Create. Connect. Control.

anyweb

Manual

AnyRover V3

Project	
Date	6 March 2023
Status	Public
Version	2.0.20
Author(s)	Marco Wirz
	Manuela Bachmann
	Armin Weiss
Distribution	

AnyWeb AG
Hofwiesenstrasse 350, 8050 Zürich, Switzerland, Phone +41 58 219 11 11
www.anyweb.ch

Table of contents

1 Overview

1.1 Description

The AnyRover is a high speed 4G router. It contains a 4 port switch, a LTE modem, a GPS receiver, and a USB and several general purpose IO (GPIO) connections. The device can be extended with power over Ethernet (PoE), wireless LAN and DSL.

The AnyRover can connect to the Internet through the LTE interface, and provide this link to the connected devices.

1.2 Hardware

Element	Specification	Possible extensions
Processor	ARM9, 800MHz Dual Core	ARM9, 800MHz Quad Core
RAM	1GB	
Flash	NAND Flash, 1GB	
Serial console	RS-232, 115200 8N1	
Switch	100Mbit, 5 Port (4 external, 1 internal)	Supports VLANs
Power over Ethernet 802.3af	PSE: Power Sourcing Equipment PD: Powered Device	Total 2 Ports, both as PSE or PD
LTE/HSPA/GPRS Modem	Huawei ME909s-120P	2x Huawei ME909s-120P
SIM-Card	2 external slots	
USB Port	USB2.0 HiSpeed	
GPS receiver	u-blox 8 Multi-GNNS Receiver	u-blox 8 Multi-GNNS 3D-Dead Reckoning Receiver
SD-Card Slot	Support for SDHC up to 32 GB	
Multipurpose inputs	3 digital or analog Inputs with 10bit ADC	
Multipurpose output	1 digital Output, max. 1.8A	
RTC	With supporting battery	
Wireless LAN	IEEE 802.11 a/b/g/n	2x IEEE 802.11 a/b/g/n
TPM		

1.3 Software

Many of the programs used are licensed under the GPS or the LGPL. The licenses are reproduced in the appendices. The source code of the respective programs can be obtained from AnyWeb. This table lists the programs used.

Function	Program	License	Website
Operating system	Linux Kernel 4.10	GPL	www.kernel.org
Command line tools	Busybox	GPL	www.busybox.net
SSH Client and Server	Busybox	GPL	www.busybox.net
Client and Server	Busybox	GPL	www.busybox.net
Firewall	IPtables	GPL	www.netfilter.org
DHCP Server	Busybox	GPL	www.busybox.net
DynDNS Support	inadyn		
PPP connection	PPPd / chat	GPL, BSD, Public Domain	
IPsec	StrongSwan	GPL	www.strongswan.org
OpenVPN	openvpn	GPL	www.openvpn.net
Web server	boa		www.boa.org
FTP Server	Busybox	GPL	www.busybox.net
TFTP Server and Client	Busybox	GPL	www.busybox.net
NTP Server and Client	ntpd		ntp.isc.org
Cron Jobs	Busybox	GPL	www.busybox.net
SMS console	gpio_daemon	Developed by AnyWeb	
Editors vi and nano	Busybox, nano	GPL	
WLAN client (opt.)	wpa_supplicant	GPL / BSD	hostap.epitest.fi/wpa_supplicant
WLAN Access Point (opt.), RADIUS server	hostapd	GPL / BSD	hostap.epitest.fi
OSPF daemon	quagga	GPL	www.quagga.net
SNMP daemon	net-snmp	div BSD	www.net-snmp.org

1.4 Libraries and tools

Several libraries and tools are used in the AnyRover. This tables provides details.

Library / Tool	License	Web site
uClibc	LGPL	www.uclibc.org
USL	LGPL	opensource.katalix.com/openl2tp/
iproute2	GPL	www.linux-foundation.org/en/Net:Iproute2
libnl	LGPL	people.suug.ch/~tgr/libnl/
libpcap	BSD	www.tcpdump.org
libncurses	MIT license	www.gnu.org/software/ncurses/ncurses.html
tcpdump	BSD	www.tcpdump.org
wget	GPL	www.gnu.org/software/wget/

2 Short overview or only cowards read manuals

For operation, the AnyRover must be connected to the power supply. Input voltage is in the range of 8..52V DC. A LTE and a (passive) GPS antenna must be connected to the respective sockets.

With the default configuration, a SIM card from Swisscom can be inserted, and through the Ethernet interface (using DHCP), the internet is available shortly thereafter.

Access to the device is through the console using a standard Cisco console cable with the parameters 115200 8N1. As an alternative, access is possible over the network using SSH (Ethernet and 3G/4G) or telnet (only Ethernet).

Login as user config, password cabtronix, or user root, password root. The config user can only modify the configuration and show some system stats (help shows the available commands), the root user has complete access and should act cautiously.

Configuration is done in one central file /etc/cablynx.conf. The default configuration is also saved in /etc/conf.d/cablynx.factory. On the device, two editors are installed, vi and nano (if you know neither, use nano!). The configuration file is extensively commented. If the configuration is edited as user config, the system will update right after terminating the editor, when working as user root, the update has to be started manually (reboot or /etc/init.d/rcS config).

For security reasons the passwords should be changed before putting the device in a productive environment (use the command passwd both for user root and config).

The full documentation is stored as a PDF file in English and German on the device in the /root directory and on the webpage www.anyrover.ch

3 Operation of the AnyRover – Hardware

3.1 Front side

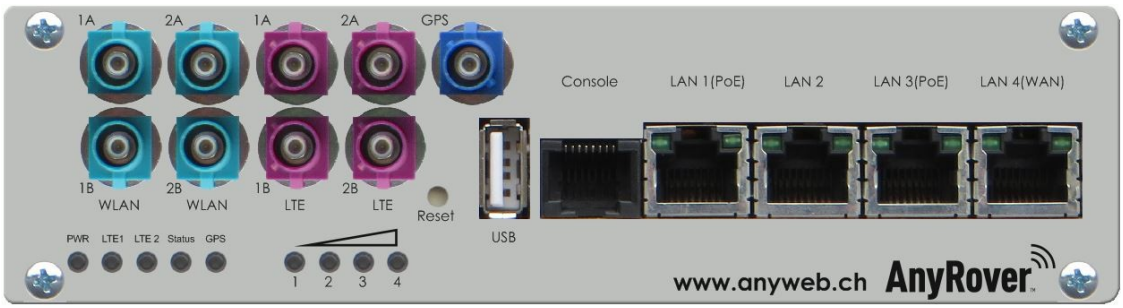


Figure 1: AnyRover V3 front side

3.1.1 Antennas

The AnyRover has two to four antenna connectors that can be supplied with SMA or Fakra connectors. The GPS receivers requires an active GPS antenna with 3V supply (passive available on request), the GSM modem requires an antenna that is UMTS and LTE capable. For WLAN and Modem there are two connectors available to use MIMO technology.

GPS: Center Frequency 1575.42MHz, Bandwidth ±1.023MHz, Impedance 50Ω

WLAN: Frequency: 2.4GHz, 5GHz

M1 (modem): Frequency Range: 824-960MHz, 1710-1880MHz, UMTS 1900-2170MHz, LTE 800-2600MHz, Impedance 50Ω

A suitable combination antenna is the model CT-AT104m from Celphone (www.celphone.ch).

Warning: When connecting a GPS antenna to the GSM connector, the antenna can be destroyed.

3.1.2 USB

The USB connector accepts any USB2.0 HiSpeed devices. By default, connected memory sticks will automatically be mounted into the system.

Basically, any USB device can be operated, as long as a driver for the Linux kernel is available. Depending on the device, the driver must be installed manually, and the configuration for the device implemented.

3.1.3 Console

The console connector provides access to the system console, which is a serial RS-232 interface, accessible through a Cisco console cable. The description of the pins is shown in Table 1: Description of the pins of the console. Baud rate is 115200 8N1.

Pin	Function
1	CTS
2	NC
3	TxD (AnyRover → PC)
4	Gnd
5	Gnd
6	RxD (PC → AnyRover)
7	NC
8	RTS

Table 1: Description of the pins of the console

Security warning: Through the system console, complete system access is possible.

3.1.4 Network

The four network ports are identical as per default configuration. It is possible to configure VLAN such that the ports are in different logical nets.

The 2 ports 1 and 3 can be equipped with PoE modules, both ports either as PSE (AnyRover supplies the peripheral with energy) or as PD (AnyRover is supplied through PoE).

3.1.5 LEDs

There are four LEDs for power, modem, status and gps to show the current state of the AnyRover. The other four LEDs are freely configurable. With the default configuration they show the signal of the mobile network.

3.1.6 Mode and Reset Button

The two buttons can be configured in the cablynx.conf file. In the default configuration, the reset button resets the configuration to default when it is pushed between 2 and 5 seconds. When it is pushed for more than five seconds, the AnyRover performs a reboot. The mode button has no function in the default configuration. On devices with MIMO antenna connectors, there isn't a mode button.

3.2 Back side

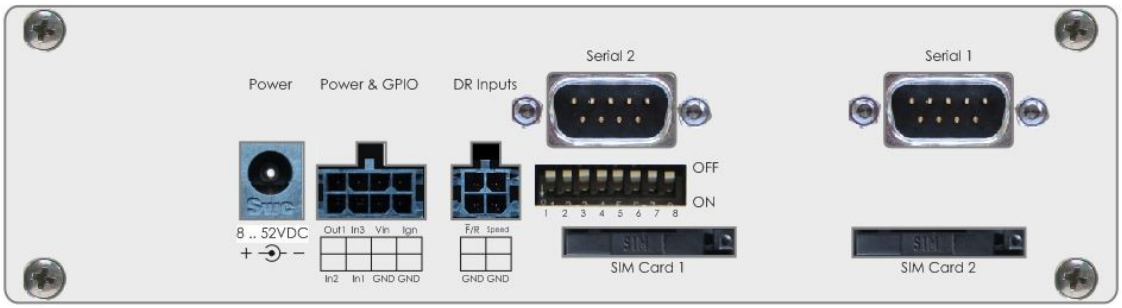


Figure 2: AnyRover V3 back side

3.2.1 Power

Power supply of the AnyRover is established either through the DC power or the GPIO connector.

The power connection is a common power supply with a 5.5mm connector with a length of 9.5mm. The positive connector is in the center, the pin has a diameter of 2.1mm. Input voltage can range from 8V to 52V DC. The output power of the power supply is recommended to be at least 10W for devices without power over Ethernet. If there are PoE devices connected, then the output power should be at least 50W with an output voltage of 10V or more.

If the AnyRover is supplied through the round power connector, the device remains switched on independent of the state of the ignition signal.

3.2.2 GPIO connector

This connector allows the supply of the AnyRover as well as automatically switching the device on and off depending on the ignition line of the vehicle. The pin placement is indicated in figure Figure 3: GPIO connector with labelled pins.

Assembled cables are available, for specific requirements, these components can be used:

Manufacturer / series	Molex / MicroFit 3.0
connector housing:	43025-0800
crimp connections:	43030-0009
crimp tool:	63911-2800
contacts ejection tool:	11-03-0043

Figure 3: GPIO connector with labelled pins

Pin number	Function	connector DIN 72552	Wire color	Remarks
1	GND	31	blue	Either one or both ground connections can be used.
2	GND	31	brown	Either one or both ground connections can be used.
3	Input 1	-	yellow	Digital switching level at approx 4.6V (raising) and 2.0V (falling). Analog measure interval: 0 ... 6.6V. Input impedance: 94kΩ.
4	Input 2	-	orange	Digital switching level at approx 4.6V (raising) and 2.0V (falling). Analog measure interval: 0 ... 6.6V. Input impedance: 94kΩ.
5	Ignition	15	black	Switching level: switch on at approx 2.0V, ignition detection at approx 4.6V (raising) and 2.0V (falling). Input impedance: 20kΩ. To keep the device always on, connect this pin to Vin.
6	Vin 8..52V	30	red	Standby current: <100µA. Supply current 12V / 24V without PoE: <460mA / <230mA. Supply current 12V / 24V with 2x PoE: <1.9A / <950mA.
7	Input 3	-	purple	Digital switching level approx 4.6V (raising) and 2.0V (falling). Analog measure interval: 0 ... 6.6V. Input impedance: 94kΩ.
8	Output	-	green	Switches to Vin. Guaranteed current of at least 1.8A.

Table 2: Pin configuration of the GPIO connection

When operation in a vehicle, the AnyRover can switch off automatically when the ignition line is low. This way, the car battery can be protected. As soon as the ignition line raises above 4.6V again, the AnyRover is switched back on.

3.2.3 DR inputs (only for dead reckoning variants)

Assembled cables are available, for specific requirements, these components can be used:

Manufacturer / series	Molex / MicroFit 3.0
connector housing:	43025-0400
crimp connections:	43030-0009
crimp tool:	63811-2800
contacts ejection tool:	11-03-0043

Dead reckoning GPS combines pure satellite navigation with a gyroscope, the tachometer and an indicator signal for forward / backward motion. This way, position finding can be continued even if no satellites are visible, e.g. in tunnels.

Certain preconditions must be met in order for dead reckoning to work reliably. The device orientation (see Error: Reference source not found) as well as voltage levels, polarities and signal properties must be correct. Otherwise, the results may, in certain cases, be worse than without dead reckoning.

The system works best if the tachometer impulse is taken from the rear axle, and the GPS antenna is positioned above the rear axle as well. The source of the tachometer impulse

may not be changed with reasonable effort, but with a good placement of the antenna, a lot can be gained, especially in long vehicles.

Special care has to be taken for vehicles containing a trip recorder. Because this device is calibrated, signals are often taken from there or from a trip recorder simulator (e.g. Siemens/VDO „TSU 1391"). It is possible that these devices deliver individual pulses even if the vehicle is stationary which renders the measurements of the dead reckoning system completely unusable.

The tachometer signal must be proportional to the cruising speed – even if the vehicle is stationary, e.g. it must not deliver any impulses to the AnyRover in this case. Additionally, the signal must reach certain voltage levels during the impulse to be recognized by the AnyRover. These levels are 4.6V (low to high) and 2.0V (high to low). To be on the safe side, try to obtain the pulses directly from the pulse generator, if the necessary levels are reached.

The pin assignment of the DR connector is given in Figure 4: DR connector with pin assignment and Table 3: Pin assignment of the DR connector.

Figure 4: DR connector with pin assignment

Pin number	Function	Pin DIN 72552	Wire color	Remarks
1	GND	31	black	Connected to gnd, use if required.
2	GND	31	green	Connected to gnd, use if required.
3	Speed Tick / tacho signal	-	white	High level: +4.75 V bis +30 V Low level: -30 V bis +1.5V Impulse: 1 impulse/meter up to 50 impulses/meter Input impedance at least10kΩ.
4	Forward / Reverse	-	red	High level: +4.75 V bis +30 V Low level: -30 V bis +1.5V

Table 3: Pin assignment of the DR connector

The forward/reverse signal must be a switched voltage of the reversing light. 12V or 24V corresponds to reverse cruise, and 0V to forward cruise. If the signal is inverted, it can be changed in the software configuration.

The AnyRover only draws at most 2mA@24V and 1mA@12V vehicles. There are no known devices that cannot provide this current.

Important: After installation, the GPS calibration has to be reset and a calibration drive has to be performed! See chapter Error: Reference source not found

3.2.4 Serial ports (optional)

Serial1 and Serial2 are the serial ports according to EIA/RS232. When used as normal COM ports both ports support all common baud rates up to 115200 Bits/s.

Serial1 is without HW-Handshake. It can get the GPS data directly from the GPS receiver with baud rate 57600 8N1 (activation in cablynx.conf).

Serial2 has a HW-Handshake by RTS/CTS.

The following table displays the configuration of the COM ports:

Pin	Serial1 function	Serial2 function
1	NC	NC
2	RxD (Device → AnyRover)	RxD (Device → AnyRover)
3	TxD (AnyRover → Device), GPS TxD	TxD (AnyRover → Device)
4	NC	NC
5	Gnd	Gnd
6	NC	NC
7	NC (GPIO, ask if used)	RTS (AnyRover → Device)
8	NC (GPIO, ask if used)	CTS (Device → AnyRover)
9	NC	NC

Table 4: COM ports pin configuration

3.2.5 DIP Switch

The eight switch at the back of the device are made to make various configurations around the dead reckoning function. Switch up means OFF, switch down means ON.

Switch	Function	Resulting mode
1 & 2	Mounting Orientation	See chapter Vehicle Integration

3	Disable Dead Reckoning Function	OFF: DR is enabled ON: DR is disabled
4	Invert Reward Signal	OFF: high = reward, low = forward ON: high = forward, low = reward
5	DR Pulse Filter	OFF: Pulses <100 µm are getting filtered out ON: Pulses <500 µm are getting filtered out
6		
7	Input1 = Speedticks	OFF: Input1 and Speedticks are separated ON: Input1 is connected to Speedtick signal
8	Input 2 = Forward/Reward Signal	OFF: Input2 and Fwd/Rwd signal are separated ON: Input2 is connected to Fwd/Rwd signal

Tabelle 5: DIP Switch Settings

3.2.6 SIM card

The internal modem cannot create a connection without a SIM card. PIN protected SIM cards are supported, the PIN code has to be saved in the configuration file. Different PIN codes such as PIN2 or PUK are not supported. In case the SIM card requires one of those codes, manual intervention is necessary, by either removing the SIM card and entering the PIN using a mobile phone (recommended), or by entering the necessary commands on the command line. The system logs this case in the system log file.

3.3 Internal connections

Usually, the AnyRover does not have to be opened, all necessary connections are accessible from the outside. Some modifications require the device to be opened though. Before opening, the device has to be shut down, and power supply has to be disconnected.

To open the device, a philips screw driver is required.

3.3.1 MicroSD Card

The AnyRover can be supplied with a MicroSD card to extend hard disk space. To replace a MicroSD card, the back cover has to be removed (where the power connectors are).

3.3.2 Modem

The AnyRover can be operated with different modem types. To change the modem, the top cover has to be removed. After inserting the modem, make sure to properly connect the antenna cable to the modem to ensure good signal quality. The antenna cable must be handled carefully and must not be bent.

The modem is a mini PCI express card in standard format (30x56mm). When using a supported modem, no software change is required after changing the modem. A supported modem ist the Huawei ME909s-120P.

The AnyRover does not support voice capabilities of the modems.

3.3.3 PoE

To change the PoE modules, the top cover has to be removed. The PoE modules are placed behind the Ethernet ports. No software change is necessary after changing the PoE configuration. To switch the PoE PSE modules on or off, a configuration change might be necessary.

3.3.4 Wireless LAN

The AnyRover can be equipped with one or two wireless LAN cards. The cards are connected through USB.

The default card is from DeLock (www.delock.de), with a Ralink chip set. The card supports IEEE 802.11b/g/n and can be operated both as client and as access point.

The card is placed between the processor module and the back of the device, two holes in the PCB are already present. The connection is established with a cable to the internal USB connector right behind the external USB socket. For the WLAN antenna, a hole is already spared in the front plate.

The card is attached on the main board with suitable screws and distance bolts. The card must be placed high enough above the PCB to avoid any contact between the components.

Other WLAN cards can also be used as long as they are connected via USB. With different cards, a suitable attachment must be found, and possibly the driver manually inserted into the system, if the card uses another chip set than Ralink RT73 or RT2800.

3.4 Vehicle integration

A properly planned and cleanly executed installation can prevent many problems later on, therefore this should be given proper attention.

3.4.1 Installation location

The selection of the location in the vehicle where the AnyRover is installed can be determined according to these factors:

Length of antenna cabling

Try hard to keep antenna cables as short as possible. For active GPS antennas, losses are only encountered if cable attenuation reaches the range of antenna gain (depending on antenna and cable this is the case for lengths of 10 to 20m). For passive GPS antennas and for all GSM, UMTS, and WLAN antennas, every meter of cable decreases range. For large vehicles like passenger buses, placing the AnyRover above the windows and close to the antenna is highly recommended.

Vibration, heat and dirt
All these factors diminish the live expectancy of the AnyRover. In vehicles, temperatures in the range of 70°C can easily occur, even more in the engine compartment. For dead reckoning devices, heavy vibrations can distort measurements of the gyroscope. An installation in the engine compartment is therefore not recommended.

Accessibility of signal and supply pins
One or two supply voltages (switched and/or battery voltage), ground and – for dead reckoning systems the tachometer impulse and reverse signal – are necessary for proper installation of the AnyRover. Installation of the cabling can be time consuming and may thus have an impact on installation.

Device alignment (non dead reckoning devices)
Basically, the AnyRover can be installed in any orientation, but it is better not to orient the connectors towards the top, since dirt can then assemble in the connectors.

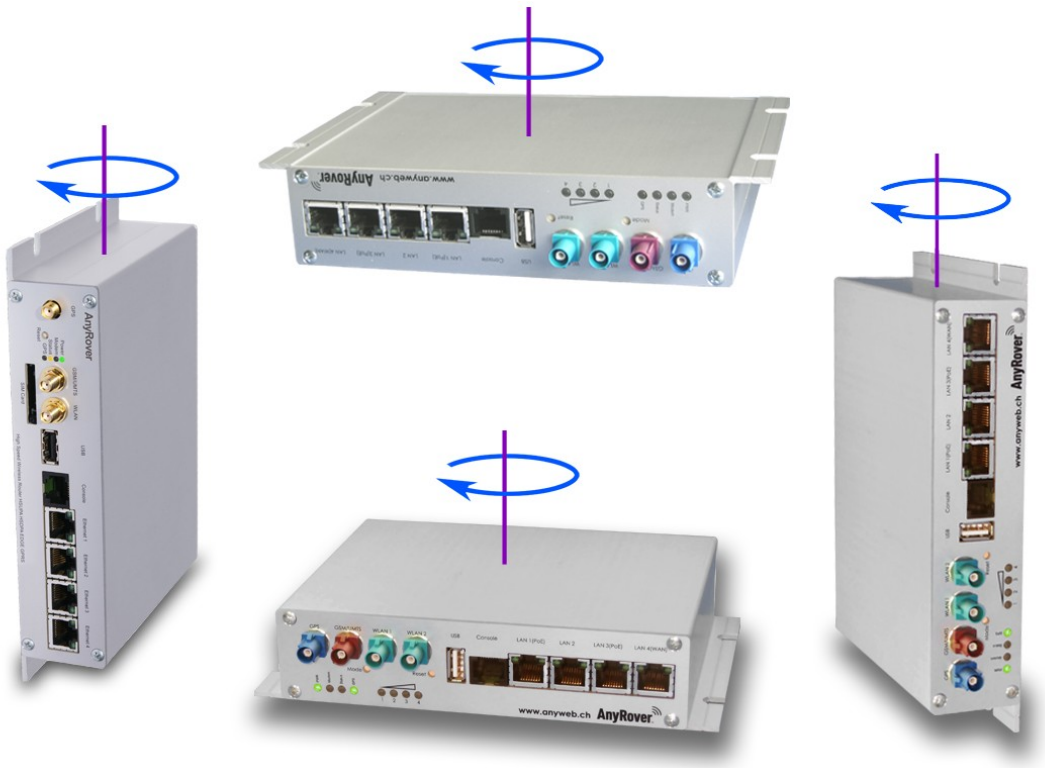


Figure 5: Possible device orientations for dead reckoning systems

Device alignment for 3D dead reckoning systems
With a new 3D dead reckoning GNSS (GPS, GLONASS and Galileo), the AnyRover V3 can be installed in any orientation. There are no more limitations like there were on the older

models. For a maximal precision, one of the six sides should lie parallel to the ground. But there will be no big difference to any other alignment.

3.4.2 Antenna installation

Several problems with localization and disposition systems can be traced back to unsuitable antenna installation locations. Individual wrong position calculations that are off by several hundred meters up to several kilometres occur with every GPS receiver, but are usually filtered in the system. Even so, good signal reception is essential for good precision and should be treated accordingly.

Basically, a good antenna position and quality is one of the most important factors for successful operation.

Cabling

Cables must not be bent to a smaller radius than 20mm. Also, they must not be able to chafe anywhere even if there are movable parts involved (e.g. trunk lid).

Mounting of SMA connectors: Pushing the cable slightly into the socket eases tightening the nut. The maximal moment for tightening the nuts is 0.2Nm, which corresponds to a force of 1.5N or 150g on a 12cm wrench.

Installation locations

The more visibility the antenna has towards the sky above the better GPS reception will be. The signal will pass unimpeded through uncoated glass and most plastic materials, but not through metal sheets.

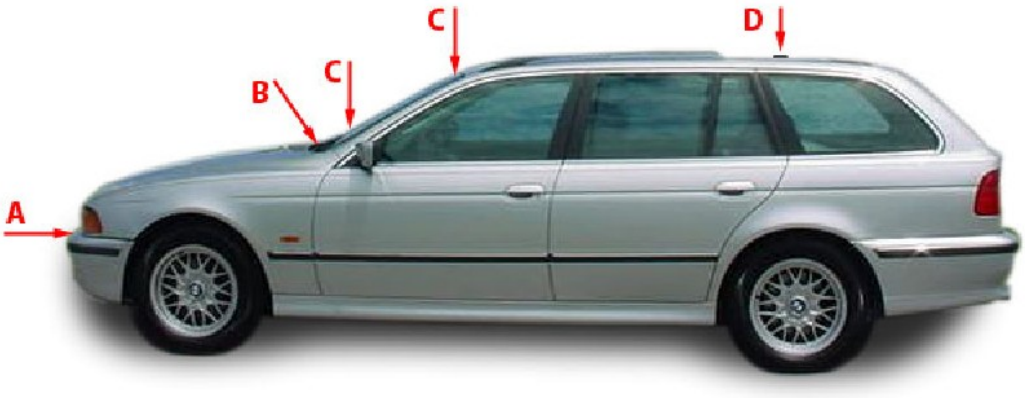


Figure 6: Possible mounting points for GPS antenna

3.4.3 Set installation position and backup signal

Unlike the older AnyRover Models, there is no need to set the installed alignment manually. When driving, this happens automatically, when the tachometer signal and

the polarity of the back driving signal are applied correctly.

