proxy2ip} := ${user.nrplip};
${user.proxy2port} := ${user.nrplport};
${user.proxy2secret} := ${user.nrplsecret};
${user.proxy2retry} := ${user.nrplretry};
${user.proxy2timeout} := ${user.nrpltimeout};
${user.proxyname} := \"Proxy RADIUS\";"

ReadWrite-NewUser = "FALSE"

doProxy1 Method-Type=Radius Method-Next=acceptAuthentication Method-On-Fail=rejectAuthentication

Radius-ServerAddress = "${user.proxy1ip}:${user.proxy1port}"
Radius-Secret = "${user.proxy1secret}"
Radius-Dictionary = "#default"
Radius-Timeout = "${user.proxy1timeout}"
Radius-Retries = "${user.proxy1retry}"
Radius-RequestMap = "${*}=${request.*};"
Radius-PacketType = "${packet.Packet-Type}"
Radius-ClientAddress = "**"
Radius-CharSet = "UTF8"
Radius-InauthenticFailure = "FALSE"
Radius-CheckAuthenticator = "TRUE"
Radius-RemoveTrailingNul = "TRUE"
Radius-AppendTrailingNul = "FALSE"
Radius-StrictEncoding = "FALSE"
Radius-CopyMode = "FALSE"
Radius-Mib = "AUTO"
Radius-RecvBufferSize = 8192
Radius-SendTimeout = 8192

doProxy2 Method-Type=Radius Method-Next=acceptAuthentication Method-On-Fail=rejectAuthentication Method-On-Error=discardProxiesError Method-Disabled=FALSE

Radius-ServerAddress = "${user.proxy2ip}:${user.proxy2port}"
Radius-Secret = "${user.proxy2secret}"
Radius-Dictionary = "#default"
Radius-Timeout = "${user.proxy2timeout}"
Radius-Retries = "${user.proxy2retry}"
Radius-RequestMap = "${*}=${request.*};"
Radius-ReplyMap = "${reply.*}=${*};"
Radius-PacketType = "${packet.Packet-Type}"
Radius-ClientAddress = "**"
Radius-CharSet = "UTF8"
Radius-InauthenticFailure = "FALSE"
Radius-CheckAuthenticator = "TRUE"
Radius-RemoveTrailingNul = "TRUE"
Radius-AppendTrailingNul = "FALSE"
Radius-StrictEncoding = "FALSE"
Radius-CopyMode = "FALSE"
Radius-Mib = "AUTO"
Radius-RecvBufferSize = 8192
Radius-SendTimeout = 8192

acceptAuthentication Method-Type=Return
Return-Disposition = "ACCEPT"
Return-LogLevel = "INFO"
Return-LogMessage = "AUTH: ACCEPT for user ${packet.Base-User-Name}@${packet.User-Realm} authenticated on ${user.proxyname}"

rejectAuthentication Method-Type=Return
  Return-Disposition = "REJECT"
  Return-LogLevel = "INFO"
  Return-LogMessage = "AUTH: REJECT for user ${packet.Base-User-Name}@${packet.User-Realm} authenticated on ${user.proxyname}"

cHECKEAP Method-Type=Compare Method-On-Error=discardPolicyError Method-On-Fail=discardEAPError Method-Next=doEAPTLS
  Compare-Input1 = "${packet.EAP-Identity}"
  Compare-Input2 = ""
  Compare-Operator = "!="

doEAPTLS Method-Type=AuthEapTtls Method-On-Error=discardPolicyError
  AuthEapTtls-TunnelMethod = "setLocalVars"
  AuthEapTtls-TunnelWriteMap = "${request.*}:=${tunnel.*};
${user.localrealm}:= ${user.localrealm};"
  AuthEapTtls-RsaCertFile = "server.pem"
  AuthEapTtls-RsaKeyPassword = "secret"
  AuthEapTtls-DsaCertFile = ""
  AuthEapTtls-DsaKeyPassword = ""
  AuthEapTtls-TrustedFile = "trust.pem"
  AuthEapTtls-FragmentSize = "253"
  AuthEapTtls-CertificateMap = "${packet.X509-Serial-Number} = ${Serial-Number};
${packet.X509-Subject-DN} = ${subject-DN};
${packet.X509-Issuer-DN} = ${Issuer-DN};"
  AuthEapTtls-KeyMap = "${reply.MS-MPPE-Recv-Key} := ${1-32[isRadius]};
${reply.MS-MPPE-Send-Key} := ${33-64[isRadius]};
${reply.EAP-Master-Session-Key} := ${1-64[isDiameter]};"

setLocalVars Method-Type=ReadWrite Method-Next=checkRealm Method-On-Fail=discardPolicyError Method-On-Error=discardPolicyError
  ReadWrite-Map = "${user.LDAPIP} := "192.168.100.1";
${user.LDAPport} := "389";
${user.uid} := ${packet.Base-User-Name[trim,toLower]};"
  ReadWrite-NewUser = "FALSE"

cHECKRealm Method-Type=Compare Method-On-Error=discardPolicyError Method-On-Fail=rejectRealm Method-Next=getRealmCountry
  Compare-Input1 = "${packet.User-Realm[trim,toLower]}"
  Compare-Input2 = "${user.localrealm[trim,toLower]}"
  Compare-Operator = "=="

getRealmCountry Method-Type=PatternMatch Method-Next=checkPassword Method-On-Fail=rejectRealm Method-On-Error=discardPolicyError
  PatternMatch-SearchKey = "${packet.User-Realm[toLower]}"
  PatternMatch-Mode = "REGEX"
  PatternMatch-Operation = "MATCHES"
  PatternMatch-Case = "(.+)(.+)
${user.realm}=${1};
${user.country}=${2};"
  PatternMatch-IgnoreCase = "TRUE"
  PatternMatch-SingleLine = "FALSE"
  PatternMatch-MultiLine = "FALSE"
  PatternMatch-Extended = "FALSE"

cHECKPassword Method-Type=Compare Method-On-Fail=rejectPassword Method-On-Error=discardPolicyError Method-Next=doLDAP
  Compare-Input1 = "${request.password}"
doLDAP
Method-Type=Ldap
Method-On-Fail=rejectLocalAuthentication
Method-On-Error=discardLDAPError
Method-Next=acceptLocalAuthentication

Ldap-ServerAddress = "${user.LDAPIP}:${user.LDAPport}"
Ldap-Version = "3"
Ldap-BindDn = "uid=${user.uid},dc=${user.realm},dc=${user.country}"
Ldap-BindPassword = "${request.password}"
Ldap-Operation = "BIND"
Ldap-SearchBase = "ou=People, o=${packet.User-Realm[escape]}"
Ldap-SearchScope = "SCOPE_ONE"
Ldap-SearchFilter = "(mail=${packet.Base-User-Name[escape]}@*)"
Ldap-ConnectionLimit = "2147483647"
Ldap-BindTimeout = "600000"
Ldap-ConnectionTimeout = "10000"
Ldap-OperationTimeout = "10000"
Ldap-CacheConnections = "TRUE"
Ldap-AuthFailureIsError = "FALSE"
Ldap-NewUser = "FALSE"
Ldap-StartTls = "FALSE"

rejectPassword
Method-Type=Return
Return-Disposition = "REJECT"
Return-LogLevel = "INFO"
Return-LogMessage = "AUTH: REJECT for user ${packet.Base-User-Name}@${packet.User-Realm} no valid password"

rejectRealm
Method-Type=Return
Return-Disposition = "REJECT"
Return-LogLevel = "INFO"
Return-LogMessage = "AUTH: REJECT for user ${packet.Base-User-Name}@${packet.User-Realm} no valid realm"

acceptLocalAuthentication
Method-Type=Return
Return-Disposition = "ACCEPT"
Return-LogLevel = "INFO"
Return-LogMessage = "AUTH: ACCEPT for user ${packet.Base-User-Name}@${packet.User-Realm} authenticated on LDAP"

rejectLocalAuthentication
Method-Type=Return
Return-Disposition = "REJECT"
Return-LogLevel = "INFO"
Return-LogMessage = "AUTH: REJECT for user ${packet.Base-User-Name}@${packet.User-Realm} authenticated on LDAP"

discardUnknownCountry
Method-Type=Return
Return-Disposition = "DISCARD"
Return-LogLevel = "INFO"
Return-LogMessage = "AUTH: DISCARD unknown country ${user.country}" 

discardEmptyCountry
Method-Type=Return
Return-Disposition = "DISCARD"
Return-LogLevel = "INFO"
Return-LogMessage = "AUTH: DISCARD empty country ${packet.User-Realm}" 

discardPolicyError
Method-Type=Return
Return-Disposition = "DISCARD"
Return-LogLevel = "INFO"

discardProxiesError   Method-Type=Return
   Return-Disposition = "DISCARD"
   Return-LogLevel = "INFO"
   Return-LogMessage = "AUTH: DISCARD proxies error ${user.proxyname}" 

discardEAPError       Method-Type=Return
   Return-Disposition = "DISCARD"
   Return-LogLevel = "INFO"
   Return-LogMessage = "AUTH: DISCARD not EAP ${request.User-Name}" 

discardLDAPError      Method-Type=Return
   Return-Disposition = "DISCARD"
   Return-LogLevel = "INFO"
   Return-LogMessage = "AUTH: DISCARD LDAP not respond"

In the above file enter the correct info about the proxy servers, (for IdP case ONLY) the local realm and the IdM system:

${user.nrp*} – enter correct info about the proxy servers (IP addresses, ports, ...)
${user.localrealm} – enter correct info about the local realm (IdP ONLY)
${user.LDAP*}   – enter correct info about the local LDAP server (IdM system)

Additional files needed (due to the use of EAP):
server.pem – server certificate
trust.pem – root certificate

A.3.2 National Navis RADIUS server (ver. 4.5.8)

Set-up of a (national) RADIUS proxy server.