3.4.4 Reset 3D dead reckoning alignment calibration

It is recommended to reset the dead reckoning alignment calibration whenever a device is used in a different vehicle. If this is not done, the GPS Position can differ highly from the actual position on the first few kilometres. To reset the alignment calibration, push the reset button for more than 10 seconds.

3.4.5 Optional Calibration ride

After the installation, the 3D dead reckoning receiver has to calibrate its alignment and the vehicle signals. When a calibration reset is done, this should work without any problem. A specific calibration ride is not necessary. The following procedure is optional, but it may help to recognize and solve problems with the calibration.

Note: To guarantee a fast and easy calibration, it is recommended to wait until all satellite orbits are actual. When the AnyRover is on and receives GPS informations (GPS LED blinks once a second), this is done after about 12 Minutes standing in the field.

Note: In Europe the constellation of the GPS-Satellites is not optimal. Between 14:00 and 17:30 a limited view southwards may prevent getting a GPS-Position. This is not because of a limited signal strength but rather because the few visible satellites are arranged in a line and prevent a three dimensional determination of the position. However, in an open field, this should not lead to problems.

The following image shows the Status on the AnyControl tool when the 3D dead reckoning GPS receiver is not calibrated yet:

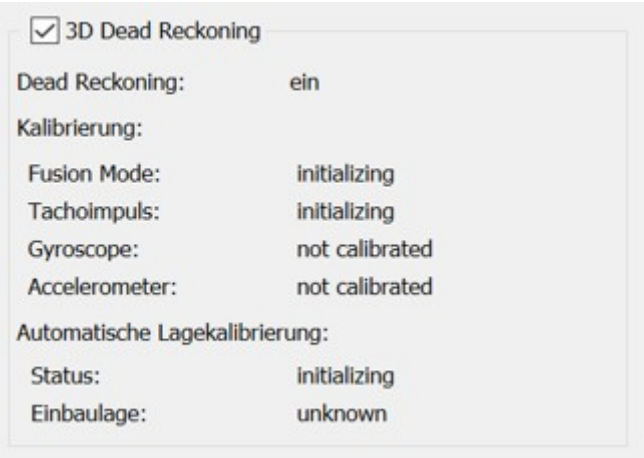


Figure 7: AnyControl 3D DR status not calibrated

The first step is to get the alignment calibration. The easiest way to do this is to drive on a straight road with accelerating and breaking. After that, a long curve in both directions and stable speed helps to get its alignment. This corresponds to the manufacturers instruction to "Add some dynamics".

Note: Please respect the provided speed limit and environmental conditions and do not perform any dangerous manoeuvres to calibrate the receiver.

After some of the described manoeuvres, the notification in the AnyControl „Automatische Lagekalibrierung / Status" changes to „coarse" and „Tachoimpuls" changes to „initialized":

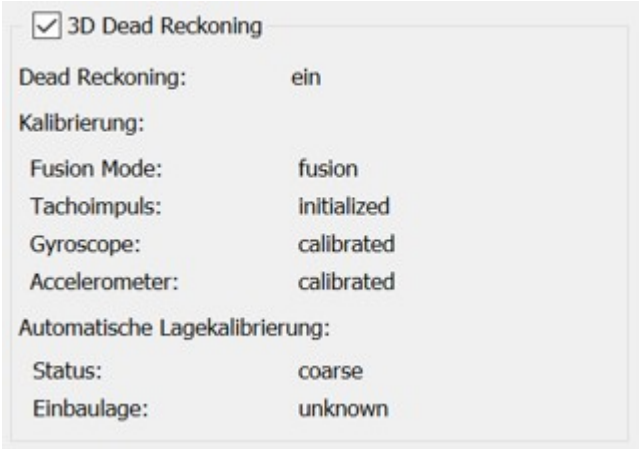


Figure 8: AnyControl 3D DR status coarse

After that, the receiver can proceed with the calibration of the gyroscope. When this is done "Fusion Mode changes to "fusion" and "Gyroscope" to "calibrated". To assist the process, drive along a straight road with a roundabout at its end for about 1km, take an extra round at the roundabout and return on the same straight road.

After a few days in use, the notification "Automaische Lagekalibrierung / Status" will change to "fine":

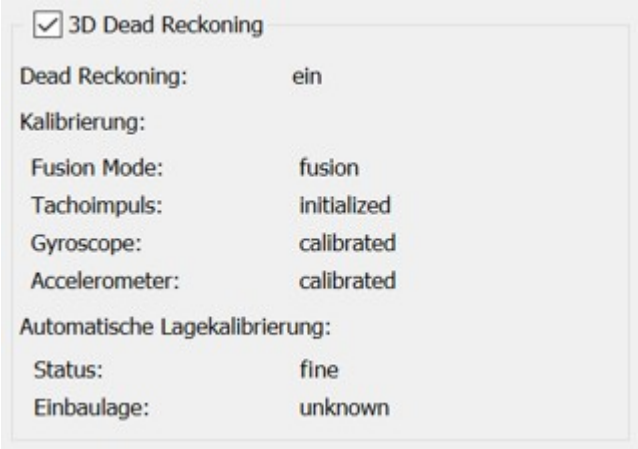
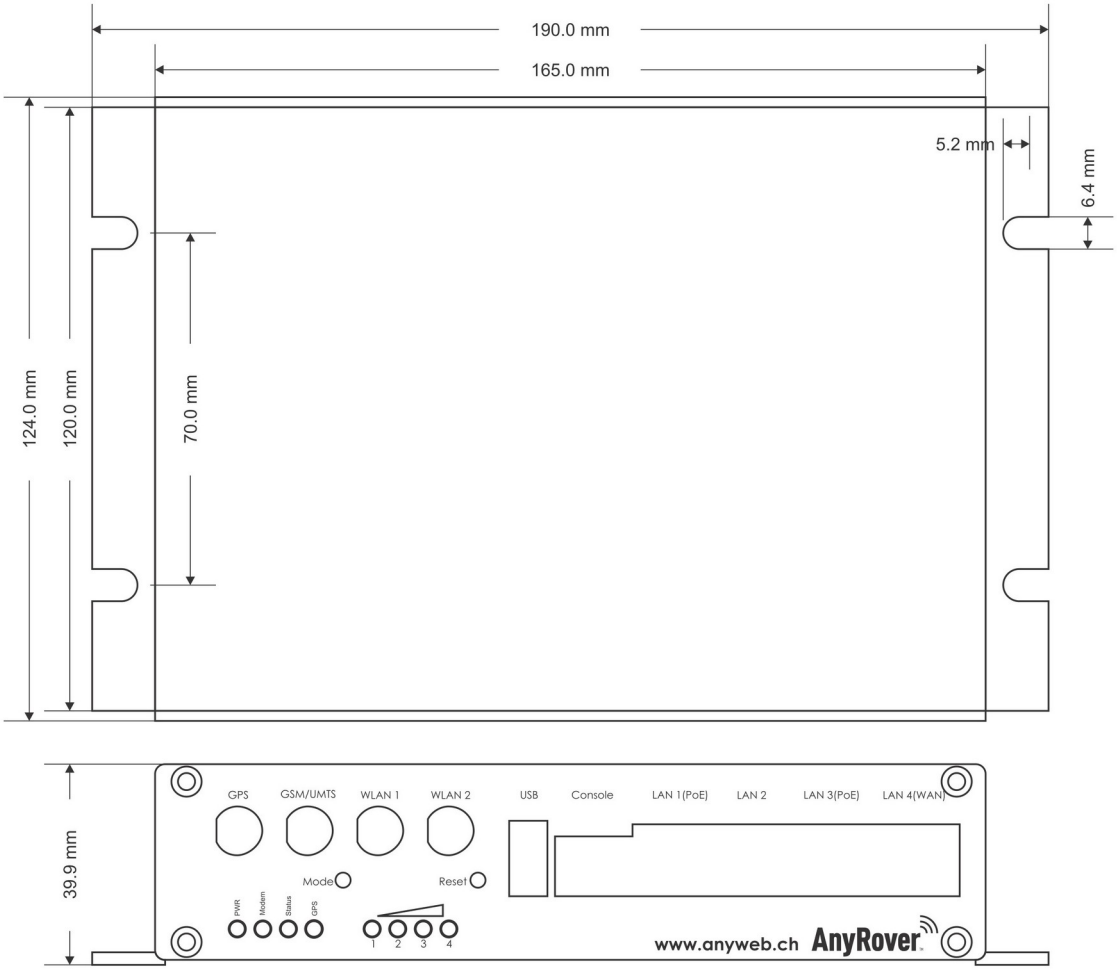


Figure 9: AnyControl 3D DR Status fine

Now the system is fully calibrated. The calibration data is stored on a volatile storage and will be erased after a power-off for more than one week.

Note: In vehicles with a lot of vibrations or if the AnyRover is not fixed properly, it is possible the the Status never changes to “fine”. Even though the functionality is provided and the data precision is sufficient.

3.5 Dimensional drawing



4 Configuration

The AnyRover can be configured on the command line.

Console access is possible over the local net and over the UMTS link. By default, ssh access is allowed on all interfaces.

A user config (password: cabtronix) is present for configuration changes. This user has a limited command set available.

Further changes in the system can be done as user root (password: root). It is possible to log in as root using ssh. To copy files back and forth, scp can be used.

Security warning: Change the passwords for both users. The command passwd on the command line achieves this.

Security warning: The root user is the standard linux administrator accout, without any securities implemented. It is possible to ruin the system if not working carefully. Users with little to no experience on Linux should not use the root user (except for changing password).

4.1 System configuration

The system configuration is stored in the file /etc/cablynx.conf. The file is a text file that consists of different sections. Every section contains the parameters for a certain service.

A section starts with the name in brackets ([section]), and ends at the start of the next section (or the end of the file).

Configuration entries are simple attribute value pairs (AVP) of the form

attribute = value

The equal sign (=) can be surrounded by spaces. Every AVP is on its own line. Empty lines are ignored.

The configuration file can contain comments. All text after a hash character (#) up to the end of line is considered as comment and is ignored. The only exception is the [chat_script] section, where the hash sign has to be in column one to start a comment (the command for dialling is 'atd*99#').

By default, the configuration file provides extensive comments for all attributes.

4.1.1 Changing the configuration on the command line

The user config can change the configuration by issuing the command

```
edit config
```

This command starts an editor with the configuration file. After saving and quitting the editor, the system is immediately updated. If you are logged in through the local network and change the IP address, you will lose connection.

When changing the UMTS configuration, the current 3G-connection is terminated and then created again. All connections on this link are terminated.

There are two editors on the system: vi and nano. With the command

```
edit
```

the user can find out which of them is set as default.

Hint: If you have never worked with vi before, use nano. To quit vi (without saving), the command :q! (colon – q – exclamation mark – enter) is used.

4.1.2 Changing the configuration with a memory stick

There is a possibility to change the configuration using a memory stick. A complete configuration file with the name cablynx.conf has to be placed on the memory stick in the root directory. With the following procedure, this file is copied over the current configuration.

When the AnyRover finds a file called /etc/reset during boot, it searches an attached memory stick for a configuration file. If it finds one, it replaces the current configuration with the new one and restarts all services. The memory stick is immediately unmounted again to be removed, and the file /etc/reset is deleted.

By default, there is a command in the [gpio] section called

```
button = 5, touch /etc/reset && sync && /sbin/reboot -d 4
```

If the reset button is held for more than 5 seconds, the AnyRover creates the file /etc/reset and reboots, thereby searching for a configuration on an attached memory stick.

4.1.3 Changing the configuration using SMS

It is possible to configure the AnyRover using SMS messages. By default, the access through SMS is disabled. There are three entries in the [sms] section that define access:

```
[sms]
```

```
console = no
console_key = abc123
eco_% = /etc/conf.s/eco.sh $@
```

There are two ways to disable access. Using console=no, it can be enabled with an SMS containing the console_key. If the console_key is set to '-', access cannot be enabled using SMS.

The SMS to enable access must contain the text „eco enable abc123“ with the correct console_key. The system will then modify the configuration file to read console=yes, and SMS access is open. Sending „eco disable“ changes the entry back to console=no. If the console_key is set to '-' before disabling access, it cannot be enabled any more using SMS (suicide method).

The SMS console understands these commands. The SMS must start with the command and not contain capital letters.

Command	Arguments	Effect
eco enable	Password	Enable the SMS console
eco disable	-	Disable the SMS console
eco conf	Section[:name] attr[+|-]=val	Change configuration
eco list	Section1 [section2]	Returns the requested section with an SMS to the sender (without comments)
eco reload		Restarts all services
eco reset		Resets config to factory default
eco templ	Name	Changes configuration to the named one
eco save	Name	Saves current config as name

The command eco conf has this syntax:

```
conf section[:name] attribute[+|-]=value
```

One SMS can contain several AVP. The first AVP must be preceded by a section name, further AVP for the same section can be added, separated by space. It is also possible to change to a different section by inserting a new section name. All words without an equal sign (=) are considered to be section names, all words with an equal sign to be AVP.

An AVP must not contain spaces. In the value part, all underscores (_) are changed into spaces (but not in the attribute part).

The assignment operator defines the action to be taken with the AVP. If the operator is just an equal sign (=), the existing AVP is replaced with the new one. If there is no such AVP, the entry is discarded. To add new AVPs, use the operator '+=', to remove entries, the operator '-=' is used.

When changing an attribute, the first match in the configuration is taken. If there are several AVP with the same attribute (e.g. in the firewall section the attribute accept), to replace another than the first one, it has to be removed using the '-=' operator, and then inserted again using the '+=' operator. Both these commands can be sent in the same SMS.

Newly inserted AVP always appear at the beginning of a section, or after a name attribute if there is one. If several AVP have to be inserted where the order is important (e.g. for the firewall rules), the last one has to be inserted first.

To delete an AVP, both the attribute and the value must match exactly.

Example: This SMS changes the IP address to 192.168.1.3, inserts a new firewall rule (accept = eth0,tcp,80) and deletes a rule accept=,udp,67 if it exists.

```
config system ipaddr=192.168.1.3 firewall accept+=eth0,tcp,80 accept-=,udp,67
```

When changing the configuration with SMS, the system is not updated automatically. To achieve this, use the command eco reload.

Security warning: The AnyRover can be completely reconfigured with this method. Be careful with this feature, and disable it if its not needed.

4.1.4 Querying the configuration using SMS

The command eco list followed by the list of requested sections returns the current configuration via SMS. Note that an SMS can only be 160 bytes long. If the command produces more output, only the first 160 bytes are returned, the rest is discarded.

When sending a command in the form

```
eco list firewall
```

to the AnyRover, the corresponding section is returned without any comments. When several sections are specified, the AnyRover returns them all. When no section is given, the complete configuration is sent.

4.1.5 Reset the configuration

The command eco reset resets the configuration to factory defaults.

The directory /etc/conf.d/ can contain different configuration templates which can be loaded using the SMS command

```
eco templ NAME
```

Note that the current configuration is overwritten.

4.1.6 Saving configuration templates

To save the current configuration as a template in the directory /etc/conf.d/ this command is used

```
eco save NAME
```

4.2 Variables

Variables can be used in the config file to collect values appearing multiple times (e.g. IP addresses) in one single definition.

4.2.1 Definitions

Variables are defined in their own section [variables], which lies at the head of the config file. The config file is parsed line by line, so variables can be used in all lines following the definition, but not on lines preceding the definition. Variable definitions can themselves use all previously defined variables.

Variables are defined using simple name=value entries. To use a variable, place the name in double curly braces: {{variable}}.

Example:

```
[variables]
ipbase = 192.168.1
gateway = 1
client = 2
[system]
ipaddr = {{ipbase}}.{{client}}
gateway = {{ipbase}}.{{gateway}}
```

Variables can be nested. Example:

```
[variables]
num = 1
config1 = file_a.conf
config2 = file_b.conf
file = {{config{{num}}}}
# file has the value file_a.conf
```

4.2.2 Runtime information

To place runtime information in variables, use the syntax {{run:VAR}} in the [variables] section. This command is replaced by the first line of the standard output of the command described by VAR. Such {{run:VAR}} elements may not be nested, although a {{run:VAR}} element may contain nested {{variables}}.

Note: do not place long running commands in variables, as the config file is parsed quite

often, and every command introduces latency.

Example:

```
[variables]
showversion = show version
sysdescr = AnyRover v3, {{run:showversion}}
[snmp]
sysdescr = {{sysdescr}}
```

4.3 Sections

All sections in the configuration file are described in this section.

The order of the sections in the configuration file is irrelevant, with these exceptions:

- [chat_script] must appear after [ppp]
- [certificate] must appear after [ipsec] or [openvpn], depending on which section references the certificates

Section	Description
system	Defines the IP address and netmask of the local network, the hostname of the system and static routes. Proxy ARP can also be enabled here.
switch	Defines the configuration of the switch (enable, VLAN)
time	Information on system time and NTP server
watchdog	Configuration of the watchdog
crontab	Configuration of crontabs. They are used to start programs at certain times.
gpio	Defines actions to be taken when events occur on the GPIO pins. This includes the reset and mode buttons on the front panel.
gps	Configuration of the GPS daemon. The daemon can send GPS data (NMEA strings) over TCP and UDP to other hosts (actively and passively).
sms	Configuration of SMS access. Defines what the AnyRover does with inbount SMS.
modem	The PIN code for the SIM card is stored in this section.
usb	Configures the USB port. Enables supply voltage on the port and defines actions for memory sticks.
dhcp	Configuration for the DHCP server.
ftp	Configuration for the FTP server.
tftp	Configuration for the TFTP server.
firewall	Firewall configuration.
dyndns	Configuration of DynDNS hostnames.
ppp	Configuration of the UMTS connection, including login and password for access.
chat_script	The chat script prepares the modem and dials the provider.
ipsec	Configure IPsec connections
certificate	This section stores certificates for IPsec connections.
openvpn	Configuration for OpenVPN server and client.

certificate	This section stores certificates for OpenVPN connections.
tunnel	This section defines IP-in-IP and GRE tunnels.
bridge	This section defines ethernet bridges.
banner	Message of the day. This message is shown on login.
daemons	Defines user applications that are started automatically.
script	This section can contain arbitrary scripts
webserver	Configuration of the web server.
wlan	Configuration for a WLAN card, if present.
authentication	Defines EAP or RADIUS server, e.g. for a WLAN AP.
certificate	This section stores the certificates for the authentication service.
ospf	Open Shortes Path First (OSPF) routing protocol.
snmp	Simple Network Management Protocol (SNMP).
serports	Configuration for the serial ports
openconnect	Configuration for Cisco AnyConnect VPN
mobileip	Configuration for MobileIP VPN.
scep	Configuration for SCEP (Simple Certificate Enrollment Protocol) certificate management.
pelix	Configuration of position data transmission to Pelix server.
login	Configuration of Login with Radius or Tacacs-Server
8021x	Configuration of IEEE 802.1X Port Security
tpm	Configuration of the Trusted Platform Module
power	Configuration of system power management
Bondix	Configuration of the Bondix-Software

4.3.1 [system]

Attribute	Value (Default)	Description
ipaddr	192.168.1.3/24	IP address and netmask or prefix of the local interface. The address can be given as 192.168.1.3/24 or 192.168.1.3 255.255.255.0. Using the parameter mtu:1492, the MTU of the interface can be set. To dynamically configure the interface, dhcp can be configured. If the value is “dhcp default”, the dhcp client will also set the default route. More possible parameters after dhcp: metric:M sets the metric of the route (default: 0) timeout:T sets the timeout to T seconds (default: 30) dns: Queries the DHCP server for DNS server addresses, and replaces current DNS configuration hostname: queries the DHCP server for a hostname,and replaces the hostname if it is localhost. nolinklocal: do not use link local addresses (169.254.X.Y) noarp: No ARP request to check whether address is still free. vendor:V set Vendor Class identifier string to V. clientid:I set Client ID string to I require:A,B Attributes A,B[,…] must be present in the offer, otherwise the offer is rejected. Examples: bootfile_name,

		tftp_server_name, ntp_servers, ... 8021x: IEEE 802.1X Port Security is activated on this interface. Details are configured in a separate [8021x] section.
loopback		Address that is assigned to the loopback interface. This attribute can be present multiple times.
gateway		IP address of the default gateway. Dynamically configured interfaces (dhcp) can set the default gateway, so this option is only used if the default gateway is on a statically configured interface.
policy		Rule for policy based routing. policy = SELECTOR ACTION where SELECTOR is one of from PREFIX, to PREFIX, tos TOS, dev DEVICE and ACTION can be table NUM, prohibit, reject, unreachable Example: policy = from 1.2.3.4 table 200
static_route		Configure static routes. Format: [target][/prefix] [netmask] gateway [metric:M] [table:T] [src:S] If both prefix and netmask are omitted, the default class-based prefix is chosen; if both are present, the prefix is ignored. If no target is given, the default route is set. gateway can either be an IP address or the name of an interface. A gateway IP address must already be reachable with the existing routing table, an interface must exist. table:T assings the route to the routing table T as created with the policy option above. With src:S the source address for this route can be set.
proxy_arp		Space separated list of interfaces that have proxy ARP enabled.
hostname	cablynx	Host name of the system. The hostname can be set by a DHCP client if it is set to localhost here.
nameserver		Defines a name server for the system. This parameter can appear multiple times to define up to 3 name servers. Additional entries are ignored. The first two entries are also used by IPsec to hand out to clients doing a mode config request (e.g. Cisco VPN Client).
domain		When doing DNS name lookups, if the name is not found, this suffix is appended and name is tried again.
winsserver		Up to two WINS server can be specified that are handed out by IPsec to mode config clients. These servers can only be set globally for all IPsec connections, not per connection.
log_server	192.168.3.2	When given, all log messages are sent to this log server
log_level		All messges with a log level less than this value are written to the log file /var/log/messages. By default, everything is written to the log file.
log_file	/var/log/messages	Name of the log file. The default log file lies on a RAM disk and is lost upon reboot. This parameter allow s to specify a different log file. If no path is given, the file is placed in /var/log. If the given path does not exist, it is created.
log_rotate_size	200	The log file is rotated automatically whenever it reaches a certain size. This parameter defines that size in KB.
log_rotate_files	1	When the log file is rotated, older files are deleted. This number defines how many rotated files to keep (max 99).
tftp_server	192.168.3.2	Default entry for the command *update system*

Partition	Define more partitions to be mounted upon start. Syntax: partition = device, filesystem, mountpoint [, option [,option]] The mount point will be created if not already present. Options as known from /etc/fstab. Option noatime is set by default. Possible options: rw (default), ro, [no]exec, … Example: partition = /dev/mtdblock4, jffs2, /media/log, noexec

4.3.2 [switch]

Attribute	Value (Default)	Description
start	yes	If set to yes, the switch is switched on. Otherwise, access over the network is not possible. During boot, the switch is only enabled after the firewall has been set up.
poe1	no	If set, the PoE PSE module on ethernet port 1 is switched on. Has no effect on a PD module.
poe2	no	If set, the PoE PSE module on ethernet port 3 is switched on. Has no effect on a PD module.
ports	4,4,4,4	Switch port configuration. Every port is configured by one number, separated by commas. Omitted values are assumed to be 4. The values mean: 0: 10 Mbit, half duplex 1: 10 Mbit, full duplex 2: 100 Mbit, half duplex 3: 100 Mbit, full duplex 4: auto negotiation
start_vlan	no	VLANs are only activated if enabled here. This causes the ip address from the [system] section to be assigned to VLAN 0. This VLAN 0 contains all ports that are not explicitely assigned to another VLAN.
port_disable		List of switch ports that are to be disabled (comma separated, ports number 1 to 4). This only works if start_vlan is set to yes. Example: port_disable = 3, 4
vlanX	-	This parameter defines one VLAN by listing the numbers of the ports that belong to this VLAN separated by commas. X can take a value from 1 to 4. vlan1 = 1,2
ipaddrX	-	The IP address and netmask or prefix of the interface in VLAN X on the AnyRover is defined here. ipaddr1 = 192.168.1.1/24 ipaddr2 = 192.168.2.1 255.255.255.248 Further values are identical to the parameter ipaddr in the system section. Interfaces with X=1..4 can be used for internal vlans, interfaces with X>4 can only communicate through an external trunk. At most 16 interfaces can be configured.
trunk		Using this parameter, external VLAN trunks can be configured on the switch ports. The parameter expects number of the switch port that is to be configured as trunk. This parameter can

		appear multiple times. Example: trunk = 3 If the trunk port is included in a vlanX parameter, that port can directly communicate through the trunk, otherwise the packets have to be routed by the AnyRover.
dhcp_rebind	no	If set to yes, DHCP clients will execute a rebind if a cable is plugged on the corresponding switch port.
reset	50	Internal signal. Number of the switch reset pin.
ps1	123	Internal signal. Number of the switch config enable pin.
port_poe1	53	Internal signal. Number of the PoE module 1 power pin.
port_poe2	54	Internal signal. Number of the PoE module 2 power pin.

4.3.3 [time]

Attribute	Value (Default)	Description
timezone	UTC	Timezone information The value is one of the entries in /etc/timezones (first column). Some possible values are: CET, GMT (incl. Daylight saving time), UTC (no daylight saving time), EET, EST, ... Alternatively, specify exact location, e.g. Europe/Zurich, America/Vancouver, Pacific/Auckland, ...
start	yes	If set to yes, a NTP server is run on the system. Don't forget to enable port 123 UDP in the firewall.
time_source	gps	gps: system time is set after the GPS receiver ntp: system time is set after a NTP server none: system time is not set
ntp_server	pool.ntp.org	Name or address of NTP server. Only used if time_source=ntp
ntp_flags		List of flags to restrict the service of the ntp server. The flags are entered into ntp.conf in a line like "restrict default <flags>". Possible flags: kod, limited, lowpriotrap, nomodify, noquery, nopeer, noserve, notrap, notrust, ntpport, version The list can be comma or space separated. Documentation of the flags can be found in the ntp.conf man page: http://www.google.com/?#q=man+ntp.conf Recommended value: ntp_flags = kod nomodify notrap nopeer noquery Without flags, the ntp service can be used for Denial of Service attacks on other hosts: sending an NTP-query to the server generates an answer that is much larger; couple that with a faked source IP address, and the answer will be sent to the target host.
localaccess		If set to yes, local processes have access to NTP status information. Needed to make show ntp work correctly.
ntp_option		Using this attribute, additional configuration options for ntpd can be specified.
jump_clock		NTP daemon normally adjusts the clock one after startup if the difference is less than 1000s. Otherwise, it exits with an error

		message. If this parameter is set to yes, it sets the clock on startup no matter the difference, and continues to keep the time current.
modemclock	Yes	check system clock with modem time. When there is a 3G connection, the system clock is checked automatically from the modem time every time, the connections is established.
syncled		If set to yes, time sync will be shown on external LEDs.

4.3.4 [watchdog]

Attribute	Value (Default)	Description
start	yes	The watchdog resets the sysetm if the feed line has not changed in about 1 minute. This attribute starts the watchdog. The system automatically feeds the watchdog when enabled.
interval	15	Interval in seconds for changing the watchdog line. The watchdog triggers after about 45-75 seconds. Keep below 30.
gpio_on	122	Internal signal. Number of watchdog enable pin.
gpio_feed	124	Internal signal. Number of watchdog feed pin.
cmd_on	1	Internal signal.

4.3.5 [crontab]

Attribute	Value (Default)	Description
start	no	If set to yes, cron daemon is started and the following entries entered into the crontab. The cron daemon can be started by other services, even if this value is set to no. But then, the following entries are not inserted into the crontab.
loglevel		Define the amount of logging of the cron daemon. Possible values (lower values include log messages of higher values): 8: Write a log message for every command executed 9: Only warnings and errors Default-value: 8
entry		Crontab entry. This line is copied directly to the crontab file, so the crontab syntax applies.

4.3.6 [gpio]

This section defines different actions to be taken when events occur on the GPIO lines. An action definition has the form

```
time, action
```

If time is present, the action is delayed for time seconds. Precision is only in the range of one second.

The action can be any command or program including arguments (the whole action is passed to the command „sh -c“).

There are some predefined action that can also be used in the [sms] section.

Action	Description
OFF	The system is shut down. This only works if the system is supplied through the GPIO connector, and the ignition line is low. The system starts again when the ignition line goes high.
CANCEL	Cancels a running OFF action. The use of these two is basically limited to ignition = (CANCEL), (60, OFF)
RESET	Resets the configuration to factory defaults. The current configuration is not saved.
REBOOT	Restart the system.
HALT	Shut down. If not switched of afterwards, will eventually reboot because of watchdog.
MOUNT	Mount all USB storage devices.
UMOUNT	Unmount all USB storage devices.
OUT_ON	Switch on GPO pin (set to high).
OUT_OFF	Switch off GPO pin (set to low).
SMS_STATUS	Send information about the GPI pins as SMS back to the sender. Only works in the [sms] section.

The attributes in the [gpio] section are

Attribute	Value (Default)	Description
button	2, RESET 5, UMOUNT 10, REBOOT	Action to be taken if the reset button has been pressed for more than the indicated time (in seconds). The number of actions is not limited, but they all must have a different time value.
mode		Action for the mode button.
ignition	(CANCEL), (60, OFF)	Action to be taken when the ignition line changes. The first action is taken on the positive edge, the second on the negative.
ign_boot	0	Defines what is assumed about the ignition signal upon boot. Depending on this value and the current state of the ignition signal, an edge is immediately detected. Possible values: 0 (or unset): read current value upon start 1: assume that ignition was off during boot 2: assume that ignition was on during boot
inputX	(),()	Action to be taken when the inputX line changes. The first action is taken on the positive edge, the second on the negative. The AnyRover has 3 inputs.
hysteresis hysteresisX		Hysteresis for input signals. If not defined, 0 is assumed. This parameter must appear after the respective inputX above. Syntax: hysteresis[X] = positive[, negative] X = number of input (1-3), omit for ignition. Positive and negative denote the number of previous samples that must have the same value for the edge to be detected

		and the configured action to be executed. One sample is taken every 250ms. If negative is not given, the same value as positive is assumed.
gpio_ign	46	Internal signal.
gpio_in1	30	Internal signal.
gpio_in2	29	Internal signal.
gpio_in3	28	Internal signal.
gpio_off	126	Internal signal.
gpio_but	39	Internal signal.
gpio_mode	41	Internal signal.
gpio_status	87	Internal signal.

4.3.7 [gps]

TCP and UDP connections to send GPS data (NMEA strings) through are configured in this section. There can be any number of connections, the GPS data is sent to all targets.

4.3.7.1 Targets

Active connections are defined with tcp_target and udp_target. The AnyRover tries to set up the connection (which always succeeds for a UDP connection if a route to the target is available). The line looks like this:

```
ret,target[:port][,[source[:port]|interface[:port][,id:X]]
```

Field	Description
ret	Defines the state of the reverse connection. ret = 0: data in reverse direction is silently dropped. ret = 1: it is possible to send data in reverse direction to the GPS reciever (e.g. with the u-center software from u-blox: www.u-blox.ch) ret = 2: commands on the reverse direction in the form CBCTL:{COMMAND} are interpreted, where command is a valid command for cablynxctrl (e.g. CBCTL:ekfreset). Output from the commands is sent over the link again as GPTXT string with the tag CBCTL. ret = 3: ret = 1 and ret = 2
target	IP addres or host name of the target. When using host names, make sure they can be properly resolved (e.g. via DNS).
port	Port number of the target. If no port is given port 13179 is used.
source	Source IP address for the traffic. The AnyRover must have an interface with this IP address. If not given, the system uses the address of the interface that the packets leave the system. Can be used if the packets have to be sent through an IPsec tunnel.
port	Source port for the traffic. Can be interesting if there are firewalls between the AnyRover and the target.
interface	Network interface that will be used as source for the packets. Possible interfaces are ppp (the modem) and eth0 (ethernet). Can be used if the packets have to be sent through an IPsec tunnel.
id	References a filter rule with name X (see below). If no filter rule is defined, all messages are passed through this connection.

For passive connections, the attributes tcp_server and udp_server are used. In this case, the AnyRover waits for incoming connections and starts sending data as soon as the connection is established. The fields are the same as above. Currently, udp_server doesn't work yet. Syntax:

```
ret[,source[:port]|interface[:port][,id:X]]
```

serial_target defines a serial interface that is used to send data. Syntax:

```
ret,serport[,baudrate[,id:X]]
```

Field	Description
ret	If ret = 1, then data can send in the opposite direction to the GPS receiver (i.e. with the u-center software by u-blox: www.u-blox.ch). If ret = 0, return traffic is silently dropped.
serport	Name of the serial port, e.g. /dev/ttyS1 for the first serial port, /dev/usbser0 for a USB-serial converter.
baudrate	Baudrate of the transmission. Possible values are 2400, 4800, 9600, 19200, 38400, 57600, 115200
id	References a filter rule with name X (see below). If no filter rule is defined, all messages are passed through this connection.

With file_target, data can be written to a file. The file is automatically rotated an gzipped, if a defined size is reached. Old zip files are deleted. Syntax:

```
ret,file[,maxsize[,rotate[,hook[,id:X]]]]
```

Field	Description
ret	For file_target, the parameter ret is ignored.
file	Name of the file to write data to. When rotating, the suffix ".XX" is appended, and the file gzipped, which results in files named [file].01.gz. If the files are written to the root partition, the size is limited to 10MB and one old file (rotate = 1), to prevent the partition from filling up.
maxsize	The file is rotated if it becomes larger than maxsize (in bytes). The size is periodically checked, so the exact size when rotation occurs cannot be predicted. Possible values: 1 – 2'147'483'647 (=2GB) Default value: 4MB
rotate	Defines how many old files are kept. Rotated files are saved as [file].01.gz, [file].02.gz, [file].03.gz etc. where [file].01.gz is the youngest file. The oldest file is deleted when maximal number is reached. Possible values: 0 – 99 Default value: 5
hook	Define a program that is executed whenever the file is rotated. In this case, the file is not gzipped. The program gets the name of the rotated file as parameter.
id	References a filter rule (see below) named X. If no rule is given, all messages are sent out on this connection.

4.3.7.2 Filter

Generally, all messages obtained from the GPS reciever or defined through gptxt attributes are sent to all targets. To reduce the number of messages, filters can be used that are referenced in target definitions with the attribute id:X, where X is the name of the filters consisting of letters and numbers. The filter names anytracker and at_script are used internally. Comments are not allowed on lines with target definitions.

If no filters are defined for a target, all messages are sent. Filter rules are processed in order until one rule matches, after which processing stops.

Every filter can switch between different profiles, such that filter rules can be changed based on external events. Profile switching is done through the cablynxctrl utility with the command profile. By default, profile 0 is active.

Syntax rules for filters:

```
filter = ID[:profile]; PATTERN = time[,dist][;PATTERN = time[,dist]]
```

Note: semicolon is used as field separater in filter rules, not comma.

Field	Description
ID	Name of the filter rule, referenced from the target definition.
profile	Profile number. Arbitrary number >= 0.
pattern	Text that is compared to the beginning of the message. A question mark (?) can be used to represent any character.
time	Time interval in seconds, after which the next message is sent. Value 0 means that no messages are sent.
dist	Distance interval in meters. The device must have moved at least this distance until the next message is generated.

Rules for one ID can be listed on one filter line or multiple lines, there is no difference.

A filter rule never generates new messages. Every message is passed through the filter rules immediately after creation, and then passed or discarded. If no messages are generated, no filter is activated. This means that if a filter is defined to send $GPGGA messages every 5 seconds, but the GPS receiver produces $GPGGA messages only every 10 seconds, then a message is sent only every 10 seconds.

UBX messages (for information to the UBX protocol see relevant information on ublox homepage) can also be filtered, with the pattern

```
UBX[-class[-id]]
```

The list of UBX class and id names can be found on the AnyRover in the file /etc/ubx.txt.

Examples:

Send only GPGGA and GPRMC messages:

```
filter = ruleA; $GPGGA = 1; $GPRMC = 1; $GP = 0; UBX = 0
```

Send GPGGA messages every second, but if ignition is off, only every 10 seconds.

```
filter = ruleB:0; $GPGGA = 1; $GP = 0; UBX = 0
filter = ruleB:1; $GPGGA = 10; $GP = 0; UBX = 0
[gpio]
ignition = (echo profile ruleB 0 | cablynxctrl), (echo profile ruleB 1 | cablynxctrl)
```

4.3.7.3 Attributes

Attributes for the [gps] section are

Attribute	Value (Default)	Description
start	yes	If set to yes, the GPS receiver is switched on
run_gpsd	no	If set to yes, the gpsd service is started
gpsd_port	2947	TCP listen port for gpsd. Must be enabled in the [firewall] section.
gpsd_debug	0	Log level of gpsd. The higher the number the more messages gpsd sends to the system log. Level 2 already logs all NMEA messages.
assist_now		AssistNow is a service which provides current satellite data online that allow the receiver to acquire a position only seconds after power on. The data are valid only for a short time (a couple of days), so make sure that current data is available. If a file name is given here, the file is loaded into the GPS receiver upon system boot. AssistNow data can also be loaded into the GPS reciever later using the program cablynxctrl.
udp_target		Target for active UDP connection. See above.
tcp_target		Target for active TCP connection. See above.
tcp_server		Configuration for passive TCP server. See above.
udp_server		Configuration for passive UDP server. See above.
serial_target		Configuration for a serial interface
file_target		Writes data into a file.
filter		With this filter, the number of messages sent to each target (or server) can be limited. No new messages are created, so if the filter says to send every 5 seconds, but the source delivers messages only every 15 seconds, only these messages from the source will be sent. Syntax: filter = ID[:profile]; PATTERN=time[,dist][;PATTERN=time[,dist]] Multiple lines for the same ID are allowed. There is no difference between specifying all rules on the same line and on different lines.

	The ID is references from the (tcp | udp)_(target | server) attributes. Do not use the IDs anytracker and at_script, as they are used internally for atmsg_*. profile is a number (0, 1, 2, ...), to be able to define different rules based on external variables. On start, profile 0 is applied. Changing the profile is done through the cablynxctrl utility. The message to be sent is matched against the PATTERN, and if it matches, the messages is only sent if the time or dist constraints match. time (in seconds) defines the minimum interval between to messages. dist (in meters) defines the minimum distance the GPS receiver must have been moved between two messages. A value of 0 for both intervals means do not send any messages. Example: filter = rule1:0; $GPGGA = 5; $GPRMC = 5; $GP = 0 Send GPGGA and GPRMC messages every 5 seconds, but no other GP messages filter = rule1:1; $GPGGA = 0, 200; $GPRMC = 0, 200; $GP = 0 Profile 1 of the same rule. Send GPGGA and GPRMC after moving 200m. If the GPS receiver does not move, no messages are sent. UBX messages can also be filtered: filter = ubx; UBX-NAV-EKF=10; UBX-CFG = 30; UBX = 0 The file /etc/ubx.txt contains all available UBX class and id names.
tcp_init_str	This string is sent without modifications as the first message on ever TCP connection established.
atmsg_en	AnyRover can create messages in AnyTracker format: {2280123456789098 | POSITION | 14.07.2013 | 10.09:32 | 04724.1234 | N | 00832.9876 | E | 415.0 | 3.0 | 246 | 5 | 1.67 | 99 | 0:0:30 | 300}\r\n {ID | TYP | DATE | TIME | LAT | NS | LONG | EW | ALT | SPEED | DIRECTION | NSAT | HDOP | BATT | TIME_INT | DIST_INT}\r\n Currently, only POSITION messages can be created, and battery leven (BATT) field always contains the value 99. This parameter defines, whether such messages are created. Using filters, these messages can be limited: filter = at; { = 0
atmsg_id	Value to place into the ID filed of the message. Use <IMSI> or <IMEI> to use IMSI or IMEI from modem 1, <IMSI1> or <IMEI1> to use these values from modem 2.
atmsg_buggytime	Early versions of AnyTracker contained a bug, where the time field was sent as hh.mm:ss instead of hh:mm:ss. If set to yes, the buggy format will be sent.
atmsg_interval	Define time and distance intervals to create AT messages. Syntax: atmsg_interval = time, dist [, profile]
atmsg_use_utc_time	Use UTC (yes) or local time (no) in AT messages time stamp.1
atmsg_script	Defines a script that is called according to atmsg_script_interval and whose output is sent as message. In the script, these environment variables can be used to access current GPS data: ID, IMEI, IMSI, TIME_UNIX, TIME, DATE, LATITUDE, NS, LONGITUDE, EW, ALTITUDE, SPEED, DIRECTION, NSAT, HDOP, ATMSG_INTERVAL, ATMSG_DISTANCE, GPGGA, GPRMC
atmsg_script_interva l	Define time and distance interval for atmsg_script. Additionally, a profile can be specified, and a pattern that matches messages generated by the script, needed for internal

		processing.
gptxt		Configuration of GPTXT messages. Syntax: gptxt = interval, action Every interval seconds the action is executed, and the output sent in a GPTXT message. Internal actions: GPI: send state of all the inputs: GPTXT,GPI,A,B,#1,V1,#2,V2,#3,V3*XX A: ignition, B: reset button #1: number of GPI pin, V1: value of GPI pin XX: checksum (xor of all bytes between $ and *). WLAN: List of currently visible WLAN access points. This only works if the WLAN card is configured as client and running. Format: $GPTXT,WLAN,<ssid>,<mac>,<channel_freq>,<signal>* DIP: position of the DIP switches 1-6. Format: $GPTXT,DIP,0,0,0,1,0,0*3f Alternatively, some executable (including path) can be specified, which is run and its standard output is sent in the GPTXT message. If the output contains newline characters or is longer than 70 bytes, the text will be split into multiple GPTXT messages.
gptxt_file		Write the last GPTXT messages to this file. The directory must exist. Recommended: /var/gps/gptxt.txt The first field of the GPTXT message is used as key, and for every key only one message is stored. E.g. $GPTXT,INFO,Hello world* Here, the key is INFO. The next message with GPTXT,INFO will overwrite this one.
gptxt_writeout		Write the file every n seconds. Set to 0 to disable this feature.
gptxt_clean		Remove GPTXT messages older than n seconds. They will be reinserted if they reappear.
directory	/var/gps	Directory for fifos where the NMEA strings are made available to local programs. The fifos should not be read using cat (will never terminate), but with 'head -n 1 fifo'.
gps_bypass	no	Enable or disable the gps bypass. The gps bypass can send data directly to the external serial port. This can be achieved as well by adding a serial target, but the bypass is faster. To enable the gps bypass the serial ports have to be enabled in the serports section.
device	/dev/ttyS2	Internal signal. Serial interface where the GPS receiver is connected to.
baudrate	9600	Internal signal. Baud rate of GPS receiver. Changing this parameter will NOT reconfigure the GPS receiver.
gps_reset	38	Internal signal. Number of GPS receiver reset pin.
gps_on	125	Internal signal. Number of GPS receiver power pin.

4.3.8 [sms]

Attribute	Value(Default)	Description
start	yes	If set to yes, the AnyRover will check for incoming SMS.
phone_number		List of phone numbers that are allowed to send SMS commands.

		If the list is empty, all SMS are accepted. Example: phone_number = +41790123456, +41760987654
interval	15	Interval in seconds for SMS checks.
console	no	Defines whether the command eco_% is enabled.
console_key	-	If the SMS console is disabled, it can be enabled with an SMS "eco enable KEY". If the key is set to "-", the console cannot be enabled via SMS.
send_answer_back		If set to yes, the first 160 bytes of the output of the command are sent back to the sender of the SMS.
send_answer_to		List of phone numbers where the output of the command (160 bytes) is sent to, independent of the value of send_answer_back.
catch_all		Define a command to be executed when a SMS message cannot be assigned to a user defined command. The program is passed the phone numberr and the text of an SMS in two environment variables $PHONE_NUMBER and $SMS_TEXT.
sender_as_text	no	If an sms is received from a defined sender like SWISSCOM, the sender id can be used as text or number. Default value is number.
anygator_tpm		If wget downloads in AnyGator scripts are supposed to use a TPM key, enter key ID here.

The rest of the entries are commands that can be sent via SMS. The attribute is the command (spaces in the SMS are replaced by underscores, the attribute must not contain spaces). The value contains a flag and the command that is executed (using „sh -c").

The flag defines whether the command must be protected with a hash value. If the flag is 0, the SMS doesn't need a hash, if the flag is 1, a valid hash must precede the command (separated by '-'), and if the flag is 2, a correct 3 way handshake has to take place prior to sending the command (not implemented yet).

Commands can contain parameters. The attribute must be marked with a trailing %; in the value, the strings %1, %2, … %9 are replaced with the respective parameters from the SMS. The string %@ designates all parameter values. If the command requires a '%' character, it must be entered as '%%'.

Some examples:

ping_router	0, ping -q -c 4 `route | awk '/^def/{print $2}'` | awk 'BEGIN{a=0}/^---/{a=1;next}a'	Sends 4 ping packets to the default gateway
ping_client	0, ping -q -c 4 `awk '/ List /{a=1;next}a{print $1;a=0}' < /etc/hosts` | awk '/^---/{a=1}a'	Sends 4 ping packets to the first DHCP client
position	1, head -n 1 /var/gpgga.fifo	Returns the current GPS position
config_%	1, cp /etc/cablynx_templates/%1 /etc/cablynx.conf	Copies /etc/cablynx_templates/XXX to /etc/cablynx.conf XXX is replaced with the first

		word in the SMS after “config”.
eco_%	0, /etc/scripts.d/eco.sh %@	Some predefined actions

The predefined commands from the [gpio] section can also be used here.

4.3.9 [modem]

Attribute	Value (Default)	Description
name	modem1	Identifies the modem if more than one are available
band	3	Defines the radio band for the modem: For 3G modems: 0 = Automatic 1 = UMTS 3G only 2 = GSM 2G only 3 = UMTS 3G preferred 4 = GSM 2G preferred In 2G mode, the modem cannot receive SMS under load For LTE modems: ## 0-2: identical to 3G modems ## 3, 4: Automatic ## 5: GSM and UMTS only ## 6: LTE only ## 7: GSM, UMTS, LTE ## 11, UMTS and LTE Only ## 12, GSM and LTE Only
disable_roaming	No	Disable roaming to foreign mobile networks. This function is activated when the parameter is set to yes.
sim_pin		PIN code for the SIM card.
imsi		IMSI checker: Rules are defined on whether to start ppp depending on the currently inserted SIM card, and start on which interface. To find out the IMSI of the currently inserted SIM card, issue this command: show id The rules are evaluated in the order they appear in the config file. The rules have the form: <IMSI>, X (-1 <= X < 2147483648). If the IMSI matches, ppp is started on interface pppX. If X is negative, ppp is not started. The IMSI “-” matches any SIM card; therefore it makes no sense to put further rules after one having “-”, they are never tested. Examples (with IMSI 228013520284438): To start ppp only for one particular SIM card: imsi = 228013520284438, 0 imsi = -, -1 To start ppp on ppp0 for one particular IMSI, and on ppp100 for all others: imsi = 228013520284438, 0 imsi = -, 100 To not start ppp for one particular IMSI, but for all others: imsi = 228013520284438, -1 imsi = -, 0
wait_for_sim		Whether to wait until SIM card is ready. Some SIM cards need

		some time after entering the PIN until they are ready. Sometimes, the connection is started too quickly which results in a failed connection attempt. Should always be set to yes.
gpio	47	Internal signal. Number of modem power pin.
disable	48	Internal signal.
cmd_on	0	Internal signal. Command to switch modem on.
slot	0	Slot where the modem sits. For AnyRover always 0.
get_modem_status	yes	Get modem status informations and store them to a file
status_interval	60	Define how often modem status informations get collected. This value must not be smaller than 20 seconds.
show_rx_led	yes	Show signal RX level on external LEDs. When set to no, this command will not touch the external LEDs. This may be needed if the LEDs are used to display something else.
log_value		Log bad modem connection if signal level is below this value.
ringtime	30	Time in seconds the phone is left ringing when initiating a phone call (using cablynxctrl).
calltime	10	Time in seconds an established call is kept open.
max_call_retry	10	Number of times the device tries to establish a phone call if something fails.
modem	/dev/clmodem	Internal signal.
hip	/dev/clhip	Internal signal.
ctrl	/dev/clctrl	Internal signal.
gps	/dev/clgps	Internal signal.

4.3.10 [usb]

Attribute	Value (Default)	Description
poweron	yes	If set to yes, the power line on the USB interface is switched on. The internal WLAN card also requires this parameter to be set.
usb1	yes	Switch on power on for external USB Port (connector for WLAN module). To enable the port, poweron has to be set to yes.
usb2	yes	Switch on power on for external USB Port (connector for WLAN module). To enable the port, poweron has to be set to yes.
usb4	yes	Switch on power on for external USB Port (connector on the outside of the device). To enable the port, poweron has to be set to yes.
switch_wlan	no	Using this parameter, the two WLAN modules wlan0 and wlan1 can be switched.
restart_wlan		If set to yes, the WLAN card is automatically started new if Vendor Request Failed messages appear.
start_sdcard		Switch on power of the SD-card.
automount	yes	If set to yes, USB memory sticks and SD-cards are mounted automatically when connected.
ignore_errors		If set to yes, VFAT filesystem errors on the SD-card are ignored. If set to no, the SD-card will be remounted read-only upon errors.

sdpart	Mount points for partitions on the SD card. This parameter can appear multiple times, once for each partition to mount. Partition numbers start at 1. This parameter is only active if automount = yes. Syntax: sdpart = PartNum, MountPoint Example: sdpart = 1, /media/sdcard1

4.3.11 [dhcp]

This section defines a DHCP server on one interface. If multiple servers on different interfaces are needed, this section can be used several times.