The following configuration files have to be changed:

• server_properties
• method_select
• clients

In addition the following files have to be replaced:

• acct_methods
• auth_methods

Changes in the server_properties file:

Radius-Acct-Address = "*:1813"
Radius-Auth-Address = "*:1812"
Database-Address = "0"
Radius-CharSet = UTF8
Delimiter-Precedence = "@"
Suffix-Delimiters = "@"

Add the following line in the method_select file:
Add the lines with the eduroam proxy server and local RADIUS servers information to the clients file:

```
192.87.106.34 <eduroam_secret>
130.225.242.109 <eduroam_secret>
<local_server_1_IP> <local_server_secret>
<local_server_2_IP> <local_server_secret>
```

Create the file `acct_methods` with the following content:

```
# This file specifies the processing steps (methods) to take for each RADIUS accounting request. The initial method is selected by rules in the method_select file.
#
#-----------------------------------------------------------------------------------
# Drop any request Recieved
#-----------------------------------------------------------------------------------

doDropRequest Method-Type=Return Method-Disabled=FALSE
Return-Disposition = "DISCARD"
Return-LogLevel = "INFO"
Return-LogMessage = "ACCN: DISCARD Client ${packet.Source-Address} sent accounting"
```

Create the file `auth_methods` with the following content:

```
setWorkingVars Method-Type=ReadWrite Method-Disabled=FALSE Method-On-Next=getCountry Method-On-Fail=discardPolicyError Method-On-Error=discardPolicyError
ReadWrite-Map = <<
${user.erp1ip} := "192.87.106.34";
${user.erp1port} := "1812";
${user.erp1secret} := "some-secret";
${user.erp1retry} := "1";
${user.erp1timeout} := "7000";
${user.erp2ip} := "130.225.242.109";
${user.erp2port} := "1812";
${user.erp2secret} := "some-secret";
${user.erp2retry} := "1";
${user.erp2timeout} := "7000";
${user.lop1ip} := "192.168.200.1";
${user.lop1port} := "1812";
${user.lop1secret} := "local-some-secret";
${user.lop1retry} := "1";
${user.lop1timeout} := "7000";
${user.lop2ip} := "192.168.200.2";
${user.lop2port} := "1812";
${user.lop2secret} := "local-some-secret";
${user.lop2retry} := "1";
${user.lop2timeout} := "7000";
```
PatternMatch-SearchKey = "${packet.User-Realm}"
PatternMatch-Mode = "REGEX"
PatternMatch-Operation = "MATCHES"
PatternMatch-Case = "(.+)\.(.+)$  ${user.country} = ${2};"
PatternMatch-IgnoreCase = "TRUE"
PatternMatch-MultiLine = "FALSE"
PatternMatch-Extended = "FALSE"

setUserCountry Method-Type=ReadWrite Method-Disabled=FALSE Method-Next=checkeduroam Method-On-Fail=discardEmptyCountry Method-On-Error=discardPolicyError
ReadWrite-Map = "*${user.country} := "*${user.country[toLower]}";"
ReadWrite-NewUser = "FALSE"

checkeduroam Method-Type=Compare Method-Disabled=FALSE Method-On-Fail=discardUnknownCountry Method-On-Error=discardPolicyError Method-Next=setProxyVarseduroam
Compare-Input1 = "${user.eduroamcountry}"
Compare-Input2 = "${user.country}"
Compare-Operator = "contains"

checklocal Method-Type=Compare Method-Disabled=FALSE Method-On-Fail=discardUnknownCountry Method-On-Error=discardPolicyError Method-Next=setProxyVarslocal
Compare-Input1 = "${user.localcountry}"
Compare-Input2 = "${user.country}"
Compare-Operator = "=="

setProxyVarseduroam Method-Type=ReadWrite Method-Disabled=FALSE Method-On-Fail=discardEmptyCountry Method-On-Error=discardPolicyError Method-Next=doProxy1
ReadWrite-Map = «
${user.proxy1ip} := ${user.erp1ip};
${user.proxy1port} := ${user.erp1port};
${user.proxy1secret} := ${user.erp1secret};
${user.proxy1retry} := ${user.erp1retry};
${user.proxy1timeout} := ${user.erp1timeout};
${user.proxy2ip} := ${user.erp2ip};
${user.proxy2port} := ${user.erp2port};
${user.proxy2secret} := ${user.erp2secret};
${user.proxy2retry} := ${user.erp2retry};
${user.proxy2timeout} := ${user.erp2timeout};
${user.proxyname} := "eduroam";
»
ReadWrite-NewUser = "FALSE"

setProxyVarslocal Method-Type=ReadWrite Method-Disabled=FALSE Method-On-Fail=discardEmptyCountry Method-On-Error=discardPolicyError Method-Next=doProxy1
ReadWrite-Map = <<
${user.proxy1ip} := ${user.lop1ip};
${user.proxy1port} := ${user.lop1port};
${user.proxy1secret} := ${user.lop1secret};
${user.proxy1retry} := ${user.lop1retry};
${user.proxy1timeout} := ${user.lop1timeout};
${user.proxy2ip} := ${user.lop2ip};
${user.proxy2port} := ${user.lop2port};
${user.proxy2secret} := ${user.lop2secret};
${user.proxy2retry} := ${user.lop2retry};
${user.proxy2timeout} := ${user.lop2timeout};
${user.proxyname} := "eduroam";
>>
ReadWrite-NewUser = "FALSE"
\$\{user.proxy1.timeout\} := \$\{user.lop1.timeout\};
\$\{user.proxy1.ip\} := \$\{user.lop1.ip\};
\$\{user.proxy1.port\} := \$\{user.lop1.port\};
\$\{user.proxy1.secret\} := \$\{user.lop1.secret\};
\$\{user.proxy1.retry\} := \$\{user.lop1.retry\};
\$\{user.proxy1.timeout\} := \$\{user.lop1.timeout\};
\$\{user.proxyname\} := "local";

```
>> ReadWrite-NewUser = "FALSE"

doProxy1 Method-Type=Radius Method-Disabled=FALSE Method-On-Error=doProxy2 Method-Next=acceptAuthentication Method-On-Fail=rejectAuthentication
   Radius-ServerAddress = "\$\{user.proxy1.ip\}:\$\{user.proxy1.port\}"
   Radius-Secret = "\$\{user.proxy1.secret\}"
   Radius-Dictionary = "#default"
   Radius-Timeout = "\$\{user.proxy1.timeout\}"
   Radius-Retries = "\$\{user.proxy1.retry\}"
   Radius-RequestMap = "\$\{\}"="\$\{request.\};"
   Radius-ReplyMap = "\$\{reply.\}"="\$\{\};"
   Radius-PacketType = "\$\{packet.Packet-Type\}"
   Radius-ClientAddress = "*
   Radius-CharSet = "UTF8"
   Radius-InauthenticFailure = "FALSE"
   Radius-CheckAuthenticator = "TRUE"
   Radius-RemoveTrailingNul = "TRUE"
   Radius-AppendTrailingNul = "FALSE"
   Radius-StrictEncoding = "FALSE"
   Radius-CopyMode = "FALSE"
   Radius-Mib = "AUTO"
   Radius-RecvBufferSize = "$192"
   Radius-SendBufferSize = "$192"
```

```
doProxy2 Method-Type=Radius Method-Disabled=FALSE Method-On-Fail=rejectAuthentication Method-On-Error=discardProxiesError
   Radius-ServerAddress = "\$\{user.proxy2.ip\}:\$\{user.proxy2.port\}"
   Radius-Secret = "\$\{user.proxy2.secret\}"
   Radius-Dictionary = "#default"
   Radius-Timeout = "\$\{user.proxy2.timeout\}"
   Radius-Retries = "\$\{user.proxy2.retry\}"
   Radius-RequestMap = "\$\{\}"="\$\{request.\};"
   Radius-ReplyMap = "\$\{reply.\}"="\$\{\};"
   Radius-PacketType = "\$\{packet.Packet-Type\}"
   Radius-ClientAddress = "*
   Radius-CharSet = "UTF8"
   Radius-InauthenticFailure = "FALSE"
   Radius-CheckAuthenticator = "TRUE"
   Radius-RemoveTrailingNul = "TRUE"
   Radius-AppendTrailingNul = "FALSE"
   Radius-StrictEncoding = "FALSE"
   Radius-CopyMode = "FALSE"
   Radius-Mib = "AUTO"
   Radius-RecvBufferSize = "$192"
   Radius-SendBufferSize = "$192"
```