The parameters are separated into 4 categories. First is the general information, after that the bootp specific parameters follow (next_server, hostname, and boot_file). These values are placed inside the DHCP/bootp packet. The rest contains many DHCP settings, which are appended to the packet as options. Finally, there are entries for static leases.

Attribute	Value(Default)	Description
name	-	Name of the interface the DHCP server runs on. Possibilities: eth0, vlan1 .. vlan4, wlan0
start	yes	If set to yes, the DHCP server is started.
lease_file		Location of the lease file for this server. If not given, the lease file is /var/lib/misc/udhcp.leases.<IFACE>, which lies in a RAM disk and will be gone after the next reboot.
log		If set to syslog, the DHCP server will log its actions to syslog.
port	67	UDP port to listen for DHCP requests on.
dhcpd_start	192.168.1.11	Lowest address of the range the server hands out.
dhcpd_end	192.168.1.254	Highest address of the range the server hands out.
next_server		This IP address is placed in the next server field in the bootp header.
server_hostname		This name is announced as hostname of the server.
boot_file		Name of the file the bootp client is using as boot file.
prefix		Prefix for the dynamic range.
netmask		Netmask of the dynamic range. If both prefix and netmask are given, the value of netmask is used. If neither is given, the class based prefix of the dhcpd_start address is used.
router	192.168.1.3	Default router for the clients. This parameter can appear multiple times to enable sending multiple router addresses. If set to default, the IP address of the interface the server is running on is sent.
dns	192.168.1.3	Name server the DHCP server hands out to clients. This parameter can appear multiple times.
lease	864000 (10 days)	Lease time in seconds
timezone		Offset in seconds of local time to UTC. This can be used to define local time zone.
*logsrv *lprsrv *nissrv *ntpsrv		IP address for the named server. Entries marked with * can appear multiple times to announce multiple servers.

*wins swapsrv tftp	
hostname	Hostname sent to the clients.
bootsize	Size of the boot file, in 512 byte blocks.
domain	Domain the clients should use for DNS queries.
rootpath	Path to the root disk of the client.
ipttl	TTL the client should use.
mtu	MTU of the local network.
broadcast	Broadcast address for the local network.
nisdomain	NIS domain name.
requestip	IP address
dhcptype	Number
serverid	IP address that will be sent as server ID.
message	Text
vendorclass	Text
clientid	Text
bootfile	Name of the file the DHCP client should use as boot image.
wpad	Settings for MSIE Web Proxy Autodiscovery Protocol.
vendorspec	This can be an arbitrary hex string. The format is: vendorspec = 41:65:d:a:0
sipsrv	Type and URL of a SIP server. If given multiple times, all URLs must have the syme type. Syntax: type:URL Type = 0: URL = real URL; type = 1: URL = IP address
captiveport114	Captive Portal option according to RFC 8910 (option 114)
captiveport160	Captive Portal option according to RFC 7710 (option 160)
static_lease	Defines a static lease for a specific MAC address. The MAC address and IP address are separated by space. This parameter can appear multiple times. Example: static_lease = 00:11:22:33:44:55 192.168.1.11

4.3.12 [dhcprelay]

This section defines DHCP relay services.

Attribute	Value (Default)	Description
start	no	Defines if the service is started.
client		List of interfaces (comma separated)to listen for DHCP requests on. If empty, listen on all interfaces. If an interface is prepended with a '!', the interface is excluded from the list. The entry "client = !vlan1" means to listen on all interfaces except vlan1.
server		List of servers that DHCP requests are forwarded to. Can be IP addresses or interfaces. If set to an IP address, the packets are unicast to the address. If set to an interface, the packets are broadcast on that interface. The gw-addr field in the DHCP header is filled with the interface

	the packet was received on.

4.3.13 [ftp]

Attribute	Value (Default)	Description
start	no	If set to yes, the FTP server is started.
basic	yes	If set to yes, some basic configuration options are used.
anonymous	yes	If set to yes, anonymous login is allowed.
anonymous_dir	/media/sda1	Directory for anonymous users. They cannot leave this directory.
anonymous_write	no	If set to yes, anonymous users can upload files.
anonymous_delete	no	If set to yes, anonymous users can delete files.
option		These options are written directly to the vsftpd.conf file. Hint: vsftpd does not allow spaces in the options.

4.3.14 [tftp]

Attribute	Value (Default)	Description
start	no	If set to yes, the TFTP server is started.
upload	no	If set to yes, uploads to the AnyRover are allowed.
rootdir	/tftp	Directory for the TFTP server. Only files in this directory can be loaded over TFTP, and uploads are stored here.
port	69	UDP port where the TFTP server listens on. Don't forget to open this port on the firewall.

4.3.15 [firewall]

Define firewall rules. The rules are applied and later checked in the order they are placed in the config file.

Rules are divided into two categories: firewall rules are rules that only inspect a packet and then decide what to do. Mangle rules are the rules that modify packets, e.g. nat or port forwarding.

Attribute	Value (Default)	Description
filter_bridged	yes	If set to yes, packets on the bridge are seen by the firewall. This can only be set globally, not per bridge.
filter_vlan	yes	If set to yes, VLAN tagged packets on the bridge are seen by the firewall. This can only be set globally, not per bridge.
forward	yes	The kernel forwards packets from one interface to another. If set to no, local clients have no access to the internet.
nflog_start		NFLOG is a logging method where logged packets can be received and evaluated by user space programs. If this

		parameter is set to yes, the system will evaluate NFLOG packets.
nflog_script	/etc/scripts.d/nflog.sh	If a packet is logged through NFLOG, the system will execute this script and passes all relevant information through environment variables (NFLOG_*). The default script will execute all scripts in /etc/scripts.d/nflog/ in alphabetical order.
nflog_group	7	The NFLOG target knows different groups. This parameter defines which group is used in the system. Possible values: 1-32
nflog_payload_length	64	Number of bytes to copy for UDP packets to the variable NFLOG_PAYLOAD. If non-printable characters are encountered, copying stops immediately.
start_firewall		If set to yes, firewall rules are applied.
basic	yes	If set to yes, some basic rules are implemented: – Block all connections to AnyRover and through AnyRover – Allow ICMP echo requests (ping) – Allow established connections – Allow related connections (e.g. FTP data, ICMP errors)
new_chain		Syntax: new_chain = NAME[,CHAIN[:POS]] Allows the creation of a new filter chain, the value is the name of the chain. The name must not contain spaces or underscores. To create multiple new chains, use the attribute repeatedly. If CHAIN is given, a jump rule to NAME is inserted into CHAIN at position POS.
accept accept_fw accept_out accept_*chain* drop drop_fw drop_out drop_*chain* reject reject_fw reject_out reject_*chain* return return_fw return_out return_*chain* log log_fw log_out log_*chain* nflog nflog_fw nflog_out nflog_*chain* chain		Definition of firewall rules. Syntax: TARGET = [SRC][,[!]proto[,DST]][,R:RATE][,L:prefix][,I:ICMP][,MAC:[!]ADDR] where: TARGET = RULE[_CHAIN] RULE = (accept | drop | reject | return | log | nflog | custom chain name) _CHAIN = (| _in | _fw | _out | _NAME) SRC,DST = [[!]if] [ipsec] [[!]net][:[!]ports] RATE = [rate][:burst] ADDR = {MAC address} ICMP = (icmp-port-unreachable | icmp-host-unreachable | icmp-port-unreachable | icmp-proto-unreachable | icmp-net-prohibited | icmp-host-prohibited | icmp-admin-prohibited) An exclamation mark inverts the matching, i.e. the rule then matches everything except the given value. if: input or output interface. For bridge interfaces, the physical interface can be specified: br0>vlan1. ipsec: the rule only matches if the unencrypted traffic enters (SRC) or leaves (DST) through an IPsec tunnel. The keyword ipsec can only be used either with SRC or with DST, but not both (e.g. for _fw rules). net: source or destination network port: source or destination port; only active if protocol is tcp or udp. If there are more than one port, they have to be seperated with a colon. proto: protocol (tcp, udp, esp, icmp) rate: Can be used to rate limit the rule. Possible values are e.g. 3/sec, 6/min, 13/hour, 2/day. This function is mainly used for the log target to prevent filling up the log file. This parameter is not suited for bandwidth control. burst: maximal initial number of hits (default: 5)

		ADDR: Filter based on source MAC address of packet. ICMP: when using the reject target, the sender is notified with this message. prefix: text to place in front of the packet information in the log file. The text must not contain , or ' characters. The text can be enclosed in quotes ("), but this is only needed if the text ends with white space. The difference between drop and reject is that drop silently discards the packet, while reject informs the sender. Use drop unless you know that you need reject. Return stops processing in the current chain and returns to the parent chain, or applies the chain policy in the root chains. If RULE is set to the name of a custom chain, upon matching of the rule parsing is continued in the named chain. The _in chain applies to packets addressed to the AnyRover, the _fw chain to packets being routed through the AnyRover, and the _out chain for packets generated on the AnyRover and leaving. _NAME can be the name of a custom filter chain as defined above. If no chain is given (e.g. accept = ...) the input chain is used. When a rule matches, processing stops, and the packet is treaded according to the rule. The exception are the (nf)log rules, which do not stop processing on a match.
rule		Place any additional iptables rules here. The value is directly passed to iptables.
start_mangle	yes	If set to yes, mangle rules are applied.
nat	ppp0	List of interfaces where NAT will be enabled.
new_natchain		Same as new_chain, but for the NAT tables.
portfw		Defines port forwarding rules. Syntax: [proto],target[:tport],dest[:dport][,source] proto: protocol. If omitted, all protocols match. target: interface name or address the packet is originally addressed to. Can be a network addres (e.g. 192.168.2.0/24). tport: port the packet is originally addressed to. dest: IP address the packet is resent to. dport: Port the packet is resent to. If omitted, the port value is not changed. source: the source address the packet has to come from. If omitted, any source address matches.
snat[_chain] dnat[_chain]		Source- and destination-NAT rules. Syntax (similar to firewall rules above): snat = [SRC],[proto],[DST],T:target dnat = [SRC],[proto],[DST],T:target target defines the source or destination address to be set. For snat, setting input interface is not possible, and output interface for dnat (but addresses are). Both IP address and port can be specified as ranges, e.g. 10.0.0.1-10.0.0.10:1000-1200. Destination NAT rules are applied on incoming packets before a routing decision is taken, source NAT rules on outgoing packets after a routing decision has been taken, but before the packet is checked for IPsec encryption. Destination NAT is the same as portfw above, but with different rule syntax. Use case: for syslog and NTP traffic, no source can be specified, the interface address on the direct path to the destination is taken. But to send this traffic over IPsec, another source address

	might be necessary. Having 10.11.12.13 the syslog server and 192.168.1.3 the internal IP address, the appropriate rule is: snat = ,udp,10.11.12.13:514,T:192.168.1.3 Further examples: dnat = ,tcp,192.168.1.0/24,T:10.1.2.3:8000-8020 snat = 10.11.12.0/24,tcp,192.168.1.24,T:10.1.2.1-10.1.2.5
tcpmss_*chain*	Used to modify TCP MSS (maximum segment size). The MSS is only transmitted in the first packet of a TCP connection (SYN bit set). Syntax: tcpmss_*chain* = [SRC],tcp,[DST],M:*{mss}* Usable for the value chain are in, out, fwd, PREROUTING, and POSTROUTING. SRC and DST are similar to the accept_* rules above. In PREROUTING, no output interface, and in POSTROUTING, no input interface must be used. Source and destination networks can be used though.

4.3.16 [dyndns]

Attribute	Value (Default)	Description
start	yes	If set to yes, the DynDNS daemon is started.
username	user	User name with the DynDNS provider.
password	pass	Password with the DynDNS provider.
hostname	myhost.dyndns.org	Host name of the DynDNS server where the update must be performed. This attribute can appear multiple times if the provider has several servers.
option	syslog	These options are placed directly into the config file of INADYN

4.3.17 [ppp]

This section defines a PPP connection on a 3G modem. It can appear multiple times for multiple connections (but then multiple modems are required).

To perform action on PPP state changes (connection up or down), scripts can be deposited in /etc/scripts.d/ppp-up-hooks and /etc/scripts.d/ppp-down-hooks. The scripts get some information through parameter.

The scripts can be defined through script sections.

Attribute	Value (Default)	Description
modem		This parameter which must be the first one in this section references a modem section and defines the modem to use to set up the ppp connection.
start	yes	If set to yes, the ppp daemon is started. Without ppp daemon, no UMTS connection can be established.
user		User name for login at the provider. If not used, comment this line out.

password		Password for login at the provider. If not used, comment this line out.
defaultroute	yes	If set to yes, the PPP connection will be set as default route.
defaultmetric		Defines the metric of the default route. PPP will not replace an existing default route with the same metric (default: 0).
usedns		Defines whether to use name servers as advertised by peer.
debug	no	If set to yes, pppd logs additional information to the system log.
basic	yes	If set to yes, use standard options for ppp daemon.
chat_verbose	yes	If set to yes, chat logs the execution state as well as all text sent and received during dialling. Only has an effect if basic=yes
chat_script	basic	Selection of the chat_script section. This value references the name attribute in the chat_script section.
restart	yes	If set to yes, the modem is reset when ppp connection goes down.
timeout	2	Time in seconds to leave the modem switched off.
hold_nocarrier		If set to yes, do not restart modem when connection fails with NO CARRIER. This leads to faster reconnect times after losing connection because of no reception in dead zones.
filter		Packets that match this filter will trigger dial-on-demand and reset the idle-counter. If not set, all packets match. Syntax is similar to tcpdump, see tcpdump man-page for further details. Expressions that are inappropriate for ppp links such as ether and arp are not permitted. Syntax: [([] [not] expr [)] [and|or] [[)] [not] expr [)]] if src and dst are omitted, both directions match [src|dst] host HOST [src|dst] net NET [mask MASK] [src|dst] port PORT [src|dst] portrange RANGE ip proto \\(icmp|ah|esp|tcp|udp) (inbound|outbound) expr RELOP expr RELOP is one of <, >, <=, >=, =, != expr can contain integers, +, -, *, /, <<, >>, &, | PROTO[expr[:size]] Example: filter = outbound and not ((tcp[13] & 4 != 0) or (icmp[0] = 3))
option	demand persist idle 300 holdoff 15	These options are written directly into the configuration file of the pppd. If the option demand is used without persist, then nopersist has to be given explicitely.

4.3.18 [chat_script]

Attribute	Value (Default)	Description
name	basic	Identifies the section. This name is referenced in the [ppp] section.
apn	gprs.swisscom.ch	APN for the UMTS access. This parameter is only relevant in the chat_script section named basic.

script	All script lines are copied to the chat script file. These lines are only relevant in chat_script sections other than basic.

4.3.19 [wan]

Currently not used.

4.3.20 [ipsec]

This section defines an IPsec tunnel. If multiple tunnels to different peers are needed, the section can be used multiple times.

To perform actions on IPsec state changes, scripts can be deposited in the /etc/scripts.d/ipsec-hooks/ directory. There can either be an executable file or a directory with one of the names prepare-host, prepare-client, route-host, route-client, unroute-host, unroute-client, up-host, down-host, up-client, down-client. If it is a directory, all executable scripts within named *.sh are executed.

The scripts are executed at the respective state change, and all information is passed in environment variables.

The scripts can be defined through script sections.

Attribute	Value (Default)	Description
start	yes	If set to yes, IPsec daemon is started.
name		Name of the connection. This name is used in the config file /etc/ipsec.conf.
setup		Define the action to take on this connection upon IPsec start. Possible values are: start: try to setup connection route: prepare everything, but only start when traffic flows add: prepare, but wait for peer to setup connection
ike	2	Version of IKE protocol to use. Possible values: 1 or 2. Cisco devices (and Cisco VPN client) use version 1.
mode	tunnel	Use to choose IPsec operating mode. Possible values are tunnel or transport. If not set, defaults to tunnel. If set to transport, both remote and local must be set, while local_net and remote_net are ignored.
fragmentation		If set to yes, large IKEv2 messages are fragmented. If set to no, IP must fragment inside tunnel. Only relevant for IKEv2.
mobike		Enable MobIKE (RFC 4555) extension. Used to renegotiate a tunnel if the IP address of one endpoint changes. Only for IKEv2.
remote	192.168.17.42	IP address or host name of peer. If using a host name, enclose in " and make sure it can be resolved. To allow clients with unknown IP address (road warrior) to connect, set this to "any".
local		List of interfaces the IPsec daemon listens on for incoming connections.
local_within		If the route to remote goes over one of the interfaces listed here, the IPsec connection is started. If the interface is not listed, the connection is not started. If the parameter is not defined or

Attribute	Value (Default)	Description
		empty, the connection is started.
local_net		List of local networks or interfaces that can use the IPsec tunnel. If not set, the first configured interface from this list is taken: eth0, vlan1-4
Protocol		Limit IPsec tunnel to a single protocol and/or port. Both protocol and port can be specified by name or number. Syntax: [proto][,[sport][,dport]]. Port numbers are given for traffic to the peer; for traffic from the peer, the port numbers are exchanged. Example: protocol = tcp, http protocol = udp protocol = , 443 protocol = udp, 67, 68
remote_net	192.168.1.0/24	Network on the other side of the tunnel. Several networks separated by space can be given. In case of a road warrior, where the remote net is unknown, specify any, and the server will use the remote net as advertised by the client.
remote_range		When giving a network address here, road warriors can only connect if their IP address is in this range.
remote_address		The internal source IP address to use in a tunnel for the remote peer. This is needed for example with Cisco VPN client. If set to %config, the IP address proposed by the peer is echoed back to the peer.
tunnel		Defines which local_nets can communicate with wich remote_nets. If not set, all possibilities are allowed. All local_nets are labeled with a number (1, 2, …) All remote_nets are labeled with a letter (a, b, …) The parameter lists all pairs that are allowed. If it starts with a slash (/), the list contains the pairs that are prohibited. The parameter can be used to define source policy routing through the IPsec tunnel. If the parameter is followed by a colon an an optional interface or IP address, then upon completion of the tunnel a route is set to the remote net with the IP address of the given interface as source. If no source is given, the IP address of the interface on the local subnet is used. To set the source policy route for all connections, put the colon as the first character of the string, even before e possible "/". Example: local_net = loc1 loc2 remote_net = rem1 rem2 tunnel = 1a 1b 2b tunnel = /2a Both entries define that loc1 can connect to both rem1 and rem2, while loc2 can only connect to rem2. tunnel = 1a:eth0 1b:192.168.1.1 2a: 2b tunnel = :1a 1b 2b tunnel = :/2a tunnel = :/
dpd	5,5,5	Defines the parameters for dead peer detection. The values are: dpd_delay, dpd_retry, dpd_max dpd_delay: Send a DPD packet every N seconds dpd_retry: time in seconds until packet is considered as failed dpd_max: number of consecutively failed packets until peer is considered dead.

Attribute	Value (Default)	Description
dpdaction		Defines the action to perform when a dead connection has been recognized. Possible values: restart: try to re-establish the connection clear: delete and unroute the connection. It cannot be reestablished afterwards hold: hold the connection
tries	0	How many attempts should be made to negotiate a connection, or a replacement for one (DPD), before giving up. Can be a positive integer value, or 0 for forever (default).
closeaction		What to do if peer properly closes tunnel: restart: try to reestablish tunnel clear: close connection with no further action hold: install trap to reestablish tunnel upon traffic
my_identifier	asn1dn,	How to identify with the peer (the comma after asn1dn is crucial). Using hostname: fqdn, NAME Using address: address, ADDRESS Using certificates: asn1dn, When using address or fqdn with certificate based authentication, the address or fqdn must be present as subjectAltName in the certificate, otherwise the AnyRover cannot match the certificate to the ID.
peers_identifier	asn1dn,	How the peer identifies itself. Cf. my_identifier
auth_method	cert	Authentication method for IKEv1. psk: Pre shared key cert: Certificates xauth-psk: XAUTH using pre-shared key xauth-cert: XAUTH using certificates
local_auth remote_auth		Authentication method for IKEv2. any: (default for remote_auth if not set) psk: pre-shared key pubkey: certificates
psk	mysupersecretkey	Value of the pre shared key.
xauth		Specify the role in XAUTH authentication. Possible values are server and client. This is relevant only if auth_method is xauth-psk or xauth-cert.
xauth_id		Username and password pair for XAUTH authentication. This parameter can appear multiple times. Syntax: xauth_id = username:password
cert	ipsec-cert	Certificate for identification. References a [certificate] section.
root	ipsec-root	Root certificate for identification. References a [certificate] section.
key	ipsec-key	Private key for identification. References a [certificate] section.
pkcs11		Private key in PKCS#11 storage (TPM). Syntax: pkcs11 = file:<filename>, <PKCS#11 user password> pkcs11 = cert:<sectname>, <PKCS#11 user password> pkcs11 = id:<key_id>, <PKCS#11 user password> <filename>: Name of file containing hex ID of PKCS#11 key <sectname>: Name of [certificate] section referencing file containing the hex ID of PKCS#11 key. <kex_id>: hex ID of PKCS#11 key.
crl	ipsec-crl	Certificate revocation list. References a [certificate] section.

Attribute	Value (Default)	Description
ph1_encryption ph2_encryption	aes 256 aes 256	Encryption algorithm for phase 1 (ISAKMP-SA) and phase 2 (IPsec-SA). Supported algorithms: aes, twofish, blowfish, 3des Key length can be given after the algorithm, separated by space. Only use key lengths that are supported by the respective algorithm: aes, twofish: 128 (default), 192, 256 blowfish: 40-448 (default: 128) 3des: 168 (fix) The 3des algorithm is quite old and should not be used any more.
ph1_hash_alg ph2_hash_alg	sha256 sha256	Hash algorithm for authentication for phase 1 (ISAKMP-SA) and phase 2 (IPsec-SA). Supported algorithms: md5, sha1, sha256, sha384, sha512 md5 and sha1 are not considered secure any more.
ph1_prf		Defines the PRF algorithm to use for phase 1. If not defined, use the same algorithm as for hashing. Possible values: md5, sha1, sha256, sha384, sha512, aesxcbc, aescmac
ph1_strict ph2_strict		If set to yes, only accept the algorithms defined above. Otherwise, all supported algorithms are accepted. This should only be used for debugging purposes, to find out how a peer is configured.
ph1_lifetime ph2_lifetime	time 86400 sec time 3600 sec	Lifetime for phase 1 (ISAKMP-SA) and phase 2 (IPsec-SA). Lifetime defines the time after which new keys are generated. The keyword time gives a time span, the unit can be sec, min, and hour.
dh_group	5	Diffie-Hellmann group for encryption phase 1 (ISAKMP-SA). The higher the number, the more secure, the slower calculations. Possible values: 1, 2, 5, 14, 15, 16, 17, 18 Has to be the same on both ends. Group 1 is not considered secure any more.
pfs_group	5	Diffie-Hellmann group for phase 2 (IPsec-SA). Cf. dh_group. If no value is given, no pfs is used (not recommended). PFS (perfect forward secrecy) ensures that after a change of keys in phase 2 the new keys cannot be derived from the old ones.

4.3.21 [certificate]

The certificates that are defined in a [certificate] section can be used for IPsec and for OpenVPN connections. The [certificate] section has to be placed after the respective [ipsec] or [openvpn] section to be found.

Attribute	Value (Default)	Description
name		The name identifies the certificate and is referenced in the [ipsec] or [openvpn] section.
type	pem	Certificate type. Possible values are pem, p12 and file. Encrypted p12 files are not supported.
file		File that contains the certificate. Only valid if type=p12 or type =file. P12 only works with OpenVPN.
-----BEGIN MIICWwIBAAKB...		The rest of the section between the lines -----BEGIN

-----END	and -----END is the certificate.

4.3.22 [openvpn]

Attribute	Value (Default)	Description
start_server	yes	If set to yes, start OpenVPN server.
start_client	no	If set to yes, start OpenVPN client.
basic_server	yes	If set to yes, basic options for OpenVPN server are used.
basic_client	yes	If set to yes, basic options for OpenVPN client are used. These options are sufficient to connect to an IPCop machine.
server_net	192.168.0.0 255.255.255.0	Virtual network of the OpenVPN connection.
server_remote_net	192.168.2.0/24	Networks of the peer (client). A route to these nets is defined on the host. The list contains network/prefix pairs separated by space.
push_local_net	192.168.2.0/24	If set, pushes a route to these nets to the client. The list contains network/prefix pairs separated by space.
push_default	yes	If set to yes on the client, the default route is set through the tunnel.
remote	vpnserver.example.org	IP address or host name of the OpenVPN server. If the server doesn't listen on the default port (UDP 1194), the correct port can be appended, separated by colon. vpnserver.example.org:1234
client_remote_net		Networks of the peer (server). A route to these nets is defined on the host. The list contains network/prefix pairs separated by space. Note: these routes can be pushed by the server, cf. push_local_net.
server_auth_method		Authentication method when running as a server. Can be one of cert for certificate based authentication or psk for pre-shared key. Hint: if using pre-shared key, the key must be generated using openvpn and entered into a certificate section: openvpn --genkey --secret key.file
client_auth_method		Same as server_auth_method when running as client.
server_cipher		Cipher to use for encryption. Earlier versions used BF-CBC, which is no longer recommended. For better security, use AES-128-CBC. To see all available ciphers, run openvpn --show-ciphers
client_cipher		Same as server_cipher when running as client.
server_psk		If server_auth_method is set to psk, enter the reference to the certificate section containing the pre-shared key.
client_psk		Same as server_psk when running as client.
server_cert	server-cert	Certificate for the operation as server. References a [certificate] section that must be placed after the [openvpn] section.
server_root	server-root	Root certificate for the operation as server. References a

Attribute	Value (Default)	Description
		[certificate] section that must be placed after the [openvpn] section.
server_key	server-key	Private key for the operation as server. References a [certificate] section that must be placed after the [openvpn] section.
client_cert	client-cert	Certificate for the operation as client. References a [certificate] section that must be placed after the [openvpn] section.
client_root	client-root	Root certificate for the operation as client. References a [certificate] section that must be placed after the [openvpn] section.
client_key	client-key	Private key for the operation as client. References a [certificate] section that must be placed after the [openvpn] section.
server_option client_option		Additional options for client and server that are directly placed in the configuration files. When using BF-CBC, using this parameter is recommended to counter SWEET32 attacks: client_option = reneg-bytes 64000000

4.3.23 [clientconfigfile]

Using [clientconfigfile] sections, OpenVPN client config files can be placed in the configuration file. The attribute must be present at the beginning of the section, the rest up to the next section start is copied to the script file. Lines in the script file must not start with an opening square bracket '['.

Attribute	Value (Default)	Description
file		Name of the config file. The filename must be specified without a path. The file is stored in the directory /etc/openvpn/ccd.

4.3.24 [tunnel]

This section defines one IP-in-IP or GRE tunnel. If multiple tunnels are needen, every tunnel is defined in it's own [tunnel] section.

Attribute	Value (Default)	Description
name		Name of the tunnel. The tunnel interface will be named like this, so the name must not be tunl0, gre0, eth0, or ppp0.
start		If set to yes, tunnel is started.
type	gre	Tunnel type. Possible values are ipip (IP-in-IP tunnel), GRE (gre tunnel), and sit (IPv6 in IPv4 tunnel).
local		IP address or name of local tunnel endpoint. The remote endpoint is only contacted through this interface, so If the route to the remote server is not through this interface, tunnel setup will fail.
remote		IP address of the remote peer. Hint: this tunnels will not work if one of the endpoints is behind a NAT gateway. Use IPsec or OpenVPN in this case.
remote_net		Address of the networks that are behind the tunnel. Multiple networks are separated by space. The address is given with prefix, i.e. 192.168.17.42/24
vlocal		Address of the tunnel interface on the virtual network. This is a

	host address with prefix.
vremote	Address of the remote tunnel interface on the virtual network. This is a host address.

4.3.25 [bridge]

This section defines bridges between logical and physical interfaces. This section can appear multiple times, every section defines one bridge.

Some rule for using bridges:
- An interface can only be part of at most one bridge
- If an interface is part of a bridge, it cannot be used directly anymore

The STP protocol takes by default approx. 50 seconds to react on a topology change. By setting the values hello=1, age=4 and fw_delay=4 on every device taking part in STP negotiations, this time can be reduced to 12 seconds.

Attribute	Value (Default)	Description
name		The name of the bridge. The name must not collide with any other interface. It is best to used names like br0, br1...
start	no	The bridge is only created if this parameter is set to yes.
ipaddr		IP address and netmask or prefix of the bridge interface. The address can be given as 192.168.1.3/24 or 192.168.1.3 255.255.255.0. Using the parameter mtu:1492, the MTU of the interface can be set. To dynamically configure the interface, dhcp can be configured. If the value is "dhcp default", the dhcp client will also set the default route. More possible parameters after dhcp: metric:M sets the metric of the route (default: 0) timeout:T sets the timeout to T seconds (default: 30) dns: Queries the DHCP server for DNS server addresses, and replaces current DNS configuration hostname: queries the DHCP server for a hostname,and replaces the hostname if it is localhost.
iface		List of interfaces separated by space, that are part of this bridge.
stp	no	If set to yes, the spanning tree protocol (STP) is enabled on the bridge. All further parameters concert STP and are ignored if this is set to no.
prio		Priority of the bridge for the election of the root switch.
portprio		List of interface:priority values. This is used to set the priority of the interfaces. Example: portprio = vlan2:34 vlan3:77
hello	2	Hello timer of the STP protocol.
age	20	Ageing timer of the STP protocol.
fw_delay	15	Forward delay timer of the STP protocol.
cost		List of interface:cost values. This defines the costs of the individual paths. Example: cost = vlan2:46 vlan3:84

4.3.26 [banner]

Attribute	Value (Default)	Description
start		The banner is only shown if this is set to yes.

The whole text of this section (after the start attribute) is printed upon login (on the console as well as on network login). The first line starting with '--- END MOTD ---' or the start of the next section ends the message (and is not included itself).

4.3.27 [daemons]

Attribute	Value (Default)	Description
start		Defines a program that is started at the end of the boot process. This program can be a script defined via a [script] section. The attribute start can appear multiple times, the scripts are started in the order they appear in the config file.
stop		Defines a program that ist started upon system shutdown. Do not place long running programs here, as the system will not wait for them to terminate but proceed shutting down.

4.3.28 [script]

Using [script] sections, arbitrary scripts or other text files can be placed in the configuration file. The three possible attributes must be present at the beginning of the section, the rest up to the next section start is copied to the script file. Lines in the script file must not start with an opening square bracket '['.

Hint: Files in /etc/scripts.d/ are deleted upon system boot and recreated. Files not in this directory are **not** automatically deleted, espescially not if the section is removed from the config file. It is thus not recommended to place files in other directories than /etc/scripts.d/ as this can have hard to find side effects if the configuration is changed later on.

Attribute	Value (Default)	Description
name		Name of the section. Currently not used.
file		Name of the file. If the name begins with a slash '/', the path is taken absolute, otherwise it is taken relative to /etc/scripts.d/. Non-existing directories are automatically created.
mode		Mode of the file, given as standard Unix file modes. For scripts that are executed, use 755, otherwise 644 is fine. If the value is of the form Link:FILENAME, then the name of file is created as a symlink to FILENAME, and the rest of the section is ignored. In this case, the mode parameter must appear after the file parameter in the config file.

4.3.29 [webserver]

Configuration for the web server and the WebGUI.

Attribute	Value (Default)	Description
start	yes	Start the web server?
port	80	TCP port to listen on.
interface	all	Network interface to listen on. Can be one of eth0,ppp0,vlanX,brX, an IP address, or all.
document_root	/usr/share/www	Document root for the web server.
user		User the web server is run as. If not specified, nobody is used.
group		Group the web server is run as. If not specified, nogroup is used.
access_log	/var/log/boa/ access_log	Log file of the web server where all accesses are logged.
error_log	/var/log/boa/ error_log	Log file of the web server where all errors are logged.
default_mime		The MIME type of files is determined according to file extension. For missing or unknown extensions, the type can be specified here. Default for unknown files is text/plain. Examples: text/html, application/x-httpd-cgi
option		These options are directly written to the boa.conf file.
anygator		If set to yes the AnyGator scripts are activated.

4.3.30 [wlan]

This section configures the WLAN connection. It has no effect if no WLAN card is included in the AnyRover.

The WLAN card can be operated in access point, client mode, or mesh (pre IEEE 802.11s) mode. Some options in this section are not for all modes. In the table, the mode specific options are marked with (AP) for access point, (CL) for client, and (M) for mesh. Unmarked options are used in all modes.

This section can appear multiple times.

It is possible to run multiple SSID on a single access point. To do this, a separate section must be defined for every SSID, where the device name of further sections must have the device name of the firsts section as a prefix (e.g. first section: device = wlan0; second secstion: device = wlan0_1). Parameters concerning the radio must not be redefined (e.g. channel).
The additional devices will appear as network interfaces in the system, and can be used for routing, firewalling, DHCP server etc.

In access point and client mode, scripts are called for every connect and disconnect event. These scripts are placed in /etc/scripts.d/wlan-ap-hooks (for AP mode) and /etc/scritps.d/wlan-client-hooks (for client mode). The scripts have these parameters:

```
interface cmd [clientMAC]
```

interface defines the interface the event occurred on, cmd has the values AP-STA-CONNECTED and AP-STA-DISCONNECTED for AP mode, or CONNECTED and DISCONNECTED for client mode. In AP mode, the third parameter is the MAC address of

the client.

Attribute	Value (Default)	Description
start	yes	Start the wlan card? Make sure to enable power on the USB bus on the usb section.
mode		Select the mode. Possible values: ap, client and mesh
device		Defines the wireless device. For the optional internal WLAN card, this is wlan0. If the device is to run as a standalone Radius server, this parameter is set to none (and mode to ap).
country		Defines the country the system is operated in. This is used to select the valid WLAN channels. Possible values: ch, de, fr, us, ...
channel		Select WLAN channel. Can be a list of channel in (CL) mode.
ipaddr	dhcp	IP address and netmask or prefix of the WLAN interface. The address can be given as 192.168.1.3/24 or 192.168.1.3 255.255.255.0. Using the parameter mtu:1492, the MTU of the interface can be set. (CL, M) To dynamically configure the interface, dhcp can be configured. If the value is "dhcp default", the dhcp client will also set the default route. More possible parameters after dhcp: metric:M sets the metric of the route (default: 0) timeout:T sets the timeout to T seconds (default: 30) dns: Queries the DHCP server for DNS server addresses, and replaces current DNS configuration hostname: queries the DHCP server for a hostname,and replaces the hostname if it is localhost.
ssid		(AP, CL) SSID of the network. Can be an ASCII string or a hex value. Hex values must start with 0x.
key_management	WPA-EAP	(AP, CL) Key management protocol. Possible values are: WPA-PSK, WPA-EAP, FT-PSK, FT-EAP, IEEE8021X, NONE FT-* values for Fast BSS Transition (IEEE802.11r). Multiple values can be given separated by space.
pairwise	TKIP	(AP, CL) List of accepted pairwise (unicast) ciphers. Possible values are: CCMP, TKIP, WEP104, WEP40 If not set defaults to all.
wep_key		(AP, CL) WEP keys. Up to four WEP keys can be entered, each on a separate line. The keys can be ASCII text or a hex value (starting with 0x).
wep_default_key		(AP, CL) Index of key to use for transmission. Possible values: 0-3
eapol_version	1	(AP, CL) Many APs only support EAPOL v1.
scan_ssid	no	(CL) If set to yes, the network is scanned with SSID specific frames. This is used if the access point does not broadcast its SSID. Do not enable if not needed, since it increases latency while scanning.
scan_interval		(CL) Interval in seconds for periodic AP scans when not connected to an AP.
scan_short scan_long scan_threshold		(CL) Interval parameters for AP scanning when connected to an AP. scan_threshold is the signal threshold in dB (negative value) that

Attribute	Value (Default)	Description
		separates between long and short interval. scan_short is the interval in seconds that is used when the signal level is below (i.e. worse) than threshold. scan_long is the interval in seconds that is used when the signal level is above (i.e. better) than threshold. If scan_long is not defined, the same value as for scan_short is used. Note: a scan takes approximately 15s, so setting these intervals to values smaller than ca 20s makes no sense and will probably not work as excpected.
pre_shared_key		(CL) Pre-shared key for the network. Can be an ASCII string or a hex value. Hex values must start with 0x.
eap	PEAP	(CL) Space separated list of accepted EAP methods. Possible values: MD5, MSCHAPV2, OTP, GTC, TLS, PEAP, TTLS
group	TKIP	(CL) List of accepted group (broadcast/multicast) ciphers. Possible values are: CCMP, TKIP, WEP104, WEP40 If not set defaults to all.
anonymous_identity		(CL) Identity string for EAP phase 1 authentication.
identity		(CL) Identity string for EAP phase 2 authentication.
password		(CL) Password for EAP phase 2 authentication.
root		(CL) Root certificate for certificate based authentication. References a [certificate] section.
cert		(CL) Certificate for certificate based authentication. References a [certificate] section.
key		(CL) Private key for certificate based authentication. References a [certificate] section.
pkcs11		(CL) Private key in PKCS#11 storage (TPM). Used instead of key= Syntax: pkcs11 = file:<filename>, <PKCS#11 user password> pkcs11 = cert:<sectname>, <PKCS#11 user password> pkcs11 = id:<key_id>, <PKCS#11 user password> <filename>: file containing hex ID of PKCS#11 key <sectname>: name of [certificate] section referencing file <key_id>: hex ID of PKCS#11 key
phase1		(CL) Phase 1 (outer authentication) parameters.
phase2	auth=MSCHAPV2	(CL) Phase 2 (inner authentication) parameters.
mesh_id		(M) The mesh ID. All stations taking part in the mesh must have the same mesh ID.
wpa		(AP) Selection of WPA standard. Possible values are wpa and wpa2. If not set, no WPA is used.
broadcast_ssid	yes	(AP) If set to no, the SSID will not be broadcast. Clients can only connect if they know the SSID. Is not suitable as security element, since this will not hinder an attacker.
ieee80211d	no	(AP) The AP advertises its regulatory domain according to standard IEEE 802.11d.
Ieee802.11n	no	Use IEEE 802.11n. If set to yes, set hw_mode = g for a 2.4GHz access point or hw_mode = a for a 5GHz access point.
hw_mode		(AP) Select one of 802.11a,b,g. Only place the letter (a,b,g) here.
txpower		(AP) Allows to set TX power to some value between 0dBm and 20dBm. Default value is 20dBm.

Attribute	Value (Default)	Description
macacl		(AP) Enable or disable MAC address filter. Possible values. no: MAC address filter is disabled. accept: Clients can connect unless the MAC address is in the deny list. deny: Clients cannot connect unless the MAC address is in the accept list.
acl_accept acl_deny		(AP) List of accepted or denied MAC addresses. These paramters are only relevant if macacl ist set accordingly. macacl=accept: acl_deny is active macacl=deny: acl_accept is active The value is the MAC address. For multiple addresses, use multiple entries. Example: acl_accept = 00:11:22:33:44:55
cap_htgf		(AP) For 802.11n. Enable High Throughput Mode (Greenfield Mode). This mode should only be used if no 802.11b/g clients are connecting, otherwise the network will not work reliably.
cap_40mhz		(AP) For 802.11n. Enable support for 40MHz channels. Can be set to 40- or 40+. If set to 40-, then only channel 5-13 can be used with 2.4GHz and 40,48,56,64 for 5GHz. If set to 40+ it is channel 1-7 for 2.4GHz and 36,44,52,60 for 5GHz.
cap_short_gi		(AP) For 802.11n. Enable support for Short Guard Interval. This can increase the data rate up to 11%, at the cost of a less stable network and increased packet collisions.
cap_rx_stbc		(AP) For 802.11n. Define the number of receiving antennas used. Possible values: 1, 2
cap_amsdu		(AP) For 802.11n. Enable Frame Aggregation. Results in an increased user level data rate.
wpa_psk		(AP) WPA pre-shared key. Defines the pre-shared key for PSK. The key can be an ASCII-string (8..63 characters) or a hex value (64 hex digits). If wpa_psk is given, wpa_psk_entry is ignored.
wpa_psk_entry		(AP) WPA pre-shared key for a specific MAC address. This entry can appear multiple times. Syntax: MAC KEY The MAC address 00:00:00:00:00:00 can be used to match every client. If wpa_psk is given, wpa_psk_entry values are ignored.
ieee8021x	no	(AP) Enable or disable 802.1x (yes or no)
authentication		(AP) References a [authentcation] section for an EAP or RADIUS server.
use_radius_server		(AP) Enable or disable access to external Radius server (yes or no)
source_addr		(AP) Define the source IP address to be used in the communications to the Radius server.
radius_ipaddr		(AP) IP address of the access point. This address is used as NAS-IP address. If omitted, the IP address of the WLAN card is used.
radius_server		(AP) IP address and port of the Radius server. If no port is given, 1812 is used. Multiple Radius server can be configured by repeating this and the next parameter. Example: radius_client_server = 192.168.99.4:1812

Attribute	Value (Default)	Description
radius_secret		(AP) Password for accessing the Radius server. The password always belongs to the last server address defined previously.
radius_accounting		(AP) IP address and port of the accounting server. If no port is given, 1813 is used. Example: radius_client_accounting = 192.168.100.3:1813
radius_acct_secret		(AP) Password for accessing the accounting server. The password always belongs to the last server address defined previously.
radius_retry		(AP) Interval in seconds after which the first Radius server is tried again. If not given, the servers are used in order, and the system changes to the next one if the current server is not reachable any more.

4.3.31 [authentication]

This section defines authentication servers. These can be an internal EAP server for a WLAN access point, the connection details for an external RADIUS server, or a standalone RADIUS server. The section can be present multiple times. An integrated EAP server for a WLAN access point can be defined alongside a standalone RADIUS server.

Attribute	Value (Default)	Description
name		Unique name of the section. This parameter must be set and must be the first parameter in the section.
start		The section is only evaluated if set to yes.
standalone		Set to yes if this section defines a standalone RADIUS server. Set to no if it is referenced from a wlan section.
eap_phase1_id		Defines paramters for phase 1 authentication. Syntax: eap_phase1_id = type [username[:password]] type is one of TLS, TTLS, or PEAP Username and password can be specified if needed.
eap_phase2_id		Defines parameters for phase 2 authentication. Syntax: eap_phase2_id = type [username[:password]] type is one of MSCHAPV2, MSCHAP, CHAP, or PAP for PEAP, as well as the same values with prefix TTLS- for TTLS. Username and password can be specified if needed.
root_cert		(AP) Root certificate. This value references a [certificate] section.
server_cert		(AP) Server certificate. This value references a [certificate] section.
server_key		(AP) Server private key. This value references a [certificate] section.
radius_start		Set to yes if defining a standalone RADIUS server. The following parameters are only evaluated if this is set to yes.
radius_server	0.0.0.0	Address the RADIUS server listens on for connection requests.
radius_acct_server	0.0.0.0	Address the RADIUS accounting server listens on for connection requests.
radius_port	1812	UDP port the RADIUS server listens on for connection requests.
radius_acct_port	1813	UDP port the RADIUS accounting server listens on for connection requests.

Attribute	Value (Default)	Description
radius_client		List of IP or network address (with prefix) that are granted access to the RADIUS server. Multiple addresses can be specified separated by space. This parameter can appear multiple times to allow different addresses with different passwords.
radius_secret		Password for the access to the RADIUS server. This parameter always belongs to the last specified radius_client list.

4.3.32 [ospf]

Configuration of the Open Shortest Path First (OSPF) routing protocol.

Attribute	Value (Default)	Description
start	yes	Start OSPF daemon.
router-id		Router ID for OSPF. This value can be an IP address or an interface name. If an interface name is given, the first IP address of this interface is used as router ID. For the loopback interface this will be 127.0.0.1.
insert-default		If set to yes, the system default route is advertised through OSPF.
area		OSPF area definition. The definition contains the area number and a list of interfaces that belong to this area. Example: area = 0, vlan1, vlan2
stub		Stub area definition. Identical to area definition, but the area is marked as stub.
passive		The network of this interface is advertised through OSPF, but the protocol is not run on this interface. The value must be an interface name.
auth		Authentication options. These options contain the area number, the authentication type and a list of interface:[id:]key tuples. The authentication key can be one of key or md5. The key is defined as interface:key for type key, or interface:id:key for type md5. The id must be consistent across routers on a link. Examples: auth = 0, key, vlan1:verysecret, vlan2:evenmoresecret auth = 1, md5, vlan2:2:unbreakable
range		Route summarization. This parameter is only valid on ABRs (Area Border Router). If networks in an area are contiguous, the router can advertise a route summary into other areas. The definition contains the area id, and a list of summarized networks.

4.3.33 [snmp]

This section defines the Simple Network Management Protocol (SNMP). The AnyRover contains an SNMP agent that can answer SNMP requests and trigger SNMP traps.

Attribute	Value (Default)	Description
start	yes	Start SNMP daemon.
listen		Defines interfaces and ports for SNMP to listen on. listen = [proto:][interface/address:][port] proto can be udp or tcp, interface can be the name of an

Attribute	Value (Default)	Description
		interface (e.g. eth0) or an IP address. Default is udp:0.0.0.0:161 Multiple combinations ca be listed separated by comma.
sysdescr		Text to be returned by SNMP in sysDescription.0.
location		Text to be returned by SNMP in sysLocation.0.
contact		Text to be returned by SNMP in sysContact.0.
services		List of services to be returned by SNMP in sysServices. Can be any combination of physical, datalink/subnet, internet, endtoend, application Instead of the list of services, the corresponding number can be given, as the sum of 1, 2, 4, 8, 64 (for the respective services).
views		Views are used to restrict access to certain subtrees of an OID. Views must be defined before they are used in user directives below. The first attribute is the name of the view, the second defines whether to add this OID to the view (included), or remove it (excluded). Multiple view lines can be used to add/remove several OIDs to one single view. Syntax: view = NAME, [included|excluded], OID
user		User management. This parameter defines SNMP users. SNMP v1 and v2c User = SNMP vers,{ro|rw},community[,src[,OID]] SNMP v3 User = SNMP vers,{ro|rw},user:pw1[:pw2][,hash:enc[,priv[,OID|View]]] For SNMP version 1 and 2, two optional parameters can be given: src: source address or network where SNMP requests may originate OID: limits access to the subtree rooted at this OID. Multiple OIDs can be specifies using views. For SNMP v3: If only pw1 is given, it is used both for authentication and privacy, otherwise pw2 is used for privacy. The flag priv may contain the values priv, auth or noauth. Setting the flag enforces the use of the requested encryption or authentication scheme. If not given, the use of authentication or encryption is optional. Hash and enc specify the algorithms to use. hash: MD5 (default), SHA enc: DES, AES (default)
process		Process monitoring process = name [, max [, min]] The process with the given name must be present in the process list between min and max times, otherwise the respective error flag is set.
exec sh extend		Execute arbitrary commands exec = [OID,] name, path [,arg [,arg]] When the OID is queried, the programm path is executed and its information returned. If an OID is given, the information is rooted at this place in the tree, otherwise it is returned in the extTable subtree. exec is used for binary programs, sh for shell scripts. Extend is an improved form of exec and sh, where the results are returned in two tables, once the full output as a single string,

Attribute	Value (Default)	Description
		and once every line separately.
trapcommunity		Defines the default community for traps.
trapagent		When querying variables for traps, the agent generates internal SNMPv3 requests using this username. This user must exist and have readonly access.
trapsink trap2sink		Destination for traps trapsink = [tcp | udp]:(IP addr/hostname)[:port][,community] If no community is given, SNMP uses the community defined in trapcommunity. Default protocol is UDP, default port 162.
authfail		Defines whether Authentication Failure Traps should be generated (yes or no).
updown		Defines whether Interface Up/Down Traps should be generated (yes or no).
monitor		Defines a monitor for a MIB object. monitor = name, expr [, action [, user [, freq [, oid [, oid]]]]] The name must be unique for all monitors. Expr has the form: OID | !OID | !=OID | OID OP value | OID min max where OP is one of: ==, !=, <, <=, >, >= action is the name of an action to perform if monitor triggers (see below). Freq is the interval in seconds between two consecutive checks of expr. (Default: 600) If no action is given, a default notification is sent. If action is notification (either default or via action attribute), additional OIDs can be given, which are sent along.
action		Defines an action to perform when a monitor triggers. action = name, type, value [, oid [, oid]] The name is used in the monitor attribute to identify the action. Type can be either set or notify. If type is set, then value has the form OID = value, and the given OID is set. If type is notify, the value is the type of notification message that is sent, and is one of: coldStart, warmStart, linkDown, linkUp, authenticationFailure, egpNeighborLoss, enterpriseSpecific Further OIDs are sent along in the trap message.

4.3.34 [dns]

This section defines DNS proxy and server settings. DNS server is currently not supported.

Attribute	Value (Default)	Description
start_proxy	no	Start DNS proxy.
proxy_basic		Set to yes to enable some basic parameters: block unnecessary queries in Windows, addresses from private IP address ranges and plain hostnames without domain. Do not enable this when using Kerberos, SIP, XMMP, or Google-talk.
proxy_interface		Comma separated list of interfaces where DNS proxy shall listen for queries. If the list starts with '/', it defines the interfaces where not to listen. If left empty, the DNS proxy listens on all interfaces.

Attribute	Value (Default)	Description
		This parameter should not be used together with proxy_address.
proxy_address		List of IP addresses the DNS proxy listens on for queries. If left empty, the proxy listens on all addresses. This parameter should not be used together with proxy_interface.
proxy_port		Port where the DNS proxy listens for queries. Default: 53
proxy_domain		If set, this domain name is appended to all plain hostnames before sending them to the DNS server.
proxy_param		More parameters can be put here. Some useful parameters are - strict-order: query the name servers strictly in the order they appear in the /etc/resolv.conf file. - all-servers: query all servers at the same time. If not set, they are queried one after the other until one answers.
static_host		Define static host name entries to prevent DNS lookups. Syntax: static_host = hostname, IP address Example: static_host = google-public-dns-a.google.com, 8.8.8.8

4.3.35 [serports]

This section configures the serial ports

Attribute	Value (Default)	Description
enable	no	Enables or disables the serial ports

4.3.36 [openconnect]

OpenConnect is a client for Cisco's AnyConnect SSL VPN.

OpenConnect is not officially supported by, or associated in any way with, Cisco Systems. It just happens to interoperate with their equipment.