discardUnknownCountry Method-Type=Return Method-Disabled=FALSE
   Return-Disposition = "DISCARD"
   Return-LogLevel = "INFO"
In the above file enter the correct info about the eduroam and local servers. Also enter the list of TLDs participating in eduroam:

- \texttt{${user.erp*}$} – enter correct info about the eduroam servers (IP addresses, ports, ...)
- \texttt{${user.lop*}$} – enter correct info about the local RADIUS servers (IP addresses, ports, ...)
- \texttt{${user.eduroamcountry}$} – enter the list of TLDs participating in eduroam
- \texttt{${user.localcountry}$} – enter local TLD

\section{A.4 VitalAAA Institutional and national server}

\subsection{A.4.1 Institutional VitalAAA Server (ver. 5.0.10)}

Set-up of a local RADIUS server.

The following configuration files have to be changed:
- \texttt{server.properties}
- \texttt{method_dispatch}
- \texttt{clients}

In addition the following files have to be created.
Inter-NREN Roaming: Cookbook

- error.pf
- proxy.pf
- acct.pf
- prepare.pf
- local.pf

Changes in the `server_properties` file:
- Radius-Acct-Address = "*:1813"
- Radius-Auth-Address = "*:1812"
- Database-Address = "0"
- Radius-Charset = UTF8
- Delimiter-Precedence = "@"
- Suffix-Delimiters = "@"

Changes in the `method_dispatch` file:
- radius Auth 1 prepare setWorkingVars
  - radius acct 4 acct dumpAcct

Add the lines with the national proxy server information in the `clients` file:
- `<192.168.1.10 national_server_secret>
- `<192.168.1.20 national_server_secret>

... 