Attribute	Value (Default)	Description
start		Defines whether to start openconnect
remote		Address of the remote server, in the form https://server.example.org or https://192.168.20.12
username		Username to log in to the VPN.
password		Password to log in to the VPN.
check_certificate		Openconnect complains and asks for confirmation if it cannot verify the server certificate. Setting this parameter to no prevents this check.

4.3.37 [mobileip]

The AnyRover can play Mobile Node and set up a MobileIP connection to a Home Agent (HA). MobileIP is a VPN technology that can switch the tunnel to a new link within

seconds if the current link is no longer available, or if a higher prioritised link becomes available.

Data in a MobileIP tunnel is not encrypted, for this an IPsec tunnel is usually used.

The Mobile Node keeps a list of available interfaces for the connection to the HA. If the current connection is no longer available, it will try the other possible routes one by one, until it finds a new one. With larger lists of possible uplinks, this can lead to longer delays until a switch is completed. It takes approx. 3 seconds per link to be tested.

MobileIP will call all hook scripts in /etc/scripts.d/mip-hooks/ whenever it registers with the Home Agent, with the arguments "(RE)CONNECT" and the name of the interface the tunnel runs through. The first time it creates the tunnel, the first argument is "CONNECT" and IPsec will also be kicked, for every other call, the first parameter will be "RECONNECT". These scripts can be used to set up routing through the tunnel.

Attribute	Value (Default)	Description
start		Defines whether MobileIP is started.
mode	mn	Defines the role of the devices. Currently, only Mobile Node is supported. mode = mn
ha		Die IP address of the Home Agent.
hoa		The Home Address of the Mobile Node.
ign_interface		A list of network interfaces (comma separated) that must not be used to connect to the HA. The interfaces lo, tunl0 and gre0 are by default excluded from use. To enable one of those, put them in the list prefixed with a slash (/gre0).
routing	default	Defines the routing to be set up after the tunnel is established. Possible values: default, none, {network} Default: Set up a default route through the tunnel. None: Do not set up any routing. All necessary routes have to be set up using some kind of scripts. If a network address is given, a route to this network is set up.
spi		The Security Parameter Index, defines the Security Association on the HA for this connection. Can be given decimal or in hex (prefixed with 0x). Example: spi = 0x10a
auth	hmac-md5	Authetication algorithm. Possible values: md5-prefix-suffix, hmac-md5, sha1, hmac-sha1 md5-prefix-suffix does not work with Cisco HA devices, use hmac-md5 in this case. Additionally, md5-prefix-suffix has known weaknesses.
secret		The shared secret for tunnel authentication with the HA. According to RFC2002, the secret has 16 or 32 bytes length, but other lengths are also supported here. The secret can be given as text or hex-number, prefixed with 0x.
replay	timestamp	Method for replay protection. Possible values: none, timestamp, nonces
lifetime	3600	Tunnel life time in seconds. After this time, a new registration request is sent to the HA. Values >=65535 mean infinity, i.e. no

Attribute	Value (Default)	Description
		new registration.
udpport	434	UDP port to send registration requests to. RFC says port 434.
udpsrcport		UDP source port to use as source in the communication with the HA. If not set, a random port is used.
interval	200	Tunnel keepalive interval. An active tunnel is probed regularly. This value defines the minimum interval between probes in milliseconds.
linkdown	3	A tunnel is considered down after this number of consecutive keep alive probes are not responded to.
tunnel_rtt	500	Initial tunnel round trip time in milliseconds. The RTT is constantly adjusted to the real values, but never set below 200ms.
percentage	120	If an answer to a keep alive is not received during this percentage ot the RTT, it is considered lost.
link_priority		The Mobile Node keeps a list of currently available default routes, sorted by routing metric. If link priority is activated, the mobile node tests higher prioritised routes (i.e. lower metric) for availability and switches in case it finds one. If link priority is not used, the Mobile Node will only switch to another link if the currently used is no longer available.
link_prio_icmp	yes	This parameter has no effect if link_priority is disabled. If set, the Mobile Node will send ICMP Echo Requests to the HA to check for link availability.
link_prio_reg_valid	no	This parameter has no effect if link_priotiry=no or link_prio_icmp=yes. To check for link availability, the Mobile Node will send Registration Requests to the HA. This parameter defines whether these Registration Requests are valid. Valid Registration Requests will cause the HA to switch to the new link immediately, whereas the Mobile Node is not actually ready yet to use the new link, resulting in a short interruption of the link. Additionally, this behavior does not fit well with the delayed switching as configured with the next two parameters. Invalid Registration Requests have a timestamp which is 10 years old, causing the HA to respond with a Registration Denied message. But to check link availability, this is sufficient. As soon as the Mobile Node intends to switch link, it will send another (valid) Registration Request to the HA.
link_count	2	This parameter defines how many successful link checks the Mobile Node must receive until it switches to the newly available link. Using this and the next parameter, it can be defined how quickly the Mobile Node switches to a new link.
link_interval	2	This parameter defines the interval for keep alive messages on higher prioritised links. Using this and the previous parameter, it can be defined how quickly the Mobile Node switches to a new link.

4.3.38 [scep]

Placing certificates (e.g. for IPsec) directly into certificate sections is unpractical if the device is operated longer than the certificates are valid, since it requires manual certificate replacement in the config file. In this case, certificates can be automatically obtained and renewed before expiry using SCEP (Simple Certificate Enrolment Protocol).

For every set of certificates, one [scep] section is used, it can therefore appear multiple times.

When a SCEP request is started, it loads the CA certificates, creates a private key and a certificate signing request (CSR) which is then submitted to the SCEP server to obtain the actual certificate.

After a SCEP request hast finished, hook scripts are started from /etc/scripts.d/scep-hooks/. For every section, specific hook scripts from /etc/scripts.d/scep-hooks/<name>/ where name is the section name, are also started.

The hook scripts get some information via environment variables. These variables are defined:

SCEP_NUMCERT = number of certificates to enroll
SCEP_SUCCESS = number of successfully enrolled certificates
SCEP_TIMEOUT = number of certificates where update failed due to server timeout
SCEP_SKIPPED = number of certificates that do not need to be enrolled yet

Checking of certificates for expiry is a very cheap operation, lasting only some tenths of a second and thus can be done regularly.

Only the actual enrolment of new certificates takes some time and can take well over a minute, although it is not very computing intensive, but creating a new private key requires enough random data to be available.

Attribute	Value (Default)	Description
name		Name of the section. The config file will be named according to this value. This parameter must be first in the section.
start		Defines whether this section is active.
check		Define the time table for certificate checking and enrollment. Basically, these values are entries into the crontab and have the same syntax. The cron daemon is started, even if [crontab] start=no is set. Additionally, these values can be defined: - daily TIME: check every day at specified time. - weekly DAY TIME: check every week on DAY at TIME - Defined events: ppp-up when 3G/4G connection goes up (ppp only), mip-up when MobileIP tunnel goes up the first time, ipsec-up when IPsec tunnel is established, dhcp <if> when interface <if> gets a lease, boot after completion of boot process, wlan <if> when the wireless client interface <if> connects to an AP. Syntax: on EVENT Examples: check = 1 15 * * * check = daily 9:30 check = weekly thursday 11:23

Attribute	Value (Default)	Description
		check = on boot check = on wlan wlan0
action		Predefined action to execute on successful enrolment: -ipsec: reload all IPsec connections so they use the new certificates. - bondix: restart Bondix tunnels.
debug		If set to yes, the SCEP client will log intensively. Only recommended for debugging purposes.
directory	/etc/certs/ipsec	Directory to store the new certificates. A request can be started with this directory empty; the CA certificates are then loaded first.
days	7	Number of days before expiry that a new certificate shall be created.
key_size		Size in bit of the private key, if not yet present. Possible values: 768, 1024, 2048, 4096. For PKCS#11 (TPM) based keys, use 2048.
key_type		Define the private key type. Possible values: rsa: file based key tpm: TPM based key (not suitable for use with IPsec and WLAN) pkcs11: PKCS#11 based key (TPM based). Suitable for use with IPsec and WLAN.
signature		Algorithm for key signing. Possible values: md5, sha1, sha224, sha256, sha384, sha512
server		URL of SCEP server. For Microsoft based servers, this looks like http://<IPADDR>/certsrv/mscep/mscep.dll
fetch_ca		If set to yes, CA certificates are fetched with every SCEP request. If not set, CA certificates are only fetched if not present locally.
encryption		Encryption to use when communicating with the SCPE server. Possible values: des, 3des, blowfish
ca-file		Name of the file to write CA certificate to. If multiple certificates are included, the name is extended with -2, -3, ... Additionally, a file name enc-<ca-file> is created.
password		Challenge password for communication with SCEP server.
old_cert_auth		If set to yes, use current certificate instead of password to authenticate with SCEP server.
CA-DN		Common Name of the CA certificate to use.
cert-file		Name of the file to save the certificate to.
key-file		Name of the file to save the private key to. Created with mode 0600.
keep		If set to yes, the Certificate Signing Request (CSR) is kept around after obtaining the certificates, otherwise it is deleted.
Country State Location Organization OrgUnit CommonName		Fields for Distinguished Name of certificate.
altname		Alternative name of the certificate.
quote_altname		If set to yes, the subjectAlternativeName in the CSR is set in

Attribute	Value (Default)	Description
		quotes (“).

4.3.39 [pelix]

The AnyRover can send position information directly to a Pelix server from LogObject. Further functions with Pelix servers are not implemented yet.

Note that data transmitted to the Pelix server is encoded in a Pelix specific protocol, but not encrypted. Encryption must be provided by some other means (e.g. IPsec tunnel).

Attribute	Value (Default)	Description
start		Defines if this section is active or not
listen		Define interface to receive GPRMC data. Default: listen = tcp, 127.0.0.1:13181 For this to work, a configuration entry to send GPS data must be present: [gps] target = 0,127.0.0.1:13181
target		Defines IP address and TCP port of the Pelix server. Currently only one server can be defined. Example: target = 192.168.1.1:11310
source		Source address to use when contacting PELIX server. Syntax: source = addr[:port] Example: source = 192.168.1.3
retry		Timeout in seconds after a failed attempt until retry. Default: 5
interval		Timeout in seconds between two consecutive position messages. Default: 10
coordinates		Define format of coordinates to submit to Pelix server. Possitble values: CH1903, WGS84 microdegrees, WGS84 Default: WGS84 microdegrees
id		ID of device to transmit to Pelix server. Usually the IMEI of the modem. This must be configured manually until further notice.
username		Username for login with Pelix server.
password		Password for login with Pelix server.
retransmit		Defines whether messages that are not acknowledged by the Pelix server will be retransmitted. Default: no

4.3.40 [login]

Attribute	Value (Default)	Description
passwd		If set to yes, login password is checked against local password file, but only after checking against RADIUS and TACACS+ server. Note: if set to no, make sure to enable another method, otherwise logging into the AnyRover is not possible anymore.
passwd_console		Like passwd, but only for console logins.
passwd_ssh		Like passwd, but only for ssh logins.
tacacs_blacklist	root, config	Blackliste for Radius or TACACS+ Logins. This parameter has to

Attribute	Value (Default)	Description
radius_blacklist		be at the top of the section. All user names listed in the blacklist file are not checked with the Radius or TACACS+ server, but only locally. This parameter only affects Radius or TACACS+ server configs that appear later in the config file.
radius		Login credentials for Radius server. Syntax: radius = server, key[, timeout] server: IP address or hostname of Radius server key: shared secret of Radius server timeout: if the Radius server does not answer within this interval (in seconds), the next entry is tried.
radius_console		Login credentials for console login by Radius server
radius_ssh		Login Icredentials for SSH login by Radius server
tacacs		Login credentials for TACACS+ server. Syntax: tacacs = server, key[,[timeout][,source[:port]]] server: IP address or hostname of TACACS+ server key: shared secret of TACACS+ server timeout: if the TACACS+ server does not answer within this interval (in seconds), the next entry is tried. source: source IP address to use when contacting TACACS+ server. port: source port to use when contacting TACACS+ server.
tacacs_console		Login credentials for console login by TACACS+ server
tacacs_ssh		Login credentials for SSH login by TACACS+ server
allow_unknown	yes	If this is set to yes, then users that have not yet been registered on the system may login when a TACACS+ or RADIUS server authenticate the user. The user is then created upon login and deleted again when the last session terminates. If not set, a user must be created on the system before login is possible.
allow_unknown_console		se postfix _console to limit to console.
allow_unknown_ssh		se postfix _ssh to limit to ssh service.
tally_count	5	Count number of unsuccessful logins and block further tries. The number of unsuccessful logins on console or ssh are counted and the account blocked for a while after reaching limit. tally_count enables the feature if set to value >= 0. The account is blocked after <n> consecutive login failures. The counter is reset on successful login, or with the command pam_tally2.
tally_root	yes	Defines whether counting for root user is also active.
tally_timeout	30	Defines the interval to block the account. If not set, account is blocked forever (or until reset with pam_tally2).

4.3.41 [802.1x]

Attribute	Value (Default)	Description
name	vlan2	Name must be the first argument and defines the interface, which this section belongs to. E.g. vlan2. This section has no effect if the parameter "ipaddr" belonging to this interface does not contain the keyword 8021x.

Attribute	Value (Default)	Description
mode	client	Supplicant or authenticator. Must be the second argument after name. Alternatively: client or server.
eapol_version	1	Defines eapol version
key_management	EAP	Defines key managment protocol. Possible values: PSK, EAP
eap	PEAP	List of EAP methods to use (comma separated): Possible values: PEAP, TLS, TTLS, MD5, MSCHAPV2
pre_shared_key		Defines pre shared key for PSK authentication
anonymous_identity		Identity string for EAP phase 1 authentication.
identity		Identity string for EAP phase 2 authentication.
password		Identity string for EAP phase 2 authentication.
phase1		Phase 1 (outer authentication, i.e. TLS tunnel) parameters. This is a string with field-value pairs, e.g. peapver=0
phase2		Phase 2 (inner authentication with TLS tunnel) parameters. This is a string with field-value pairs, e.g. auth=MSCHAPV2
root	8021x-root	Root certificate to use for cert based authentication. References a [certificate] section.
cert	8021x-cert	Certificate to use for cert based authentication. References a [certificate] section.
key	8021x-key	Private key for certificate. References a [certificate] section.
pkcs11		Private key for certificate in PKCS#11 store (TPM). Same syntax as in [wlan] section.
route		Only for authenticator if the interface is part of a bridge. If set to yes, the EAPOL packets are received on the physical interface, if set to no they appear on the bridge interface. Example: vlan1 (switch port 1) and vlan2 (switch port 2) are protected using 802.1X. They are in the same IP subnet, so they are bridged to br0. But the authenticator is run on vlanX interfaces. In this case set route=yes, then EAPOL packets will appear on the vlanX interfaces, all other trafffic (especially IP traffic) on br0.
port_mode		Define the port mode (only for authenticator). Possible values: single-host: only one host can be authenticated at a time. If a second host authenticates, the first host is deauthenticated. multi-host: When one client authenticates successfully, the port is open for all possible clients. Needed e.g. if another 802.1X enabled switch is connected to this port. The switch authenticates with the AnyRover, all Clients with the switch. multi-auth: Several hosts can authenticate on this port. Used when a non-802.1X switch is connected. If the cable is plugged, all hosts are deauthenticated.
mab		Mac Authentication Bypass. Define MAC addresses that do not have to authenticate, because the client does not support 802.1X (e.g. a printer).
standalone		If set to yes, the following parameters can be used to define some local EAP credentials.
eap_phase1_id		EAP phase 1 credentials. Syntax: TYPE [user[:password]] Possible values for TYPE: PEAP, TTLS, TLS If no username is given, the rule matches all incoming requests.

Attribute	Value (Default)	Description
eap_phase2_id		EAP phase 2 credentials Syntax: TYPE [user[:password]] Possible values for Type: MSCHAPV2 (for PEAP), TTLS-MSCHAPV2 (for TTLS). TLS has no phase 2.
use_radius_server	yes	Use an external Radius server for authentication. If enabling this, don't use authentication above. Otherwise, it won't work...
source_addr	192.168.1.3	Define IP address to use as source for communication with radius server. Default: use address according to routing table
radius_ipaddr	192.168.1.3	The IP address of the access point (used as NAS-IP-Address). If not given, the system uses the IP address of the WLAN card.
radius_server	192.168.1.1:1812	IP address and port of the Radius server. If port is not given, the default port 1812 is used. Multiple Radius and Accounting servers can be configured by repeating these two statements. They are used if the first one does not reply.
radius_secret		The shared secret used for accessing the Radius server.
radius_accounting		IP address and port of the Accounting server. If port is not given, the default port 1813 is used.
radius_acct_secret		The shared secret used for accessing the Accounting server.
radius_retry		The interval (in seconds) to try and return to first radius server. If set, the system will try to return to the first server even if the current server still works. If not set, the system will try the given Radius servers consecutively and stays with a working server until it fails.
attribute		Specify additional Radius attributes that are sent to the radius server. Syntax: attribute = attr_id[:syntax:value] attr_id: Radius attribute type syntax: s = string, d = integer, x = octet string value: attribute value in format specified by syntax Examples: attribute = 6:d:2 → Service-Type = 2 (Framed) attribute = 61:d:15 → NAS-Port-Type = 15 (Ethernet)

4.3.42 [tpm]

The TPM can be used to securely store certificates, such that the private key never leaves the TPM. The TPM is initialized by default, with both owner password andSRK password set to abc123. Passwords can be changed with tpm_changeownerauth. To re-initialize the TPM, see /etc/tpm/HOWTO

The owner password is needed to take ownership of the TPM, and is not stored in the config file. The SRK password is needed to access some data when accessing stored private keys, so it must be given here.

Attribute	Value (Default)	Description
srk	abc123	SRK Password
pkcs11	abc123	Private Keys are stored in PKCS#11 containers, which is an addon to the TPM itself. The PKCS#11 container itself has two

Attribute	Value (Default)	Description
		additionalpasswords: - SO pin: security officer pin, the Master password. This pin is not stored in the config file. - user pin: this is needed to access the stored certificates.

4.3.43 [power]

The CPU can run on three different clock rates, which manifests itself in the power dissipation. The clock rate can be adjusted dynamically during runtime.

There are predefined governors that use fixed settings, or leave the setting to a user space program.

The governor performance always runs the CPU on the maximum rate, the governor powersave always on the minimum rate.

The governor ondemand changes between the available clock rates, based on the current load in the system. While the CPU is idle, it sets the mimimum rate, and increases as soon as the CPU has some work to do.

Finally, the governor userspace leaves the setting to some own applications.

The current clock rate can be read with this command:

```
cat /sys/devices/system/cpu/cpufreq/policy0/cpuinfo_cur_freq
```

To set the clock rate, use this command:

```
echo 996000 > /sys/devices/system/cpu/cpufreq/policy0/cpuinfo_cur_freq
```

The list with supported clock rates can be shown with

```
cat /sys/devices/system/cpu/cpufreq/policy0/scaling_available_frequencies
396000 792000 996000
```

Thus the CPU supports 396MHz, 792MHz and 996MHz.

An internal battery pack (BBU: Battery Backup Unit) is available as an option to the AnyRover. An additional service is running on the device that controls this battery. Several options allow to configure the behavior of the system if power fails.

Attribute	Value (Default)	Description
governor	performance	Sets the current governor which controls CPU clock rate. Possible governors: performance: CPU always runs on maximum rate powersave: CPU always runs on mimimum rate ondemand: CPU run on minimum and is increased on load userspace: clock rate is set through user space program. Such a program must be supplied. A second argument defines when the clock rate is set, either after power on, or after boot process has finished.

Attribute	Value (Default)	Description
		Syntax: governor = <type> [when] when takes the values preboot (default if not specified) or postboot.
bbu_start		Set to yes to start BBU monitoring daemon.
bbu_governor		Set the governor that ist used in battery mode. If left empty, the governor will not be changed when switching to battery mode. Default: powersave
bbu_voltage		Define the output voltage in [V] of the battery. This value must match the voltage of the power supply. The system will choose a voltage approx 10% below this value as the trigger voltage to switch to battery mode, and will provide this supply voltage when running on battery. Note: nothing will break if the voltages to not correspond, but the device may not work as expected. Possible values: 12...24 Default value: 12
bbu_regulation_voltage		Define the voltage the battery is charged to, in [mV]. The battery may be charged up to 4200mV, but charging to more than 4000mV, the lifetime of the battery is reduced significantly. Default and recommended value: 4000
bbu_disable_poe1 bbu_diasble_poe2		If set to yes, the PoE modules will be switched off when in battery mode. Default: yes

4.3.44 [bondix]

Enable the channel bonding software from Bondix to use various uplinks simultaneously. This function uses the saneclient application from bondix and needs a server with the saneserver application running to connect to. The saneclient creates a tunnel to the saneserver and uses all defined interfaces for the connection.

The connection is not encrypted. If encryption is wanted, use IPsec over the bondix-connection.

The bondix client contains a transparent proxy for TCP connections that can considerably speed up TCP sessions. If enabled, firewall rules will automatically be inserted directing suitable traffic to the proxy.

Attribute	Value (Default)	Description
start		Defines if this section is active or not
mode	client	Defines that this section is activated as client
logfile	var/log/ saneclient.log	Define the logfile directory. All entries of the saneclient.log are logged to the AnyRover messages logfile as well.
tunnelmode	bonding	Defines how multiple channels are used. Possible values are: bonding, failover, loadbalancer For bonding, the full license on the server is needed Default value: bonding
server		Defines the bondix-Server IP address.

Attribute	Value (Default)	Description
port	443	Defines the port for the connection to the server. Default value: 443
backup_server		If two servers are used for redundancy, define the IP address of the backup server here. This server is connected only if the primary server is not available.
backup_port	443	Port for the connection to the backup server. Default value: 443
use_encryption		If set to yes, the bondix clients uses internal encryption based on ChaCha20 algorithm.
auth_method	password	Defines authentication method for the bondix server. Possible values are: password, cert.
password		Defines the connection password. This password must match to the configured password on the server.
name		Define the name of the tunnel
cert		Path to the certificate file. The certificate has to be saved in PEM format
root		Path to the root certificate file.
key		Path to the private key.
interfaces		Add interfaces to the bondix tunnel as channels. The interface must correspond to the existing interfaces on the AnyRover. The channel names can be defined at will. Syntax: interfaces = interface1:channelname1, interface2:channelname2, … Example: interfaces = ppp0:Swisscom, ppp1:Sunrise, wlan0:WLAN
local_iface	bondix0	Set name of the local bondix tunnel interface
bonding_proxy	no	Activate option bonding proxy
proxy_iface		The bonding proxy receives TCP packets on this interface to submit to the server. This attribute must be defined.
proxy_host	0.0.0.0	With this attribute, an additional IP address can be given to receive packets on. This is optional.
proxy_port	18080	Define the port where the proxy listens for packets. Do not forget to open the port in the firewall section.
webgui_start		Choose wether to start the bondix webgui. For this add a local interface and port to use for the webgui Syntax: webgui_start = interface:port Example: eth0:80 To use the webgui, configure the local webserver and make sure, the port is open in the firewall section.
webgui_config		Possible values: yes, no
webgui_monitor		Possible values: yes, no
webgui_key		Define a password for the login to the webgui

4.3.45 [voltage]

Define actions to be executed when input voltage drops below a certain threshold. This is useful if the AnyRover runs on battery.

The AnyRover monitors its input voltage, and can execute an action if the voltage drops

below a certain value. This action may be to alert somebody, to power off, or anything else. There is even the possibility to perform another action if the voltage rises again above another threshold (unless the AnyRover has powered down…).

The input voltage is measured with a 10 bit analog to digital converter (ADC), and covers most of the input voltage range from 0..60V. The step size of the ADC is approx. 72mV, and is then rounded to the next 10mV.

Note: the AnyRover is a mobile router and not a precision voltage meter. Use the values with necessary caution.

Note: the AnyRover stops working if the input voltage drops below approx. 8.3V

Attribute	Value (Default)	Description
start		Defines if the monitoring should be enabled.
limit	11200	Voltage in mV below which the action is triggered.
delta	300	If after triggering the action, the voltage rises again above (limit+delta), the alarm is reset. Value is also given in mV and must be >=150mV.
count	5	Number of consecutive measurements that must be below the limit to trigger the action. It is not recommended to set this to 0 to prevent false alarms due to individual out of range measurements.
interval	30	Interval in seconds between consecutive input voltage measurements.
ignition		This parameter defines whether to check the ignition signal prior to triggering the action. Possible values: - ignore: do not check the ignition signal - on: trigger action only if ignition is on - off: trigger action only if ignition is off In case the ignition signal has the wrong value, only a log message is written, but no action triggered. If voltage stays low and ignition changes later on, the action will be triggered eventually. This parameter should be used if the AnyRover is supposed to shut down on low battery, because it will just reboot if ignition is on.
action1 action2		Actions to perform on low voltage. Action1 is executed first, followed by action2 after a 2s delay. Both may be left empty. The action may be one of these: - LOG: write a log message - POWEROFF: shut down the system - <script>: any script in /etc/scripts.d/
recover1 recover2		Actions to perform when the voltage rises again above the upper threshold. These only work if the AnyRover did not shut down on low voltage. Usage similar to actionX above.
adc_offset		This offset (in mV) will be added to all ADC conversions to voltage. If the pure ADC value (in steps) is queried, it will not be modified. Default value: 0

Attribute	Value (Default)	Description
adc_scale		For all ADC conversions to voltage, the value will be multiplied by this value (before adding the offset). If the pure ADC value (in steps) is queried, it will not be modfied. ADC[mV] = adc_scale * value + adc_offset Default value: 1

5 Support

After power on, the AnyRover automatically starts all services and is ready after a short time. There are different mechanisms to interact with the system to do maintenance and development of new features.

5.1 Lock files

With lock files, the system can be prevented from using certain resources. Lock files are in the directory /var/lock, and the only relevant fact about the lock file is its existence, the contents of the file is irrelevant. Lock files can be genareated using the command touch.

Lock file	Protected resource
/var/lock/sms	The SMS daemon no longer accesses the modem to read and send SMS.
/var/lock/ppp	The ppp daemon does not start new connections. Established connections are not terminated automatically.
/var/lock/ppp0	The ppp daemon creates this file when the ppp connection is up.

5.2 Helper programs

The AnyRover contains several helper programs that can be helpful when looking for errors or debugging new programs. They can be used as well in user defined scripts.

5.2.1 Modem status

The program *at* (which is not the Unix service with the same name) can be used to send AT commands to the modem and watch the answer. The program needs one or more AT commands as parameters. The AT commands are then passed in the given order to the modem, and all answers are put on stdout. The command at without arguments displays the usage.

5.2.2 Sending SMS

The AnyRover can – if a suitable SIM card is inserted – send SMS. The program *sms* is available for this. Incoming SMS cannot be viewed directly, but they are logged in /var/log/messages if the central service could not interpret them.

5.2.3 Central service

Many tasks are take care of by the central service. The service feeds the watchdog, monitors the GPI pins, processes GPS data, controls the modem and handles incoming SMS.

Using the program *cablynxctrl*, the central service can be controlled. With *help* all available options are listed

5.2.4 GPIO

With the program *gpio*, the GPI pins can be read and the GPO pins set. This does not apply to the pins controlled by the central service.

Usage: *gpio ionumber [val]*

If val is 1 or 0, the gpio is set to output and written. If val is ? the gpio output is read. Without val the gpio is set to input and read. *gpio* without arguments lists all options.

The inputs and outputs on the multi-purpose connector are controlled by the central service and cannot be set and read using gpio. In the config file, actions can be defined that are executed when input states change. The output can be set via such actions, or manually using the command *cablynxctrl*.

5.2.5 AD converter

The AnyRover contains several AD converters. The program *adc* reads the values of the converters.

5.2.6 Acceleration sensor

The AnyRover v3 no longer contains a dedicated acceleration sensor, since the GPS module integrates its own sensor.

5.2.7 Datcom

This tool logs in on a datcom gateway and transmits the current gps position. *datcom -h* list the description and options

5.2.8 PIC-Tool

The AnyRover contains a pic which stores some values that are configuration independent. With *pic -option val* values can be read and written. With *pic -h* all options are displayed.

5.2.9 Certificates and TPM

Certificates can be stored in a PKCS#11 store, where the TPM on the AnyRover protects the private key.

For TPM handling, a total of four passwords are defined, all of which are set to the default value of abc123.

The TPM itself has the owner key which is needed to take ownership of the TPM. This password is not needed during normal operation. Then there is the SRK password, which must be entered for every TPM operation. Therefore this password must be specified in the config file.

The PKCS#11 store also has two passwords: the Security Officer Pin is needed for setup and administration, but not normal usage. The user pin is needed for daily operations and

therefore is present in the config file.

To show the status of the PKCS#11 store, this command is used.

```
show tpm
```

The output should look like this (check the token flags line):

```
Available slots:
Slot 0 (0x0): AnyRoverTPM
 token label        : IBM PKCS#11 TPM Token
 token manufacturer : IBM Corp.
 token model        : TPM v1.1 Token
 token flags        : rng, login required, PIN initialized, token initialized, other
flags=0x40
 hardware version   : 1.0
 firmware version   : 1.0
 serial num         : 123
```

To show the contents of the PKCS#11 store, use this command:

```
# pkcs11 -l -O -p abc123
Using slot 0 with a present token (0x0)
Private Key Object; RSA
 label:      Key
 ID:         b96cb8c5887a6f29091cb417efa8715ad3edc377
 Usage:      decrypt, sign, unwrap
warning: PKCS11 function C_GetAttributeValue(ALWAYS_AUTHENTICATE) failed: rv =
CKR_ATTRIBUTE_TYPE_INVALID (0x12)
```

The warning about CKR_ATTRIBUTE_TYPE_INVALID can be ignored.

To use PKCS#11 certificates for IPsec or WLAN/802.1X the ID in the PKCS#11 store must equal the fingerprint of the key, which is automatically ensured when the cetificate is obtained with SCEP.

It is possible to import file based keys into the PKCS#11 store (only exporting is not possible). The key must be present in DER form. Importing a PEM key works like this:

```
# openssl rsa -in key.pem -pubout | openssl asn1parse -strparse 19 -out id.txt
# ID=$(openssl dgst -c -sha1 id.txt | awk -F= '{gsub(/[: ]/, "", $NF); print $NF;}')
# openssl rsa -in key.pem -outform der -out key.der
# pkcs11 -l -y privkey -w key.der --id ${ID} -a <KEYNAME> -p abc123
```

After the import it makes sense to delete the private key file.

5.3 Log files

The system writes important messages to the system log file. The log files are cleared regularly to prevent fill-up of the flash (hard disk).

The main log file is /var/log/messages, new messages are appended at the end. The command

```
tail /var/log/messages
```

can be used to show the last lines of the log files. The switch „-f“ causes the program to keep reading the files, so any new messages are shown immediately. When using „-F“, tail even stays on the file after a log file rotation has taken place.

6 Sample configurations

This section presents and explains some sample configurations for different situations. In the samples, only the lines relevant for the actual situation are printed, the other lines of the configuration files are assumed to be the default configuration.

Comments in the config file are omitted here.

In the examples, the IP addresses 172.16.X.Y – 172.24.X.Y are assumed to be public, while the addresses 10.X.Y.Z and 192.168.X.Y are assumed to be private.

Values in capital letters and angle brackets (e.g. <YOUR_KEY>) must be replaced with the correct values.

6.1 Permanent IPsec tunnel to the network

The AnyRover must maintain a permanent connection through a 3G link to the corporate network. Authentication is with a pre shared key, identification works using the host name.

A SIM card is used that obtains a local IP address with the provider and thus is behind a NAT gateway. The IPsec server has the public IP address 172.16.1.1, the internal corporate network is 10.0.0.0/8.

All other internet accesses are routed directly through the 3G link.

```
[system]
hostname = client1.example.org
[firewall]
accept = ppp0,udp,500
accept = ppp0,udp,4500
accept_fw = eth0,,,
[chat_script]
apn = <YOUR.PROVIDER.APN>
[ipsec]
start = yes
remote = 172.16.1.1
local_net = eth0
remote_net = 10.0.0.0/8
natt = yes
my_identifier = fqdn, client1.example.org
peers_identifier = fqdn, server.example.org
psk = <YOUR_PRE_SHARED_KEY>
```

6.2 IPsec tunnel on request

The 3G link and the IPsec tunnel are only built upon request. If there is no traffic in the tunnel for 5 minutes, the connection is terminated. This examples relies on the configuration „Permanent IPsec tunnel to the network“.

```
[ppp]
option = demand
option = nopersist
option = idle 300
[ipsec]
dpd = 0
```

6.3 IPsec server with multiple clients

An IPsec server has to accept connections from multiple clients. Authentication is established with certificates. The server has the public IP address 172.16.1.1/24 on VLAN1, the clients connect on the 3G network and are natted. The client subnets are in the range 192.168.X.0/24, the server subnet on VLAN2 is 10.0.0.0/8.

Server config:

```
[system]
ipaddr =
[switch]
start_vlan = yes
vlan1 = 1
ipaddr1 = 172.16.1.1/24
vlan2 = 2,3,4
ipaddr2 = 10.0.0.1/8
[firewall]
accept = vlan1,esp,
accept = vlan1,udp,500
accept = vlan1,udp,4500
[ipsec]
start = yes
setup = add
remote = any
local = eth0
local_net = 10.0.0.0/8
remote_net = any
remote_range = 192.168.0.0/16
natt = yes
my_identifier = asn1dn, C=ch, ST=zh, L=zrh, O=AW, OU=AnyRover, CN=Server, E=em@i.l
peers_identifier = asn1dn, C=ch, ST=zh, L=zrh, O=AW, OU=AnyRover, CN=*, E=em@i.l
auth_method = cert
[certificate]
name = ipsec-cert
type = pem
-----BEGIN CERTIFICATE-----
MII...
-----END CERTIFICATE-----
```

```
[certificate]
name = ipsec-root
type = pem
-----BEGIN CERTIFICATE-----
MII...
[certificate]
name = ipsec-key
type = pem
-----BEGIN RSA PRIVATE KEY-----
MII...
-----END RSA PRIVATE KEY-----
```

Clients:

```
[system]
ipaddr = 192.168.X.1/24
[firewall]
accept = ppp0,esp,
accept = ppp0,udp,500
accept = ppp0,udp,4500
[ipsec]
remote = 172.16.1.1
local_net = eth0
remote_net = 10.0.0.0/8
natt = yes
my_identifier = asn1dn, C=ch, ST=zh, L=zrh, O=AW, OU=AnyRover, CN=ClientX, E=em@i.l
peers_identifier = asn1dn, C=ch, ST=zh, L=zrh, O=AW, OU=AnyRover, CN=Server, E=em@i.l
auth_method = cert
[certificate]
...
```

6.4 2 local subnets with NAT

The AnyRover has 2 local subnets A (Ports 1 and 2, 192.168.1.0/24) and B (Ports 3 and 4, 10.1.1.0/24). Access from A to B is possible, traffic is natted. Access from B to A is prohibited.

```
[system]
ipaddr =
[switch]
start_vlan = yes
vlan1 = 1,2
ipaddr1 = 192.168.1.1/24
vlan2 = 3,4
ipaddr2 = 10.1.1.0/24
[firewall]
nat = vlan2
accept_fw = vlan1,,vlan2
```

6.5 Wireless client

The AnyRover connects to a wireless access point and shares the connection with local

clients. The access points runs WPA2, PEAP, and TKIP, the SSID is broadcast.

```
[firewall]
nat = wlan0
accept_fw = eth0,,wlan0
[wlan]
start = yes
country = <YOUR_2_LETTER_COUNTRY_CODE>
ssid = <YOUR_SSID>
identity = <YOUR_USERNAME>
password = <YOUR_PASSWORD>
```

If the connection is not successful, it might be necessary to set

```
eapol_version = 1
```

6.6 Roaming between WLAN and 3G

The AnyRover connects to a wireless network. Whenever WLAN is not available, it switches to 3G. As long as WLAN is available, the 3G link is not active.

The AnyRover verifies whether WLAN is still available by trying to renew the DHCP lease every 20 seconds. If the DHCP server can be configured with a lease time of only a short time, the sections [daemon] and [script] can be omitted.

```
[ppp]
defaultmetric = 20
option = demand
option = nopersist
option = idle 30
option = holdoff 15
[daeamon]
start = /etc/scripts.d/local/wlan_roaming.sh
[script]
name = WLAN Roaming
file = /etc/scripts.d/local/wlan_roaming.sh
mode = 755
#!/bin/sh
IFACE=wlan0
while true; do
        sleep 20
        if [ -r /var/run/dhcpcd-${IFACE}.pid ]; then
                dhcpcd -n ${IFACE}
        fi
done
[wlan]
start = yes
ipaddr = dhcp default timeout:15
```

6.7 Wireless access point with DHCP server

The AnyRover runs as wireless access point, using WPA2, PEAP, and CCMP. Clients obtain

an IP address using DHCP. Access to the AnyRover on the wireless interface is prohibited.

The certificates can be generated using the script /home/config/bin/cert.

```
[dhcp]
name = wlan0
start = yes
dhcpd_start = 192.168.1.11
dhcpd_end = 192.168.1.254
[firewall]
accept = wlan0,udp,67
accept = wlan0,udp,68
[wlan]
start = yes
mode = ap
country = <YOUR_2_LETTER_COUNTRY_CODE>
ssid = <YOUR_SSID>
ipaddr 192.168.1.1/24
pairwise = CCMP
channel = 1
authentication = eap_server
[authentication]
name = eap_server
start = yes
standalone = no
eap_phase1_id = PEAP
eap_phase2_id = MSCHAPV2 <USERNAME1>:<PASSWORD1>
eap_phase2_id = MSCHAPV2 <USERNAME2>:<PASSWORD2>
[certificate]
name = eap_ca_cert
type = pem
-----BEGIN CERTIFICATE-----
MIICWwIBAAKB...
-----END CERTIFICATE-----
[certificate]
name = eap_server_cert
type = pem
-----BEGIN CERTIFICATE-----
MIICWwIBAAKB...
-----END CERTIFICATE-----
[certificate]
name = eap_server_key
type = pem
-----BEGIN CERTIFICATE-----
MIICWwIBAAKB...
-----END CERTIFICATE-----
```

6.8 Multiple client connections over IPsec using PSK

Multiple identical clients connect over IPsec to the central server. Each client has its own small local network (192.168.2.X/30), which is reachable through the tunnel. Authentication is done using pre-shared keys, identification is by hostname.

The clients connect over the 3G link and are located behind the NAT gateway of the

provider. The tunnel must thus be initiated by the client.

The server has the public IP address 172.16.1.1 (vlan1) and the internal net 192.168.1.0/24 on vlan2.

Client configuration:

```
[system]
ipaddr = 192.168.2.1/30
[ipsec]
start = yes
remote = 172.16.1.1
local_net = eth0
remote_net = 192.168.1.0/24
tunnel = :/
natt = yes
my_identifier = fqdn, client1.example.org
peers_identifier = fqdn, router.example.org
psk = <MY_SECRET_PRESHARED_KEY>
ph1_hash_alg = sha1
ph2_hash_alg = sha1
```

AnyRover server configuration:

```
[system]
ipaddr =
[switch]
start_vlan = yes
vlan1 = 1
ipaddr1 = 172.16.1.1/24
vlan2 = 2,3,4
ipaddr2 = 192.168.1.1/24
[ipsec]
start = yes
setup = route
remote = any
local = vlan1
local_net = vlan2
remote_net = any
remote_range = 192.168.2.0/24
tunnel = :/
natt = yes
my_identifier = fqdn, router.example.org
## comment out all peers_identifier lines
psk = <MY_SECRET_PRESHARED_KEY>
ph1_hash_alg = sha1
ph2_hash_alg = sha1
```

Server configuration for a Cisco router:

```
hostname router
ip domain name example.org
crypto isakmp policy 1
 encr aes 256
 authentication pre-share
 group 2
crypto isakmp key <MY_SECRET_PRESHARED_KEY> address 0.0.0.0 0.0.0.0
```

```
crypto isakmp identity hostname
crypto ipsec transform-set ANYROVER esp-aes 256 esp-sha-hmac
crypto dynamic-map ANYDYNMAP 10
 set transform-set ANYROVER
 set pfs group2
 match address 110
crypto map ANYMAP 10 ipsec-isakmp dynamic ANYDYNMAP
interface FastEthernet0/0
 ip address 172.16.1.1 255.2552.255.0
 crypto map ANYMAP
interface FastEthernet0/1
 ip address 192.168.1.1 255.255.255.0
access-list 110 permit ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255
```

6.9 Sending files over E-mail

The AnyRover has to regularly send log files via e-mail. This script is started by cron job every thursday at 1:04am.

```
[crontab]
entry = 4 1 * * 4 /etc/scripts.d/logmailer/mkmail.sh
[script]
name = log-mailer
file = /etc/scripts.d/logmailer/mkmail.sh
mode = 755
#!/bin/sh
for i in /DIRECTORY/CONTAINNG/LOGFILES/*;do
makemime -a "Content-Disposition: inline" -a "Subject: MAIL SUBJECT" -a "Date: `date
-R`" -c application/octet-stream $i |
awk -F\" '/boundary=/ && !b { b = $2; }
        /^$/ && !a++ {
                print "\n--" b;
                while ( getline l < "/etc/scripts.d/logmailer/mail.txt" ) print l;
        } 1' |
sendmail -t -f SENDER@AD.DR -S SMTP_SERVER -auUSERNAME -apPASSWORD RECEIVER@AD.DR
rm $i
done
[script]
name = mail-content
file = /etc/scripts.d/logmailer/mail.txt
mode = 644
Content-Type: text/plain
Content-Transfer-Encoding: 7bit
Please find attached the newest log file.
```

6.10 IPsec server for Cisco VPN clients

The AnyRover is configured as a server for Cisco VPN clients. Authentication must be done using certificates, using pre-shared keys is not possible (reason: the Cisco VPN client uses aggressive mode with pre-shared keys, which is not supported by the AnyRover for security reasons).

Currently, only one connection at a time is possible. Extension for multiple concurrent connections will be implemented later.

This method also works for connections from an iPhone. In this case, the server on the iPhone must be given as a hostname, and the certificate on the AnyRover must have this hostname in the CN field.

To import the certificates on the iPhone, the CA certificate has to be given as a .pem file, and the client certificate and client key have to be packed in a .p12 file. The .p12 file can be generated with this command:

```
openssl pkcs12 -in client-cert.pem -inkey client-key.pem -out client.p12 -export
```

The command asks for an export password, which has to be set, because the iPhone cannot handle .p12 files without password protection.

```
[system]
nameserver = 172.17.100.200
nameserver = 172.21.21.3
[ipsec]
start = yes
setup = add
remote = any
local = eth0
local_net = 0.0.0.0/0
remote_net  = any
remote_address = 192.168.2.1
tunnel = /
natt = yes
tries = 1
my_identifier = asn1dn,
peers_identifier = asn1dn,
auth_method = xauth-cert
xauth = server
xauth_id = iphone:iphone
ph1_lifetime = time 24 hour
ph2_lifetime = time 1 hour
dh_group  = 5
pfs_group =
```

Finally, the certificates must be given in the respective [certificate] sections.

6.11 Setting GPO

The general purpose output (GPO) is must be switched on when a certain file is created, and switched off if the file is deleted.

The example uses the file /var/lock/ppp0 which only exists if the 3G connection is online.

```
[daemons]
start = /etc/scripts.d/output.sh
[script]
```

```
name = output
file = /etc/scripts.d/output.sh
mode = 755
#!/bin/sh
file=/var/lock/ppp0
while true; do
        test -r ${file}
        j=$?
        if [ "${j}" != "${i}" ]; then
                i=${j}
                test ${i} -eq 0 && cmd=out_on || cmd=out_off
                echo ${cmd} | cablynxctrl
        fi
        inotifyd : `dirname ${file}`:nd &>/dev/null
done
```

6.12 SCEP certificate management

The AnyRover can obtain certificates through SCEP (Simple Certificate Enrolment Protocol). Certificates are then stored in /etc/certs/, where they can be referenced from [certificate] sections.

The private key can be stored in a file or in a PKCS#11 store, where it is protected by the onboard TPM. Basically, when stored in a PKCS#11, the key is also written to a file (in /etc/opencryptoki/), but encrypted by the TPM such that it can only be decrypted inside the TPM and never exported. Thus this key can only be used on this very AnyRover, or TPM to be precise.

6.12.1 File based certificates

Obtain certificates, store in files and use for IPsec.

```
[scep]
name = ipsec-cert
start = yes
directory = /etc/certs/ipsec
days = 14
key_size = 2048
key_type = rsa
signature = sha256
server = http://<SERVER-IP>/certsrv/mscep/mscep.dll
encryption = des
ca-file = ca-cert.pem
password = <SCEP PASSWORD>
old_cert_auth = no
CA-DN = C=CH, ST=ZH, L=Zuerich, O=AnyWeb, OU=IT, CN=anyca
cert-file = cert.pem
key-file = key.pem
[ipsec]
local_auth = pubkey
root = ipsec-root
cert = ipsec-cert
key = ipsec-key
[certificate]
```

```
name = ipsec-root
type = file
file = /etc/certs/ipsec/ca.pem
[certificate]
name = ipsec-cert
type = file
file = /etc/certs/ipsec/cert.pem
[certificate]
name = ipsec-key
type = file
file = /etc/certs/ipsec/key.pem
```

6.12.2 TPM based certificates

There are only a few changes vs file based certificates. There will be a private key file as well, but the file only contains the key ID as a hex string

01:23:45:67:89:ab:cd:ef:01:23:45:67:89:ab:cd:ef:01:23:45:67

```
[scep]
key_type = pkcs11
[ipsec]
#key = ipsec-key
pkcs11 = cert:ipsec-key, abc123
[certificate]
name = ipsec-key
type = file
file = /etc/certs/ipsec/key.der
```

6.13 IEEE 802.1X port security

The AnyRover can protect its switch ports using 802.1X, such that clients have to authenticate first before using the switch port.

6.13.1 Authenticator

Every LAN port using 802.1X should be placed in its own VLAN, because for every client connected to a switch port, the list of authenticated MAC addresses for this interface is cleared.

To use multiple 802.1X protected ports in the same subnet, every port must be in different VLANs and then combined with a bridge.

The example shows port 1 using EAP-TLS, i.e. certificate based authentication. In multi-host mode, one successful authentication allows all possible clients. The idea is to connect an 802.1x capable switch to this port that protects its own ports using 802.1X as well. Then only the switch has to authenticate, and all clients may use the port as well after authenticating with the switch.

Port 2 is intended for clients using EAP-PEAP authentication, i.e. username and password. The MAC authentication bypass on port 3 allows a non-802.1x client (e.g. a printer) to use this port. Authentication data from 802.1X clients is sent to an external Radius server.