Create the file `error.pf` with the following content:

```plaintext
discardUnknownCountry
  Method-Type = "Return"
  Method-Disabled = "FALSE"
  Return-Disposition = "DISCARD"
  Return-LogLevel = "INFO"
  Return-LogMessage = "AUTH: DISCARD unknown country ${user.country}" 

discardEmptyCountry
  Method-Type = "Return"
  Method-Disabled = "FALSE"
  Return-Disposition = "DISCARD"
  Return-LogLevel = "INFO"
  Return-LogMessage = "AUTH: DISCARD empty country ${packet.User-Realm}" 

discardPolicyError
  Method-Type = "Return"
  Method-Disabled = "FALSE"
  Return-Disposition = "DISCARD"
  Return-LogLevel = "INFO"
  Return-LogMessage = "AUTH: DISCARD policy error ${packet.Last-Disposition}

Message)"

discardProxiesError
  Method-Type = "Return"
  Method-Disabled = "FALSE"
  Return-Disposition = "DISCARD"
  Return-LogLevel = "INFO"
  Return-LogMessage = "AUTH: DISCARD proxies error ${user.proxynname}" 

discardEAPError
```
Create the file `proxy.pf` with the following content:

```plaintext
doProxy1
Method-Type = "Radius"
Method-Disabled = "FALSE"
Method-On-Success = "acceptAuthentication"
Method-On-Failure = "rejectAuthentication"
Radius-ServerAddress = "${user.proxy1ip}:${user.proxy1port}"
Radius-Secret = "${user.proxy1secret}"
Radius-Dictionary = "#default"
Radius-Timeout = "${user.proxy1timeout}";
Radius-Retries = "${user.proxy1retry}";
Radius-RequestMap = "${*}=${request.*};"
Radius-PacketType = "${packet.Packet-Type}";
Radius-ClientAddress = "*"
Radius-InauthenticFailure = "FALSE"
Radius-CheckAuthenticator = "TRUE"
Radius-RemoveTrailingNul = "TRUE"
Radius-AppendTrailingNul = "FALSE"
Radius-StrictEncoding = "FALSE"
Radius-CopyMode = "FALSE"
Radius-Mib = "AUTO"
Radius-RecvBufferSize = "8192"
Radius-SendBufferSize = "8192"
Radius-SuccessMap = "${reply.*}=${*};"

doProxy2
Method-Type = "Radius"
Method-Disabled = "FALSE"
Method-On-Success = "acceptAuthentication"
Method-On-Failure = "rejectAuthentication"
Radius-ServerAddress = "${user.proxy2ip}:${user.proxy2port}"
Radius-Secret = "${user.proxy2secret}"
Radius-Dictionary = "#default"
Radius-Timeout = "${user.proxy2timeout}"
Radius-Retries = "${user.proxy2retry}";
Radius-RequestMap = "${*}=${request.*};"
Radius-PacketType = "${packet.Packet-Type}";
Radius-ClientAddress = "*"
Radius-InauthenticFailure = "FALSE"
```

```
Radius-CheckAuthenticator = "TRUE"
Radius-RemoveTrailingNul = "TRUE"
Radius-AppendTrailingNul = "FALSE"
Radius-StrictEncoding = "FALSE"
Radius-CopyMode = "FALSE"
Radius-Mib = "AUTO"
Radius-RecvBufferSize = "8192"
Radius-SendBufferSize = "8192"
Radius-SuccessMap = "${reply.*}=${*};"
Method-On-Error = "errror:discardProxiesError"

acceptAuhentication
Method-Type = "Return"
Method-Disabled = "FALSE"
Return-Disposition = "ACCEPT"
Return-LogLevel = "INFO"
Return-LogMessage = "AUTH: ACCEPT for user ${packet.Base-User-Name}@${packet.User-Realm} authenticated on ${user.proxyname}"

rejectAuhentication
Method-Type = "Return"
Method-Disabled = "FALSE"
Return-Disposition = "REJECT"
Return-LogLevel = "INFO"
Return-LogMessage = "AUTH: REJECT for user ${packet.Base-

Create the file acct.pf with the following content:
dumpAcct
Method-Type = "WriteDebug"
Method-Disabled = "FALSE"
Level-On-Success = "INFO"
Level-On-Failure = "INFO"
Level-On-Error = "INFO"
WriteDebug-Map = "${Request Variable Group}=${request.*};
${Packet Variable Group}=${packet.*};
${User Variable Group}=${user.*};
${Check Variable Group}=${check.*};
${Reply Variable Group}=${reply.*};"

Create the file prepare.pf with the following content:
setWorkingVars
Method-Type = "ReadWrite"
Method-On-Success = "checklocal"
Method-On-Failure = "errror:discardPolicyError"
Method-On-Error = "errror:discardPolicyError"
Method-Disabled = "FALSE"
Level-On-Success = "INFO"
Level-On-Failure = "INFO"
Level-On-Error = "INFO"
ReadWrite-Map = "${user.nrp1ip} := "192.168.10.10";
${user.nrp1port} := "1812";
${user.nrp1secret} := "some-secret";
${user.nrp1retry} := "1";
${user.nrp1timeout} := "7000";
${user.nrp2ip} := "192.168.10.20";
${user.nrp2port} := "1812";
${user.nrp2secret} := "some-secret";
${user.nrp2retry} := "1";
${user.localrealm} := "lorealm.tld";"
ReadWrite-NewUser = "FALSE"

cHECKLOCAL
Method-Type = "Compare"
setProxyVars
Method-Type = "ReadWrite"
Method-On-Success = "proxy:doProxy1"
Method-On-Failure = "error:discardProxyError"
Method-On-Error = "error:discardProxyError"
Method-Disabled = "FALSE"
Level-On-Success = "INFO"
Level-On-Failure = "INFO"
Level-On-Error = "INFO"

ReadWrite-Map = "${user.proxy1ip} := ${user.nrp1ip};
${user.proxy1port} := ${user.nrp1port};
${user.proxy1secret} := ${user.nrp1secret};
${user.proxy1retry} := ${user.nrp1retry};
${user.proxy1timeout} := ${user.nrp1timeout};
${user.proxy2ip} := ${user.nrp2ip};
${user.proxy2port} := ${user.nrp2port};
${user.proxy2secret} := ${user.nrp2secret};
${user.proxy2retry} := ${user.nrp2retry};
${user.proxy2timeout} := ${user.nrp2timeout};
${user.proxyname} := "Proxy RADIUS";"

In the file above enter the correct info about the proxy servers and (for IdP case ONLY) for the local realm:

${user.nrp*} – enter the correct info about the proxy servers (IP addresses, ports, ...)

${user.localrealm} – enter the correct info about the local realm (IdP ONLY)

Create the file local.pf with the following content:

checkEAP
Method-Type = "Compare"
Method-Disabled = "FALSE"
Level-On-Success = "INFO"
Level-On-Failure = "INFO"
Level-On-Error = "INFO"
Compare-Input1 = "${packet.EAP-Identity}"
Compare-Input2 = ""
Compare-Operator = "!="
Method-On-Error = "error:discardPolicyError"
Method-On-Failure = "error:discardEAPError"
Method-On-Success = "doEAPTTLS"

doEAPTTLS
Method-Type = "AuthEapTtls"
Method-On-Error = "error:discardPolicyError"
Method-Disabled = "FALSE"
Level-On-Success = "INFO"
Level-On-Failure = "INFO"
Level-On-Error = "INFO"
AuthEapTtls-TunnelMethod = "setLocalVars"
AuthEapTtls-TunnelWriteMap = "${request.*} := ${tunnel.*};
${user.localrealm} := ${user.localrealm};"

AuthEapTtls-RsaCertFile = "server.pem"
AuthEapTtls-RsaKeyPassword = "secret"
AuthEapTtls-DsaCertFile = ""
AuthEapTtls-DsaKeyPassword = ""
AuthEapTtls-TrustedFile = "trust.pem"
AuthEapTtls-FragmentSize = "253"
AuthEapTtls-CertificateMap = "${packet.X509-Serial-Number} = ${Serial-Number};n${packet.X509-Subject-DN} = ${subject-DN};n${packet.X509-Issuer-DN} = ${Issuer-DN};" 
AuthEapTtls-KeyMap = "${reply.MS-MPPE-Recv-Key} := ${1-32[isRadius]};n${reply.MS-MPPE-Send-Key} := ${33-64[isRadius]};n${reply.EAP-Master-Session-Key} : = ${1-64[isDiameter]};"

setLocalVars
Method-Type = "ReadWrite"
Method-Disabled = "FALSE"
Level-On-Success = "INFO"
Level-On-Failure = "INFO"
Level-On-Error = "INFO"
ReadWrite-Map = "${user.LDAPIP} := \"192.168.100.1\";n${user.LDAPPort} := \"389\";n${user.uid} := ${packet.Base-User-Name[trim,toLower]};"

readWrite-NewUser = "FALSE"
Method-On-Success = "checkRealm"
Method-On-Failure = "errror:discardPolicyError"
Method-On-Error = "errror:discardPolicyError"

checkRealm
Method-Type = "Compare"
Method-Disabled = "FALSE"
Level-On-Success = "INFO"
Level-On-Failure = "INFO"
Level-On-Error = "INFO"
Compare-Input1 = "${packet.User-Realm[trim,toLower]}"
Compare-Input2 = "${user.localrealm[trim,toLower]}"
Compare-Operator = "=="
Compare-ErrorMessage = "errror:discardPolicyError"
Method-On-Failure = "rejectRealm"
Method-On-Success = "getRealmCountry"

getRealmCountry
Method-Type = "PatternMatch"
Method-Disabled = "FALSE"
Level-On-Success = "INFO"
Level-On-Failure = "INFO"
Level-On-Error = "INFO"
PatternMatch-SearchKey = "${packet.User-Realm[toLower]}"
PatternMatch-Mode = "REGEX"
PatternMatch-Operation = "MATHES"
PatternMatch-Case = "(.+)\.(.+)$ ${user.realm}=${1};${user.country}=${2};"
PatternMatch-IgnoreCase = "TRUE"
PatternMatch-SingleLine = "FALSE"
PatternMatch-MultiLine = "FALSE"
PatternMatch-Extended = "FALSE"
Method-On-Success = "checkPassword"
Method-On-Failure = "rejectRealm"
Method-On-Error = "errror:discardPolicyError"

checkPassword
Method-Type = "Compare"
Method-On-Failure = "rejectPassword"
Method-On-Error = "errror:discardPolicyError"
Method-Disabled = "FALSE"
Level-On-Success = "INFO"
Level-On-Failure = "INFO"
Level-On-Error = "INFO"
Compare-Input1 = "$\{\text{request.password}\}"
Compare-Input2 = ""
Compare-Operator = "!="
Method-On-Success = "doLDAP"
Method-On-Failure = "rejectAuthentication"

**doLDAP**

Method-Type = "Ldap"
Method-Disabled = "FALSE"
Level-On-Success = "INFO"
Level-On-Failure = "INFO"
Level-On-Error = "INFO"
Ldap-ServerAddress = "$\{\text{user.LDAPIP}\}:\{\text{user.LDAPport}\}"
Ldap-Version = "3"
Ldap-BindDn = "uid=\{\text{user.uid}\},dc=\{\text{user.realm}\},dc=\{\text{user.country}\}"
Ldap-BindPassword = "$\{\text{request.password}\}"
Ldap-Operation = "BIND"
Ldap-SearchBase = "ou=People, o=\{\text{packet.User-Realm[escape]}\}"
Ldap-SearchScope = "SCOPE_ONE"
Ldap-SearchFilter = "(mail=\{\text{packet.Base-User-Name[escape]}\}@*)"
Ldap-MaxSearchResults = "1"
Ldap-ConnectionLimit = "2147483647"
Ldap-BindTimeout = "600000"
Ldap-ConnectionTimeout = "10000"
Ldap-OperationTimeout = "10000"
Ldap-CacheConnections = "TRUE"
Ldap-AuthFailureIsError = "FALSE"
Ldap-NewUser = "FALSE"
Ldap-StartTls = "FALSE"
Method-On-Failure = "error:discardLDAPError"
Method-On-Success = "acceptAuthentication"
Method-On-Success = "rejectAuthentication"

**rejectPassword**

Method-Type = "Return"
Method-Disabled = "FALSE"
Level-On-Success = "INFO"
Level-On-Failure = "INFO"
Level-On-Error = "INFO"
Return-Disposition = "REJECT"
Return-LogLevel = "INFO"
Return-LogMessage = "AUTH: REJECT for user $\{\text{packet.Base-User-Name}@\}$
{\text{packet.User-Realm}} no valid password"

**rejectRealm**

Method-Type = "Return"
Method-Disabled = "FALSE"
Level-On-Success = "INFO"
Level-On-Failure = "INFO"
Level-On-Error = "INFO"
Return-Disposition = "REJECT"
Return-LogLevel = "INFO"
Return-LogMessage = "AUTH: REJECT for user $\{\text{packet.Base-User-Name}@\}$
{\text{packet.User-Realm}} no valid realm"

**acceptAuthentication**

Method-Type = "Return"
Method-Disabled = "FALSE"
Level-On-Success = "INFO"
Level-On-Failure = "INFO"
Level-On-Error = "INFO"
Return-Disposition = "ACCEPT"
Return-LogLevel = "INFO"
Return-LogMessage = "AUTH: ACCEPT for user ${packet.Base-User-Name}@${packet.User-Realm} authenticated on LDAP"

rejectAuthentication
Method-Type = "Return"
Method-Disabled = "FALSE"
Level-On-Success = "INFO"
Level-On-Failure = "INFO"
Level-On-Error = "INFO"
Return-Disposition = "REJECT"
Return-LogLevel = "INFO"
Return-LogMessage = "AUTH: REJECT for user ${packet.Base-User-Name}@${packet.User-Realm} authenticated on LDAP"

In the file above (for IdP case ONLY) the correct info about the local IdM system:
${user.LDAP*} enter the correct info about the local LDAP server (IdM system)

Additional files needed (due to the use of EAP):
server.pem - server certificate
trust.pem - root certificate

A.4.2 National VitalAAA Server (ver. 5.0.10)

Set-up of a (national) RADIUS proxy server.

The following configuration files have to be changed:
• server_properties
• method_dispatch
• clients

In addition the following files have to be created.
• aaa.pf
• error.pf
• proxy.pf
• prepare.pf

Changes in the server_properties file:
Radius-Acct-Address = "*:1813"
Radius-Auth-Address = "*:1812"
Database-Address = "0"
Radius-CharSet = UTF8
Delimiter-Precedence = "@"
Suffix-Delimiters = "@"

Changes in the method_dispatch file:
radius Auth 1 prepare setWorkingVars
radius acct 4 aaa dropRadiusAcct
Add the the lines with the eduroam proxy server and the local RADIUS servers to the clients file:

```
192.87.106.34  <eduroam_secret>
130.225.242.109 <eduroam_secret>
<192.168.1.10> <local_server_secret>
<192.168.1.20> <local_server_secret>
...
```

Create the file `aaa.pf` with the following content:

```
# This file specifies the processing steps (methods) to take for
# each AAA request. The initial method is selected
# by rules in the method_dispatch file.
# Revision $Id: aaa.pf,v 1.1 2006/03/12 02:40:42 glenn Exp $
#-----------------------------------------------------------
dropStateServerAuth
  Method-Type = "Return"
  Return-Disposition = "DISCARD"
  Return-LogLevel = "INFO"
  Return-LogMessage = "Discarding StateServer Auth Request"

dropStateServerAcct
  Method-Type = "Return"
  Return-Disposition = "DISCARD"
  Return-LogLevel = "INFO"
  Return-LogMessage = "Discarding StateServer Acct Request"

dropRadiusAuth
  Method-Type = "Return"
  Return-Disposition = "DISCARD"
  Return-LogLevel = "INFO"
  Return-LogMessage = "Discarding RADIUS Auth Request"

dropRadiusAcct
  Method-Type = "Return"
  Method-Disabled = "FALSE"
  Level-On-Success = "INFO"
  Level-On-Failure = "INFO"
  Level-On-Error = "INFO"
  Return-Disposition = "DISCARD"
  Return-LogLevel = "INFO"
  Return-LogMessage = "ACCN: DISCARD Client ${packet.Source-Address} sent accounting"

dropDiameterAuth
  Method-Type = "Return"
  Return-Disposition = "ERROR"
  Return-LogLevel = "INFO"
  Return-LogMessage = "Discarding Diamater Auth Request"

dropDiameterAcct
  Method-Type = "Return"
  Return-Disposition = "ERROR"
  Return-LogLevel = "INFO"
  Return-LogMessage = "Discarding Diamater Acct Request"
```

Create the file `error.pf` with the following content:

```
discardUnknownCountry
```
Method-Type = "Return"
Method-Disabled = "FALSE"
Return-Disposition = "DISCARD"
Return-LogLevel = "INFO"
Return-LogMessage = "AUTH: DISCARD unknown country ${user.country}"

discardEmptyCountry
Method-Type = "Return"
Method-Disabled = "FALSE"
Return-Disposition = "DISCARD"
Return-LogLevel = "INFO"
Return-LogMessage = "AUTH: DISCARD empty country ${packet.User-Realm}"

discardPolicyError
Method-Type = "Return"
Method-Disabled = "FALSE"
Return-Disposition = "DISCARD"
Return-LogLevel = "INFO"

discardProxiesError
Method-Type = "Return"
Method-Disabled = "FALSE"
Return-Disposition = "DISCARD"
Return-LogLevel = "INFO"
Return-LogMessage = "AUTH: DISCARD proxies error ${user.proxyname}"

Create the file proxy.pf with the following content:
doProxy1
Method-Type = "Radius"
Method-Disabled = "FALSE"
Method-On-Error = "doProxy2"
Method-On-Success = "acceptAuthentication"
Method-On-Failure = "rejectAuthentication"
Radius-ServerAddress = "${user.proxy1ip}:${user.proxy1port}"
Radius-Secret = "${user.proxy1secret}"
Radius-Dictionary = "#default"
Radius-Timeout = "${user.proxy1timeout}"
Radius-Retrries = "${user.proxy1retry}";
Radius-RequestMap = "\${*}=${request.*};"
Radius-PacketType = "${packet.Packet-Type}"
Radius-ClientAddress = "**"
Radius-CharSet = "UTF8"
Radius-InauthenticFailure = "FALSE"
Radius-CheckAuthenticator = "TRUE"
Radius-RemoveTrailingNul = "TRUE"
Radius-AppendTrailingNul = "FALSE"
Radius-StrictEncoding = "FALSE"
Radius-CopyMode = "FALSE"
Radius-Mib = "AUTO"
Radius-RecvBufferSize = "8192"
Radius-SendBufferSize = "8192"
Radius-SuccessMap = "\${reply.*}=${*}"

doProxy2
Method-Type = "Radius"
setWorkingVars
Method-Type = "ReadWrite"
Method-Disabled = "FALSE"
Method-On-Success = "getCountry"
setWorkingVars = <<
${user.erp1ip} := "192.87.106.34";
${user.erp1port} := "1812";
${user.erp1secret} := "some-secret";
${user.erp1retry} := "1";
${user.erp1timeout} := "7000";
${user.erp2ip} := "130.225.242.109";
${user.erp2port} := "1812";
${user.erp2secret} := "some-secret";
${user.erp2retry} := "1";
${user.erp2timeout} := "7000";
${user.loplip} := "192.168.200.1";
${user.loplport} := "1812";
${user.loplsecret} := "local-some-secret";
${user.loplretry} := "1";
createSetWorkingVars = "prepare.pf" with the following content:
setWorkingVars
Method-Type = "ReadWrite"
Method-Disabled = "FALSE"
Method-On-Success = "getCountry"
setWorkingVars = <<
${user.erp1ip} := "192.87.106.34";
${user.erp1port} := "1812";
${user.erp1secret} := "some-secret";
${user.erp1retry} := "1";
${user.erp1timeout} := "7000";
${user.erp2ip} := "130.225.242.109";
${user.erp2port} := "1812";
${user.erp2secret} := "some-secret";
${user.erp2retry} := "1";
${user.erp2timeout} := "7000";
${user.loplip} := "192.168.200.1";
${user.loplport} := "1812";
${user.loplsecret} := "local-some-secret";
${user.loplretry} := "1";
createSetWorkingVars = "prepare.pf" with the following content:
setWorkingVars
Method-Type = "ReadWrite"
Method-Disabled = "FALSE"
Method-On-Success = "getCountry"
setWorkingVars = <<
${user.erp1ip} := "192.87.106.34";
${user.erp1port} := "1812";
${user.erp1secret} := "some-secret";
${user.erp1retry} := "1";
${user.erp1timeout} := "7000";
${user.erp2ip} := "130.225.242.109";
${user.erp2port} := "1812";
${user.erp2secret} := "some-secret";
${user.erp2retry} := "1";
${user.erp2timeout} := "7000";
${user.loplip} := "192.168.200.1";
${user.loplport} := "1812";
${user.loplsecret} := "local-some-secret";
${user.loplretry} := "1";
createSetWorkingVars = "prepare.pf" with the following content:
setWorkingVars
Method-Type = "ReadWrite"
Method-Disabled = "FALSE"
Method-On-Success = "getCountry"
setWorkingVars = <<
${user.erp1ip} := "192.87.106.34";
${user.erp1port} := "1812";
${user.erp1secret} := "some-secret";
${user.erp1retry} := "1";
${user.erp1timeout} := "7000";
${user.erp2ip} := "130.225.242.109";
${user.erp2port} := "1812";
${user.erp2secret} := "some-secret";
${user.erp2retry} := "1";
${user.erp2timeout} := "7000";
${user.loplip} := "192.168.200.1";
${user.loplport} := "1812";
${user.loplsecret} := "local-some-secret";
${user.loplretry} := "1";
${user.lopltimeout} := "7000";
${user.loplip} := "192.168.200.2";
${user.loplport} := "1812";
${user.loplsecret} := "local-some-secret";
${user.loplretry} := "1";
${user.lopltimeout} := "7000";
${user.eduroamcountry} := "at*bg*hr*cz*dk*ee*fr*fi*de*gr*hu*it*lv*lt*lu*mt*nl*no*pl*pt*ro*si*es*uk*ch";
${user.localcountry} := "*lo";

>>
ReadWrite-NewUser = "FALSE"
Method-On-Failure = "error:discardPolicyError"
Method-On-Error = "error:discardPolicyError"

getCountry
Method-Type = "PatternMatch"
Method-Disabled = "FALSE"
Method-On-Success = "setUserCountry"
PatternMatch-SearchKey = "${packet.User-Realm}";
PatternMatch-Mode = "REGEX"
PatternMatch-Operation = "MATCHES"
PatternMatch-Case = "(.+)\.(.+)$ ${user.country} = ${2};"
PatternMatch-IgnoreCase = "TRUE"
PatternMatch-SingleLine = "FALSE"
PatternMatch-MultiLine = "FALSE"
PatternMatch-Extended = "FALSE"
Method-On-Failure = "error:discardEmptyCountry"
Method-On-Error = "error:discardPolicyError"

setUserCountry
Method-Type = "ReadWrite"
Method-Disabled = "FALSE"
Method-On-Success = "checkdueroam"
ReadWrite-Map = "${user.countryf} := \"*${user.country[toLower]}\";"
ReadWrite-NewUser = "FALSE"
Method-On-Failure = "error:discardPolicyError"
Method-On-Error = "error:discardPolicyError"

checkdueroam
Method-Type = "Compare"
Method-Disabled = "FALSE"
Method-On-Failure = "checklocal"
Method-On-Success = "setProxyVarseduroam"
Compare-Input1 = "${user.eduroamcountry}";
Compare-Input2 = "${user.countryf}";
Compare-Operator = "contains"
Method-On-Error = "error:discardPolicyError"

checklocal
Method-Type = "Compare"
Method-Disabled = "FALSE"
Method-On-Success = "setProxyVarslocal"
Compare-Input1 = "${user.localcountry}";
Compare-Input2 = "${user.countryf}";
Compare-Operator = "=="
Method-On-Failure = "error:discardUnknownCountry"
Method-On-Error = "error:discardPolicyError"
In the file above enter the correct info about the eduroam and local servers. Also enter the list of TLDs participating in eduroam:

${user.eduroamcountry} – enter the list of TLDs participating in eduroam
${user.localcountry} – enter the local TLD

A.5  Microsoft Internet Authentication Service server as institutional server

(RADIUS) server and proxy. You can configure IAS in Windows Server 2003, Standard Edition, with a maximum of 50 RADIUS clients and a maximum of 2 remote RADIUS server groups. You can define a RADIUS client using a fully qualified domain name or an IP address, but you cannot define groups of RADIUS clients by specifying an IP address range. In the Enterprise and Datacenter Edition of Windows Server 2003 these limitations do not exist.

A.5.1 Installing IAS

Windows Server 2003 does not install IAS in the default installation. The IAS must be installed separately later from windows components under the **Networking Services**:

The **Internet Authentication Service** must be selected:
And wait for the installation to be finished. The IAS administrative console can be found under the Administrative Tools:

After clicking on “Internet Authentication Service” in the start menu the IAS console will start:
A.5.2 Configuring IAS to act as a university RADIUS server in the eduroam hierarchy

A.5.2.1 Configuring IAS for access points and upstream proxies

For each access point and upstream proxy (i.e. national eduroam RADIUS server) the parameters of the RADIUS Clients must be configured. When you add a new access point a wizard will start asking for the name and the IP address of the RADIUS client (i.e. Access Point, switch, or upstream RADIUS proxy):
Then you have to specify the shared secret between the RADIUS client and your RADIUS server (IAS):

You can select various vendors of RADIUS clients, but in most of the cases you should use RADIUS Standard.

A.5.2.2 Configuring Connection Request Processing Policy

The realm processing should be configured to match Eduroam hierarchy. First you have to configure a policy to catch local realms, then you have to configure a policy that forwards rest of the requests to your upstream proxy server.

A.5.2.3 Configuring policy for local realm

You should configure a Connection Request Processing Policy, that captures all the User-Names that are used for access to local realms using the policy condition ".@yourrealm.cc".
In this case the profile will be more complicated. The authentication should happen on the local server:
But the RADIUS attributes must be processed. In the case of a matching realm name the realm name must be stripped off:
A.5.2.4
Configuring policy for upstream RADIUS proxy server

You should configure a Connection Request Processing Policy, that captures all the User-Names that are potentially used for roaming with the policy condition “.*@.*”.

![Edit Profile](image)
Then you should edit the profile to be forwarded to the national proxy server:
You should configure first the remote RADIUS server group first in order to be able to select from the list.

### A.5.3 Configuring remote RADIUS servers

The national RADIUS proxy server must be added to the remote RADIUS server:
The remote RADIUS server address must be specified:

You have to enter the RADIUS server authentication port (usually 1812) and the shared secret of the remote RADIUS proxy server as well as the remote RADIUS server accounting port. You can specify different shared secrets for accounting if you wish:
A.5.4 Configuring Domain Users to be able to use the Eduroam with their credentials to Windows Domain

By default the users configured in the Windows Domain are not able to use their Windows Domain username and password to authenticate against IAS. This should be enabled in the Domain to allow access to Remote Access Permission. This can be done via the User Management interface or the Domain Manager interface with the following policy:
A.5.5 Configuration of Authentication methods

The authentication methods should be configured in the Remote Access Policies under the Profile settings. The absolute minimum that needs to be enabled is PEAP under the EAP methods, but it is useful to have PAP as well, for debugging purpose – at least for certain accounts (e.g. for test accounts):
PEAP is the easiest way to deploy eduroam authentication method under Windows. Deploying EAP-TLS can be labour-intensive:
A.5.6 Troubleshooting

The most useful information can be extracted from the Eventviewer:

But you can obtain these informations as well from the log files:
A.5.7 References


Appendix B Access Points

B.1 Cisco Aironet 1200 Series example setup

The configuration described in this section refers to the reference setup for all the VLAN's used.

List of configuration commands

NOTE: The following commands can be copy pasted into Access Point configuration mode (telnet or console access). The bold and italic bold text values need to be changed to match the implemented setup. The RADIUS Secret MUST match the secret configured on the RADIUS Software used and the Access Point's IP address (defined on the BVI interface) MUST be configured as an allowed Client in the used RADIUS Software.

Probably some stuff from the default AP configuration like RADIUS groups, default SSID (tsunami), etc has to be deleted. To erase/delete configuration parameters just

```plaintext
version 12.3
no service pad
service timestamps debug datatime msec
service timestamps log datatime msec
service password-encryption
!
hostname ap1200
!
```
logging buffered warnings
logging monitor warnings
enable secret 0 <your super passwd for accessing configuration level>

ip subnet-zero
no ip domain lookup

! aaa new-model

aaa group server radius radsrv
    server <your RADIUS Server IP address> auth-port 1812 acct-port 1813
    ! aaa authentication login default local
aaa authentication login eap_methods group radsrv
aaa authentication login mac_methods local
aaa authorization exec default local
aaa authorization network default group radsrv
aaa accounting send stop-record authentication failure
aaa accounting session-duration ntp-adjusted
aaa accounting update newinfo periodic 15
aaa accounting exec default start-stop group radsrv
aaa accounting network default start-stop group radsrv
aaa accounting network acct_methods start-stop group radsrv
aaa accounting connection default start-stop group radsrv
aaa accounting resource default start-stop group radsrv
aaa nas port extended
aaa session-id unique
no dot11 igmp snooping-helper
!
dot11 ssid eduroam
    vlan 909
    authentication open eap eap_methods
    authentication network-eap eap_methods
    authentication key-management wpa optional
    accounting acct_methods
guest-mode
!
dot11 ssid guest
    vlan 903
    authentication open
    accounting acct_methods
!
dot11 holdoff-time 60
dot11 location isocc PT cc 351 ac 21
dot11 ids eap attempts 32 period 8
dot11 network-map
dot11 arp-cache
!
username <your login> password 0 <your login password>
!
bridge irb
!
interface Dot11Radio0
    no ip address
no ip route-cache
!
encryption vlan 906 mode ciphers aes-ccm tkip wep128
!
encryption vlan 909 mode ciphers aes-ccm tkip wep128
!
broadcast-key vlan 906 change 600 membership-termination capability-change
!
broadcast-key vlan 909 change 600 membership-termination capability-change
!
ssid eduroam
!
ssid guest
!
countermeasure tkip hold-time 0
speed ofdm separate
speed basic-5.5 6.0 9.0 basic-11.0 12.0 18.0 24.0 36.0 48.0 54.0
power local cck 50
power local ofdm 30
no power client local
power client 50
fragment-threshold 512
station-role root fallback shutdown
rts threshold 2312
beacon period 500
beacon dtim-period 1
no dot11 extension aironet
world-mode dot11d country PT indoor
no cdp enable
dot1x reauth-period 300
bridge-group 1
bridge-group 1 subscriber-loop-control
bridge-group 1 block-unknown-source
no bridge-group 1 source-learning
no bridge-group 1 unicast-flooding
!
interface Dot11Radio0.903
encapsulation dot1Q 903
no ip route-cache
no cdp enable
bridge-group 3
bridge-group 3 subscriber-loop-control
bridge-group 3 block-unknown-source
no bridge-group 3 source-learning
no bridge-group 3 unicast-flooding
bridge-group 3 spanning-disabled
!
interface Dot11Radio0.906
description - eduroam VLAN for local users
encapsulation dot1Q 906
no ip route-cache
no cdp enable
bridge-group 6
bridge-group 6 subscriber-loop-control
bridge-group 6 block-unknown-source
no bridge-group 6 source-learning
no bridge-group 6 unicast-flooding
bridge-group 6 spanning-disabled

interface Dot11Radio0.909
description - eduroam VLAN for roamers (foreign users)
encapsulation dot1Q 909
no ip route-cache
no cdp enable
bridge-group 9
bridge-group 9 subscriber-loop-control
bridge-group 9 block-unknown-source
no bridge-group 9 source-learning
no bridge-group 9 unicast-flooding
bridge-group 9 spanning-disabled

interface FastEthernet0
no ip address
no ip route-cache
load-interval 30
duplex auto
speed auto
no cdp enable

interface FastEthernet0.902
description - Administrative VLAN
encapsulation dot1Q 902 native
no ip route-cache
no cdp enable
bridge-group 1
bridge-group 1 port-protected
no bridge-group 1 source-learning
bridge-group 1 spanning-disabled

interface FastEthernet0.903
description - guest VLAN - no internet connectivity on this network
encapsulation dot1Q 903
no ip route-cache
no cdp enable
bridge-group 3
no bridge-group 3 source-learning
bridge-group 3 spanning-disabled

interface FastEthernet0.906
description - eduroam VLAN for local students
encapsulation dot1Q 906
no ip route-cache
no cdp enable
bridge-group 6
no bridge-group 6 source-learning
bridge-group 6 spanning-disabled

interface FastEthernet0.909
description - eduroam VLAN for roamers (foreign students)
encapsulation dot1Q 909
no ip route-cache
no cdp enable
bridge-group 9
no bridge-group 9 source-learning
bridge-group 9 spanning-disabled
!
interface BVI1
description
ip address <your AP’s IP address> <your AP’s network mask>
no ip route-cache
!
no ip http server
no ip http secure-server
ip radius source-interface BVI1
!
no cdp run
radius-server attribute 8 include-in-access-req
radius-server attribute 32 include-in-access-req format %h
radius-server dead-criteria time 60 tries 10
radius-server host <your RADIUS server IP address> auth-port 1812 acct-port 1813
radius-server retransmit 2
radius-server deadtime 15
radius-server key 0 <your radius secret>
radius-server vsa send accounting
radius-server vsa send authentication
!
control-plane
!
bridge 1 route ip
!
!
line con 0
line vty 5 15
!
end

IMPORTANT: It is mandatory to save the configuration to the Access Point flash memory by executing the ‘write’ command in the normal command prompt, or the AP will boot with the previous saved configuration (of the previous boot)

Disclaimer: The presented Access Point configuration lacks of security measures as for the Access Point’s access to managing/configuring as for user communication (access-lists). Nevertheless the users associated to the access point (authenticated in SSID eduroam or just in the SSID guest) can’t access the managing interface (BVI) unless there is some VLAN routing elsewhere on the hotspot.
This configuration should be considered only as a starting point to make the access points eduroam compatible.

B.2 LANCOM L-54 Series Access Points

This series of Access Points offers a wide range of features for a mid-range price. One of the outstanding features in its price class is the ability to use ARP sniffing to determine a client’s IP address even if it changes during a user session. Activating this feature fulfills the requirement for MAC to IP correlation from the confederation policy and obsoletes logging of DHCP leases.
The following steps are needed to set up eduroam on a Lancom L-54 access point, describing setup via the web interface and is current as of LCOS Version 6.10:
B.2.1 NTP setup (confederation requirement: reliable timing source)

First select your Timezone under "Configuration - Date & Time – General":

( LANCOM L-54g Wireless 6.06.0012 / 27.03.2006)

Next choose “Synchronization” and check the radiobutton “Synchronize...”, then click on the link “Time Server” (NOT the menu “Time Server” on the left-hand side; this is only relevant if you want the AP to be the time server for its clients), click on add and enter your server details:
B.2.2 Logging

Select Configuration – Log & Trace – Syslog and check the box “Send information”, then click on “Syslog clients” and “add”, now add at least localhost: IP 127.0.0.1, activate all sources:

The logs that are collected with the localhost setting will show up under Expert Configuration – Status – TCP-IP - Syslog.
**B.2.3 Configuring the SSID**

Select Configuration – Wireless LAN – Logical WLAN setting – Network and click on one of the available slots, then set “WLAN network enabled” to “On” set the Network name (ssid) to eduroam and uncheck the box labeled “This radio network cannot be accessed with the network name “ANY” (closed network)”, then set MAC filter enabled to Off, Maximum count of clients to 0 and Client Bridge support to No. Optionally you can check the box “RADIUS accounting”:

![Logical WLAN settings](image)

**B.2.4 WPA Enterprise security**

First configure the RADIUS server to use. Select Configuration – WLAN security – IEEE 802.1X – RADIUS server click on add and enter your server details:
Next apply the RADIUS server and encryption scheme to the SSID eduroam. Select Configuration – WLAN security – 802.11i/WEP, click on WPA or Private WEP setting – 80211.i/WEP and click on the slot in which you previously configured the SSID eduroam. Then configure the following settings: “Encryption Activated” to “Activated”, “Method/Key 1 Length” to “802.11i(WPA)-802.1x”, “WPA Session Key Type” to “TKIP/AES” and “WPA Version” to “WPA1/2”. The other settings are irrelevant with WPA-Enterprise:
B.2.5 (optional) RADIUS accounting server

If you enabled RADIUS accounting for the eduroam SSID above, you have to configure a RADIUS server to send the accounting messages to. Go to “Expert Configuration” and select Setup – WLAN – RADIUS-Accounting and fill in your server details:

RADIUS-Accounting

- **Server-Address**: 158.64.
- **Account-Port**: 1813
- **Secret**: 
- **Backup-Server-IP-Address**: 0.0.0.0
- **Backup-Acct-Port**: 1013
- **Backup-Secret**
- **Client-Auth-Handling**: All-Traffic
- **Interim-Update-Period**: 300
- **Excluded-VLAN**: 0
Appendix C  Supplicants

C.1  SecureW2

Additionally to the SecureW2 part in the reference section, it is as well possible to use a non-preconfigured SecureW2 supplicant to connect to eduroam. For testing purposes or for very small institution where it does not seem worth the effort to prepare a preconfigured SecureW2 the following section illustrates the steps necessary.

Start the SecureW2 installer which can be downloaded from:

http://securew2.alfa-ariss.com/uk/download/index.htm

After starting the installer click on „next”, accept the EULA and click on „install” in the options panel. After successful installation you should see the following screenshot asking you to reboot:

![Completed the SecureW2 Client 3.1.2 Setup Wizard](image)

Now the wlan connection needs to be configured, Go to the network properties an right-click on „wireless networks”, where you choose properties as well. Now select „Wireless networks“ and click on add:

![Completed the SecureW2 Client 3.1.2 Setup Wizard](image)
Now enter the ssid „eduroam“ and choose WPA/TKIP as encryption. Not being able to select these might be due to two reasons: first the firmware of your wlan adapter might need to be updated, second you need to install the WPA patch for XP SP2. (Knowledge Base Article 893357).
Now click on the authentication tab and choose SecureW2 as EAP-Type. Then click on properties:

Click on „new“, select a meaningful profile name, preferably „eduroam“ and set your outer identity in the form yourname@yourrealm. It is possible to use anonymous as well but the usage of outer=inner identity is recommended.
Check the „Verify server certificate“ checkbox and add the certificates from your ca. Now check the „Verify server name“ and add the dns name of your RADIUS server.

Now click on the „Authentication“ tab, verify that „Authentication Method“ is set to pap and click on the „User account“ tab. Now enter your username in the format „yourname@yourrealm“, your password and you are almost done.

After clicking on „ok“ to close all open windows you are ready to connect. Click on the wlan network symbol in the task bar to see all wireless networks in your area, select eduroam an click on connect.
C.2 MacOS

Configuring the MacOS X Supplicant is pretty much straight forward, everything works right out of the box. To begin the configuration start the program “Internet Connect” which is part of the MacOS X default installation:
Select "802.1x" on the top bar and then choose "Edit Configurations" from the configuration pull down menu:

Now click on the "+" mark at the bottom of the configurations list to add the configuration for eduroam. Fill in a meaningful description, preferably "eduroam", and enter your username in the format user@realm and your password (the outer identity). Select eduroam as the wireless network (it is possible to just type the ssid in case there is no eduroam network available for selection in the pull down menu at the time of the configuration). Check the box selecting TTLS as the authentication protocol, and click on configure:
Leave „Inner authentication“ set to „PAP“ and enter „yourname@yourrealm“ as Outer Identity, as shown in the screenshot, it is possible to use anonymous as the outer identity as well:

After clicking „OK“ the client immediately tries to connect and given the circumstance that an eduroam network is available you are online:
C.3 WPA_Supplicant

Wpa_supplicant ([http://hostap.epitest.fi/wpa_supplicant/](http://hostap.epitest.fi/wpa_supplicant/)) is an open source 802.1x supplicant for Linux, BSD and MS Windows. This section describes its use on the Linux platform. Wpa_supplicant is available for most modern Linux distributions and seems to be the focal point of 802.1X development on the *nix platform.

It is out of scope of this cookbook to describe how wpa_supplicant can be compiled from source or what options need to be enabled in the Linux kernel to make eduroam authentication work. Modern Linux distributions with standard kernel, wireless tools and wpa_supplicant should work “out of the box”.

With below technical information it is possible to implement eduroam support so that it will be seamlessly integrated with the OS, this is however very distribution specific and therefore out-of-scope for this document.

This chapter shows the basic elements of the configuration and provides simple means of configuring eduroam in a universal way. It is assumed that the user has a working wireless card (this can be verified by using the iwconfig command).

wpa_supplicant is responsible for the (layer 2) authentication of the user, and must be followed by some means of setting up the (layer 3) IP connection by using a DHCP-client. Wpa_supplicant typically runs in the background controlling the connection, taking care of re-authentications, roaming between access points, etc. It is started with the command:

```
wpa_supplicant -i interface -c configuration_file -D driver -B
```

where `interface` is the system name for the wireless interface (like eth1, ath0, wlan0, etc.), `configuration_file` is the location of the file, that will be described later on, `driver` is one of: wext, ipw, madwifi and ndiswrapper (described below) and `-B` option means ‘run in the background’.

The driver setting depends on the particular card used.
The wext diver currently supports most of existing cards (Atheros chipset based cards being an exception, madwifi should be used there). Hence, the wext setting should be tested first.

The configuration file depends on the EAP type of choice. Example configurations are provided for EAP-TTLS, EAP-PEAP and EAP-TLS. Each of the examples contains two, nearly identical, network blocks, the only difference is that one is for WPA and the second for dynamic WEP.

In principle, one block with ‘key_mgmt=WPA-EAP IEEE8021X’ should be sufficient, but it has been found out that under certain conditions this may fail, whereas two separate blocks seem to work correctly.

The `ca_cert` points to the certificate file of the CA which has provided the certificate for the RADIUS server. This file should contain certificates for the whole certification chain, up to the root. All certificates and keys should be in PEM format.

```
# EAP-TTLS configuration
ctrl_interface=/var/run/wpa_supplicant

network=
    ssid="eduroam"
    key_mgmt=WPA-EAP
    ca_cert="/etc/eduroam/ca.cer"
    identity="user@your.domain"
    eap=TTLS
    password="xxxx"
    phase2="auth=PAP"
}

network=
    ssid="eduroam"
    key_mgmt=IEEE8021X
    ca_cert="/etc/eduroam/ca.cer"
    identity="user@your.domain"
    eap=TTLS
    password="xxxx"
    phase2="auth=PAP"
}
```
Note: this example will set the outer identity to be the same as the real, inner identity of the user. It is possible to set the outer identity to a different name (for instance to an opaque id), but for simplicity this is not shown here.

# EAP-PEAP configuration
ctrl_interface=/var/run/wpa_supplicant

network={
    ssid="eduroam"
    key_mgmt=WPA-EAP
    ca_cert="/etc/eduroam/ca.cer"
    identity="user@your.domain"
    eap=PEAP
    password="test"
    phase2="auth=MSCHAPV2"
}

network=
    ssid="eduroam"
    key_mgmt=IEEE8021X
    ca_cert="/etc/eduroam/ca.cer"
    identity="user@your.domain"
    eap=PEAP
    password="test"
    phase2="auth=MSCHAPV2"

# EAP-TLS configuration
ctrl_interface=/var/run/wpa_supplicant

network=
    ssid="eduroam"
    key_mgmt=WPA-EAP
    ca_cert="/etc/eduroam/ca.cer"
    identity="user@your.domain"
    eap=TLS
    client_cert="/etc/eduroam/user.crt"
    private_key="/etc/eduroam/user.key"
    private_key_passwd="xxxx"
}

network=
    ssid="eduroam"
    key_mgmt=IEEE8021X
    ca_cert="/etc/eduroam/ca.cer"
    identity="user@your.domain"
    eap=TLS
    client_cert="/etc/eduroam/user.crt"
    private_key="/etc/eduroam/user.key"
    private_key_passwd="xxxx"

Note: Most wpa_supplicant compilations will accept user key/certificate in one PFX (p12) file. If that is used, this file should be pointed to by private_key and client_cert should be commented out.

The script provided below starts and stops the eduroam connection. It needs to be configured by assigning correct values to the variables in the configuration section. The script kills possible wpa_supplicant processes, and DHCP clients for the particular interface. Then it starts wpa_supplicant and monitors its state with wpa_cli. If no authentication takes during the REAUTH_TIMEOUT period, wpa_supplicant is restarted. After authentication, the DHCP client is started.
#!/bin/sh
#
# WPA_SUPPLICANT="/sbin/wpa_supplicant"
# WPA_CLI="/sbin/wpa_cli"
# DRIVER="wext"
# WPA_CONF="/etc/eduroam/wpa_supplicant.conf"
# DHCPD="/sbin/dhclient"
# INTERFACE="eth1"
# REAUTH_TIMEOUT=40
# end of configuration section

dhcclient=`basename $DHCPD`
case "$1" in
  start)
    echo "starting network on ${INTERFACE}"
    pkill wpa_supplicant
    kill `ps -ef | awk "$/dhclient/ && /eth1/ && ! /awk/ {print $2}"` 1>/dev/null 2>&1
    ${WPA_SUPPLICANT} -B -D ${DRIVER} -c ${WPA_CONF} -i ${INTERFACE} -P
      /var/run/wpa_supplicant.pid 1>/dev/null 2>&1
    if [ "$WPA_CLI" ] ; then
      i=1
      echo "waiting for connection"
      while ! $WPA_CLI status | grep -q AUTHENTICATED ; do
        sleep 1
        i=`expr $i + 1`
      done
    else
      sleep 10
    fi
    echo "setting IP"
    ${DHCPD} ${INTERFACE}
  ;;
  stop)
    echo "stopping network on ${INTERFACE}"
    pkill wpa_supplicant
    kill `ps -ef | awk "$/dhclient/ && /eth1/ && ! /awk/ {print $2}"` 1>/dev/null 2>&1
    echo "Usage $0 {start|stop}"
    exit 1
  ;;
)*
esac

This script has to be run with administrator's rights, which is not very convenient. With a little extra effort one can create wrappers which can then be connected to panel buttons, and so the network can be started and stopped by clicking with the mouse and providing the administrator's password. Since, for a casual user, finding the right values of all variables, creating the correct configuration file, creating correct wrapper scripts may be far too complicated, an alternative way has been proposed and implemented in a form of a simple package, which can be downloaded from:

http://eduroam.pl/Files/prepare_eduroam_config.tgz.
This utility allows campus administrators to create a configuration script that can be distributed to the users. The script contains all necessary certificates, scans the system for the needed tools, creates configuration files, certificate files, sets up the main starting script and wrappers. A full description is beyond the scope of this document, but can be found in the documentation of the package.
Copyright

The contents of this document, including but not limited to all design, text and images, are owned, except as otherwise expressly stated, by the partners in the GÉANT2 project (http://www.geant2.net/), members of the European National Research and Education Network (NREN) Consortium, including TERENA and DANTE Ltd and may not be copied, reproduced, transmitted, displayed, distributed, sublicensed, altered, stored for subsequent use or otherwise used in whole or in part in any manner without the prior written consent of DANTE Ltd, acting under control of and on behalf of the said Consortium, except for one temporary copy in a single computer’s memory and one unaltered permanent copy to be used by the viewer for personal and non-commercial use only, with an attached copy of this page containing this notice.