The autenticator software cannot use PKCS#11 keys yet.

```
[system]
ipaddr =
[switch]
start_vlan = yes
vlan1 = 1
ipaddr1 = - 8021x
vlan2 = 2
ipaddr2 = - 8021x
vlan3 = 3
ipaddr3 = - 8021x
vlan4 = 4
ipaddr4 = 192.168.2.1/24
[bridge]
name = br0
start = yes
ipaddr = 192.168.1.1/24
iface = vlan1 vlan2
[8021x]
name = vlan1
mode = authenticator
route = yes
port_mode = multi-host
standalone = yes
eap_phase1_id = TLS
root = 8021x-root
cert = 8021x-cert
key = 8021x-key
[8021x]
name = vlan2
mode = authenticator
route = yes
port_mode = single-host
standalone = yes
eap_phase1_id = PEAP AnyRover
eap_phase2_id = MSCHAPV2 user:password
[8021x]
name = vlan3
mode = authenticator
route = yes
port_mode = single-host
mab = 00:11:22:33:44:55
source_addr = 192.168.2.1
radius_ipaddr = 192.168.2.1
radius_server = 172.16.1.1:1812
radius_secret = verysecret
[certificate]
name = 8021x-root
type = file
file = /etc/certs/8021x/ca.crt
[certificate]
name = 8021x-cert
type = file
file = /etc/certs/8021x/authenticator.crt
```

```
[certificate]
name = 8021x-key
type = file
file = /etc/certs/8021x/authenticator.key
```

6.13.2 Supplicant

The supplicant on vlan3 uses a certificate to do EAP-TLS, while the client on vlan4 uses EAP-PEAP with username and password. The supplicant code supports PKCS#11 based certificates; configuration is analogue to IPsec configuration.

```
[system]
ipaddr =
[switch]
start = yes
vlan3 = 3
ipaddr3 = dhcp default nolinklocal metric:15 8021x
vlan4 = 4
ipaddr4 = dhcp default nolinklocal metric:20 8021x
[8021x]
name = vlan3
mode = supplicant
key_management = EAP
eap = TLS
root = 8021x-root
cert = 8021x-cert
key = 8021x-key
[8021x]
name = vlan4
mode = supplicant
key_management = EAP
anonymous_identity = AnyRover
identity = username
password = password
phase2 = auth=MSCHAPV2
[certificate]
name = 8021x-root
type = file
file = /etc/certs/8021x/ca.pem
[certificate]
name = 8021x-cert
type = file
file = /etc/certs/8021x/cert.pem
[certificate]
name = 8021x-key
type = file
file = /etc/certs/8021x/key.pem
```

Using PKCS#11 keys, the configuration changes to this:

```
[8021x]
#key = 8021x-key
pkcs11 = cert:8021x-key, abc123
[certificate]
name = 8021x-key
type = file
file = /etc/certs/8021x/key.der
```

A Contact

A.1 Responsible persons

A.1.1 Commercial

Wim van Moorsel, AnyWeb AG	<wvm@anyweb.ch>	+41 58 219 11 03

A.1.2 Technical project lead

Marco Wirz, AnyWeb AG	<mwi@anyweb.ch>	+41 58 219 11 26

A.1.3 Support and maintenance

Hardware: Christian Bürki, Cabtronix AG	<buerki@cabtronix.ch>	+41 44 804 74 36
Software: Marco Wirz, AnyWeb AG	<mwi@anyweb.ch>	+41 58 219 11 26

B Default configuration file

```
###############################################################################
### Global configuration file for AnyRover
###############################################################################
###      Index
### ========================
###  (0) Variables
###  (1) Addressing
###  (2) Switch
###  (3) System Time
###  (4) Watchdog
###  (5) Crontab
###  (6) GPIO
###  (7) GPS
###  (8) SMS
###  (9) Modem
### (10) USB ports
### (11) DHCP (server and relay)
### (12) FTP
### (13) TFTP
### (14) Firewall
### (15) DynDNS
### (16) PPP
###      (16a) chat scripts
###      (16b) WAN
### (17) IPsec
###      (17a) IPsec Certificates
### (18) OpenVPN
###      (18a) Custom client config files
###      (18b) OpenVPN Certificates
### (19) Tunnel
### (20) Bridge
### (21) Message of the day
### (22) User daemons
### (23) Scripts
### (24) Web server
### (25) WLAN
###      (25a) WLAN client Certificates
### (26) Authentication (EAP/Radius server)
###      (26a) Server Certificates
### (27) OSPF
### (28) SNMP
### (29) DNS
### (30) Serports
### (31) OpenConnect VPN
### (32) Mobile IP
### (33) SCEP
### (34) Pelix
### (36) Login
### (37) IEEE 802.1X Port security
###      (37a) 802.1X Certificates
### (38) TPM - Trusted Platform Module
### (39) Power management
### (40) Bondix - Channel bonding
### (41) Voltage monitoring

###############################################################################
### all entries are of the form
### attribute = value
### default values as indicated in the comments are applied if
### the attribute is not set in the config file

[variables]
###############################################################################
###  (0) Variables
###############################################################################

## Define variables to be used throughout the configuration file.
## Variables are available on all lines following the definition.
## Definition in the [variables] section: variable = value
## Usage everywhere in the config file: {{variable}}
## Values can not contain '#', since the rest of the line is ignored.
## Variables may be nested:
## num = 1
## config1 = file_a.conf
## config2 = file_b.conf
## file = {{config{{num}}}}  # evaluates to file_a.conf
## To add runtime information to config values, use {{run:VAR}} in
## the [variables] section.
## The contents of variable VAR is then executed, and the first line
## of the output used as value for the variable.
## NOTE: do not use long running commands here, as the config file
## is parsed quite often.
## Example:
## showversion = show version
## sysdescr = AnyRover v3, {{run:showversion}}
## [snmp]
## sysdescr = {{sysdescr}}

[system]
###############################################################################
###  (1) Addressing
###############################################################################

## Network configuration
## The value can be an IP address/prefix or addr/mask pair, or dhcp
## When using a static address, the parameter mtu:VALUE can be added
## to set the MTU of the interface.
## When using dhcp, these parameters are valid:
##    default   set default route on this interface, as advertised
##              by the dhcp server
##    metric:M  set metric of default route to M (default: 0)
##    timeout:T set timeout to T (default: 15s)
```

```
##    dns       ask the dhcp server for DNS server addresses and replace
##              the currently configured servers with these.
##    hostname  ask the dhcp server for and set hostname.
##              This only works if the current hostname is localhost.
##    sendhostname[:hostn]  Submit our current hostname or
##              the value 'hostn' to the DHCP server.
##    nolinklocal  Do not use link local addresses (169.254.X.Y)
##    noarp     Do not check address using ARP. Implies nolinklocal.
##    vendor:V  Set Vendor Class Identifier string to V. Probably only
##              needed for DSL links.
##    clientid:I   Set Client ID string to I.
##    require:A,B  Required parameters A,B to be supplied by DHCP
##              server. Offer will be ignored if they are missing.
##              Possible options: bootfile_name, tftp_server_name,
##                      ntp_servers, ...
## To set an interface without an IP address, use - for the address.
## To enable 8021x on this interface (both authenticator and supplicant),
## the parameter 8021x can be added. All 802.1X parameters are then
## configured in a separate [8021x] section.
#ipaddr = 192.168.1.3/24 mtu:1496
#ipaddr = 192.168.1.3 255.255.255.0
#ipaddr = dhcp
ipaddr = 192.168.1.3/24

## Loopback addresses
## The loopback interface has the address 127.0.0.1 by default.
## With this option, additional addresses can be added to the
## loopback interface.
#loopback = 1.1.1.1/32

## Routing
## gateway:  ip address of default gw.
##           When configuring some interfaces using dhcp, they can
##           be told to set the default route, so this option is
##           only used to set the default route through a statically
##           configured interface.
## The parameter metric:M can be appended to specify the routing metric
#gateway = 192.168.1.1
#gateway = 192.168.1.1 metric:10
gateway =

## Policy based routing
## policy = SELECTOR ACTION
## When SELECTOR matches, the routing decision is based on ACTION.
## SELECTOR can be one of
## - from PREFIX
## - to PREFIX
## - tos TOS
## - dev DEVICE
## ACTION can be one of:
## - table NUM
## - prohibit | reject | unreachable
## The table number can be used again in static_route below, the
## number may be in the range from 1 to 32765.
## (0, 32766, and 32767 are already defined by the system)
## HINT: IPsec uses tables 200 and 220, don't use them.
## For dynamic links with non-zero metric, a source route is set.
## They use tables 100-199: specifically (metric mod 100)+100
#policy = from 1.1.1.1 table 300

## Static routes
## Static routes are inserted into the routing table after all
## interfaces, VLANs and bridges are configured.
## Only IPsec and OpenVPN are started later so the OpenVPN interface
## is not available here.
## static_route = [target][/prefix] [netmask] gateway [metric:M]
##                [table:T] [src:S]
## if both prefix and netmask are omitted, the default class-based
## prefix is chosen; if both are present, the prefix is ignored.
## If no target is given, the default route is set.
## gateway can either be an IP address or the name of an interface.
## A gateway IP address must already be reachable with the existing
## routing table, an interface must exist.
## table:T assings the route to the routing table T as created with
## the policy option above.
## With the src:S parameter, the source address can be set for this
route.
## The source address must be set on one interface of the system.
#static_route = 192.168.92.0/24 192.168.1.3
#static_route = 192.168.93.0/24 ppp0 table:300
#static_route = 192.168.94.0 255.255.255.0 vlan1 metric:10
#static_route = 192.168.93.0/24 ppp0 table:300 src:192.168.1.3

## Proxy ARP
## List of interfaces (space separated) that have proxy arp enabled.
#proxy_arp = eth0 vlan1
proxy_arp =

## hostname for the system. (default: localhost)
## The hostname can also be set by the DHCP client on any interface.
## If this is desired, it has to be set here to localhost (and configured
## on the respective dhcp client).
hostname = anyrover

## name servers to use for DNS lookups. Up to three name servers can
## be specified, additional entries are ignored.
## These name server entries can be overridden by the dhcp entries
## on the respective interfaces, which has to be configured on the
## respective dhcp client.
## The first two name servers are also used in IPsec to hand out to
## clients when they make a mode config request. One example where
## this is necessary is when using an iPhone as VPN client.
nameserver = 9.9.9.9
nameserver = 8.8.8.8
nameserver = 1.1.1.1
```

```
## Search domain.
## When doing name lookups, if the name is not found, this suffix
## is appended and name is tried again.
#domain = anyweb.ch

## WINS server to hand out to IPsec clients with mode config requests.
## Up to two wins servers can be specified.
## WINS server (as well as name server) can only be set globally for
## all IPsec connections, that is why these parameter is not in the
## ipsec section.
#winsserver = 192.168.84.13

##---------------------------------------------------------------------
## Logging
##---------------------------------------------------------------------

## If log_server is specified, syslog messages are sent to this address
## (default: none)
#log_server  = 192.168.1.1

## Log level. All log-messages with log level less than this value are
## logged to /var/log/messages. By default, everything is logged.
#log_level = 6

## Log file. By default, the system log file is /var/log/messages.
## This file is on the RAM disk, so upon reboot, everything is lost.
## Using this parameter, another file can be specified as the log file.
## If only a file name is given, the file will be placed in /var/log/
## If the path does not exist, it will be created.
#log_file = /opt/log/messages

## Log file rotation. The log file is automatically rotated whenever it
## reaches a certain size (default: 200KB), and older rotated files are
## deleted (default: one old file is kept).
## These two parameters define the size in KB the log file must have
## to be rotated (default: 200KB), and how many old log files are kept
## (default: 1)
#log_rotate_size = 200
#log_rotate_files = 1

##---------------------------------------------------------------------
## System updates
##---------------------------------------------------------------------

## tftp_server: default entry for system updates as user config
tftp_server = 192.168.1.1

##---------------------------------------------------------------------
## Mount partitions
##---------------------------------------------------------------------

## Mount additional partitions.
## Syntax: device, filesystem, mountpoint [, option [, option]]
## The mount point will be created if not already present.
## Options as known from /etc/fstab. Option noatime is set by default.
## More options: rw (default), ro, [no]exec, ...
#partition = /dev/mtdblock4, jffs2, /media/log, noexec

[switch]
##############################################################################
###  (2) Switch
##############################################################################

## enable the switch? (default: yes)
## If set to no, the external ethernet ports will not work.
start = yes

## Power over Ethernet (PoE)
## Ports 1 and 3 support PoE
## The PoE modules can be enabled
## Note: For PoE in BBU mode, see bbu_disable_poeX in [power] section.
poe1 = no
poe2 = no

## Switch ports
## Each port can be set to auto-negotiation, or fixed on
## 10/100M half/full duplex.
## ports = port1,port2,port3,port4
## portX = 0,1,2,3,4
##   0: 10M, half duplex
##   1: 10M, full duplex
##   2: 100M, half duplex
##   3: 100M, full duplex
##   4: auto negotiation
## omitted values are set to 4 (auto negotiation).
## Example: ports = 1,,2,4
## Here, port1 is set to 10Mfull, port2 to auto, port3 to 100Mhalf,
## and port4 to auto
ports =

## The switch ports can individually be disabled. The parameter takes
## a comma separated list of port numbers to be disabled.
## All ports not mentioned here are enabled.
## This only works if start_vlan is set to yes.
#port_disable = 3, 4

## VLANs: it is possible to define up to 4 VLANs. Every switch port
## can be in one of the VLAN, or left as it is.
## Port 5 (to the processor) is in all VLANs.
## vlanX is a list of switch ports to participate in this VLAN
## ipaddrX defines the IP address of VLAN X. The syntax is identical
## to the parameter ipaddr in the system section.
## When setting ipaddrX=noif, then the corresponding VLAN will be
## configured on the switch, but no local interface will be created.
start_vlan = no
#vlan1 = 1,2,3
```

```
#ipaddr1 = 192.168.1.3 255.255.255.0
#ipaddr1 = 172.24.34.1 255.255.0.0
#vlan2 = 4
#ipaddr2 = dhcp default nolinklocal metric:20
#ipaddr2 = 192.168.3.3 255.255.255.0

## VLAN Trunking:
## It is possible to configure VLAN trunks on external switch ports.
## Example: A trunk on port 3 is defined with
## trunk = 3
## If one of the 4 internal vlans above contains the trunk port in its
## port list, the packets from the other ports will be sent over the
## trunk with the respective tag. If the list does not contain the port,
## it will never be sent over the trunk.
#trunk = 3

## Additional vlan interfaces are configured as usual:
## ipaddrX = ...
## X must have the value 5 or larger. All such interfaces can only
## communicate through trunk ports, and packets are properly tagged.
## At most 16 different interfaces can be configured (including the
## internal ones ipaddr1..4).
#ipaddr8 = 192.168.8.1/24
#ipaddr9 = dhcp default metric:90 nolinklocal dns

## Rebind dhcp lease when a network cable is plugged in a switch port
## Only the dhcp client on the corresponding switch port is rebound.
dhcp_rebind = yes

## internal signals, don't change
reset = 50
ps1 = 123
port_poe1 = 53
port_poe2 = 54

[time]
###########################################################################
###  (3) System time
###########################################################################

## Set timezone info
## timezone = zone
## zone is the name of a file in /usr/share/zoneinfo/
## Some possible values: CET, GMT (includes Daylight saving time),
## UTC (no daylight saving time), EET, EST, ...
## Alternatively, specifiy location, e.g. Europe/Zurich,
## America/Vancouver, Pacific/Auckland, ...
## e.g. timezone = CET
timezone = CET

## start ntp daemon? [yes|no] (default: yes)
## This is needed both to synchronize to another time source
## and to play ntp server ourselves.
## If we are ntp server, enable port 123 in the firewall section
start = yes

## What to use as source for the system time (default: gps)
## gps: GPS receiver
## ntp: ntp server
## none: do not try to synchronize system time (but still let
## others sync their clock after us).
time_source = gps

## ntp_server is only used if time_source = ntp
ntp_server  = pool.ntp.org

## ntp_flags allows to restrict ntp service. List of flags,
## separated by comma or white space.
## The flags are appended to a line of the form
## restrict default [ntp_flags]
## Possible flags (see ntp.conf man page):
## kod, limited, lowpriotrap, nomodify, noquery, nopeer, noserve,
## notrap, notrust, ntpport, version
## Recommended value: ntp_flags = kod nomodify notrap nopeer noquery
ntp_flags = kod nomodify notrap nopeer noquery

## Allow local access to ntp status info.
## Set this to yes to make "show ntp" work.
localaccess = yes

## Additional options for ntp client.
## Syntax:
#ntp_option = <whatever>

## jump_clock: ntp daemon normally adjusts the clock once and in
## one step after startup, but only if the difference is less
## than 1000s. Otherwise, it exits with an error message.
## If this parameter is set, then it jumps the clock once for
## whatever it needs to get current time, even if the step is
## larger than 1000s.
jump_clock = yes

## check system clock with modem time. When there is a 3G connection,
## the system clock is checked automatically from the modem time every
## time, the connections is established.
## default value: modemclock = yes
modemclock = yes

## Display time sync on external LEDs.
## LEDs will blink toghether if time was correct.
## LEDs will show animation from left to right if time was synched.
## default (if not given): yes
syncled = yes

[watchdog]
###########################################################################
```

```
###  (4) Watchdog
#############################################################################

## start (and feed) the watchdog (default: yes)
start      = yes

## feed the watchdog every n seconds (default: 15)
interval   = 15

## internal signals, don't change
gpio_on    = 122
gpio_feed  = 124
cmd_on     = 1

[crontab]
#############################################################################
###  (5) Crontab
#############################################################################

## start cron daemon
## Note: the cron daemon can be started by other functions.
## But the actions described in this section are only activated
## if start is set to yes here.
start = no

## Cron log level: default value = 8
## level = 8: log every call of command
## level = 9: log only warnings and errors
#loglevel = 8

## crontab entries
## the crontab is parsed every minute
## entry = min hour dayofmonth month dayofweek command [parameter]
## entry = [0-59] [0-23] [1-31] [0-12] [0-7] command [parameter]
## ranges can be given using /X, i.e. */2 means every two
## a '*' means any
## examples:
## 5 0 * * * cmd        runs 5 min past midnight every day
## 15 14 1 * * cmd      runs at 14:15 on the first of every month
## 0 22 * * 1-5 cmd     runs every weekday at 22:00
## 23 0-23/2 * * * cmd  runs daily at 0:23, 2:23, 4:23, ... , 22:23
## 5 4 * * sun cmd      runs every sunday at 5:04
## 0 15 1 * 2 cmd       runs at 15:00 on the first of every month and
##                           on tuesday
#entry = 0 0 * * * echo "It's midnight, beware of ghosts!"

[gpio]
#############################################################################
###  (6) GPIOs
#############################################################################

## defines actions to perform when the GPIO signals change or
## the reset or mode button is pressed
##
## syntax:
## {button|mode} = time, action
## {gpio} = ([time,]action1), ([time,]action2)
##      {gpio}: [ignition|inputX]
##      where X is the number of the input port
##      The CabLynx Eco / AnyRover has 3 input ports
## time is in seconds
##
## Description:
## ------------
## button, mode:
## action is executed when the button is pressed longer than time
##
## GPIO:
## action1 is performed when the line is externally set to high
## action2 is performed when the line is released (or set to low)
##   (if the line is not connected, the value is low)
##
## if time is set, the action is executed after time seconds
##
## the actions can be any shell command (but must not contain '(' and
')')
##
## predefined actions:
##  - OFF:    power the system down. This only works if the system is
##            powered through the multi-purpose connector and the
ignition
##            signal is deasserted. After power down, the system boots
again
##            when ignition is asserted.
##            Before shutting down, all scripts in
/etc/scripts.d/shutdown.d/
##            are executed.
##  - CANCEL: cancel a running OFF countdown
##  - RESET:  resets the config (i.e. copy cablynx.conf.orig to
cablynx.conf)
##  - REBOOT: reboot the system
##  - HALT:   halts the system (will reboot eventually because of
watchdog)
##  - MOUNT:  mount all USB drives (usually done automagically)
##  - UMOUNT: unmounts all USB drives
##  - OUT_ON: switch GPO on
##  - OUT_OFF: switch GPO off
##  - SMS_STATUS: send information about all GPI pins as SMS back to the
##                sender of the command. Only works in the SMS section.

## actions for the Reset Button
button = 2, /bin/dd if=/etc/conf.d/cablynx.factory of=/etc/cablynx.conf
button = 5, /bin/touch /etc/reset && sync && /sbin/reboot -d 4
button = 10, echo resetalg | cablynxctrl

#mode = 2, /etc/scripts.d/local/test.sh
```

```
## actions for ignition signal
ignition = (/usr/bin/logger "Ignition on"), (/usr/bin/logger "Ignition
off")

## ign_boot: Tell the system what to assume about the ignition signal
## upon boot. Depending on this value and the current state of the
## ignition signal, an edge is immediately detected, e.g. if set to 1,
## and ignition is on when the system starts, it will immediately
## detect a positive edge and perform the corresponding action.
## Possible values;
## 0 (or unset): read current value of ignition upon start
## 1: assume that ignition was off during boot
## 2: assume that ignition was on during boot
#ign_boot = 2

## actions for input signals
input1 = (/usr/bin/logger "input 1 on"), (/usr/bin/logger "input 1 off")
input2 = (/usr/bin/logger "input 2 on"), (/usr/bin/logger "input 2 off")
input3 = (/usr/bin/logger "input 3 on"), (/usr/bin/logger "input 3 off")

## Hysteresis for input signals
## If not defined, 0 is assumed.
## These parameters must appear after the respective inputX above.
## Syntax: hysteresis[X] = positive[, negative]
## X = number of input (1-3), omit for ignition
## positive and negative values are number of previous samples that
## must have the same input value for the edge to be detected.
## One sample is taken every 250ms.
## If negative value is not given, the same value as for the positive
## edge is assumed.
## Examples:
## hysteresis = 0: as soon as a new value is detected on the input, the
## configured action is executed (behaviour in earlier versions).
## hysteresis = 2, 3: on a positive edge, 3 consecutive samples (i.e.
## two previous samples plus the current one) must be detected before
## the action is executed. This introduces a delay of 500ms compared
## to the default case (hysteresis = 0).
## On a negative edge, 4 consecutive samples must be found, introducing
## a delay of 750ms.
#hysteresis = 1
#hysteresis1 = 1
#hysteresis2 = 1
#hysteresis3 = 1

##  internal signals, don't change
gpio_ign = 46
################################################
## GPIO Port nums for AnyRover / CabLynx Eco
gpio_in1 = 30
gpio_in2 = 29
gpio_in3 = 28
gpio_off = 126
################################################
gpio_but = 39
gpio_mode = 41
# status led
gpio_status = 87

[gps]
#############################################################################
###  (7) GPS
#############################################################################

## start the GPS relay daemon (default: no)
start      = yes

## start gpsd program? (default: no)
## start = yes needs to be set for gpsd to be started.
## "run_gpsd = yes" does not imply "start = yes"
run_gpsd   = no

## gpsd port to listen for TCP connections (default: 2947)
## don't forget to open the port in the firewall section
gpsd_port  = 2947

## verbosity level of gpsd. 0=quiet (default: 0). The higher this
## value, the more messages gpsd will send to syslog.
## cf. documentation of gpsd
gpsd_debug = 0

## AssistNow: Send current almanac and ephemeris data to GPS receiver
## This speeds up time to first fix.
## The data in the file is only valid for a short period of time (days),
## so make sure the file is always current.
## If a file is specified here, it will be loaded into the GPS receiver
## upon system boot.
## This can also be done later through the cablynxctrl utility.
#assist_now = /etc/gps/current_1d.alp

## Targets to send GPS data to.
## - {tcp|udp}_target actively connect to the target host
## - tcp_server waits for incoming connections (don't forget to open
##   the port in the firewall section)
## - Default target port is 13179.
## - If the source address is omitted, the address of the interface
##   on the route to the target is used. Setting a different source
##   address is required when working with IPsec tunnels, since only
##   data originating in the internal net will be sent through the
##   tunnel. To send the GPS data through the tunnel, the source
##   must be set to eth0.
## - If the source port is omitted
##    - tcp uses a random source port
##    - udp uses source port 13179
## - serial_target sends data to serial port
## - file_target writes data to a file. If maxsize is given (in bytes),
```

```
##   then the file will be rotated and gzipped once it reaches this size
##   (i.e. when the file size is greater than maxsize, which is only
##   checked every 10 seconds). With the parameter rotate, the number
##   of old files to keep can be specified. rotate must be < 100.
##   If the logfile is on the root partition, then file size is limited
##   to 10MB and rotate to 1, to prevent filling the partition.
##   It is recommended to write to an SD-Card or USB Stick, where those
##   limits do not apply.
##   hook defines a program that is executed whenever the file is
##   rotated. It gets the file name as parameter. If no hook is defined,
##   the file is compressed using gzip. If you want to keep the files
##   uncompressed, but without defining a hook script, use /bin/true
##   as hook.
##   Default values: maxsize = 4MB, rotate = 5
## ** don't use source ports for tcp unless you really really need it
## udp_target = ret,target[:port][,[source[:port]|interface[:port]]
[,id:X]]
## tcp_target = ret,target[:port][,[source[:port]|interface[:port]]
[,id:X]]
## serial_target = ret,serport[,baudrate[,id:X]]
## file_target = ret,filename[,maxsize[,rotate[,hook[,id:X]]]]
## interfaces: eth0 (ethernet), ppp0 (modem), vlanX
## ret defines the return path, both for the GPS receiver and for
## commands to the system.
## ret=0 means that data is silently dropped.
## ret=1 means the return path for this connection is open,
##       i.e. all data sent are forwarded to the GPS receiver.
## ret=2 means the return path is open to send commands of the form
##       CBCTL:{command}, where command is one of the commands that
##       are valid within cablynxctrl.
## ret=3 means that both the GPS and the CBCTL paths are open.
## target,source can be IP addresses or host names. When using host
## names, make sure the system is able to resolve them (e.g. via DNS).
## id:X references a filter rule (see below) named X. If no rule is
given,
## all messages are sent out on this connection.
#udp_target = 1,192.168.3.2:13179,ppp0
#tcp_target = 0,192.168.17.42:12345,192.168.3.1:13245,id:rule
#serial_target = 0,/dev/ttyS1,38400
#file_target = 0,/media/sda1/logfiles/gpslog.txt,4000000,5
#file_target = 0,/media/sda1/logfiles/gpslog2.txt,4000000,5,/bin/true
#tcp_server = ret[,source_ip[:port]|interface[:port]]

## target filter rules
## With this filter, the number of messages sent to each target (or
server)
## can be limited. No new messages are created, so if the filter says to
## send every 5 seconds, but the source delivers messages only every
## 15 seconds, only these messages from the source will be sent.
## Syntax:
## filter = ID[:profile]; PATTERN=time[,dist][;PATTERN=time[,dist]]
## Multiple lines for the same ID are allowed. There is no difference
## between specifying all rules on the same line and on different lines.
## The ID is references from the (tcp|udp)_(target|server) attributes.
## Do not use the ID anytracker, as this is used internally for atmsg_*.
## Do not use the ID at_script, as this is used internally for
atmsg_script.
## profile is a number (0, 1, 2, ...), to be able to define different
## rules based on external variables. On start, profile 0 is applied.
## Changing the profile is done through the cablynxctrl utility.
## The message to be sent is matched against the PATTERN, and if it
matches,
## the messages is only sent if the time or dist constraints match.
## time (in seconds) defines the minimum interval between to messages.
## dist (in meters) defines the minimum distance the GPS receiver must
## have been moved between two messages.
## A value of 0 for both intervals means do not send any messages.
## Example:
## filter = rule1:0; $GPGGA = 5; $GPRMC = 5; $GP = 0
## Send GPGGA and GPRMC messages every 5 seconds, but no other GP
messages
## filter = rule1:1; $GPGGA = 0, 200; $GPRMC = 0, 200; $GP = 0
## Profile 1 of the same rule. Send GPGGA and GPRMC after moving 200m.
## If the GPS receiver does not move, no messages are sent.
## UBX messages can also be filtered:
## filter = ubx; UBX-NAV-EKF=10; UBX-CFG = 30; UBX = 0
## The file /etc/ubx.txt contains all available UBX class and id names.
#filter = rule; $GP = 1; UBX = 0

## tcp_init_str: Send this as the first string whenever a new tcp_target
## connection has been established. The value of the parameter is
## sent verbatim, without any modifications.
# tcp_init_str = $GPTXT,INIT,AnyRover (c) 2008-2012 by AnyWeb AG*20

## Send AnyTracker format messages to every _target. Messages
## can be filtered with filter statements:
## filter = at; { = 0

## Message format:
## Format of string to send (without line break):
## {2280123456789098|POSITION|14.07.2013|10.09:32|04724.1234|N|
##     00832.9876|E|415.0|3.0|246|5|1.67|99|0:0:30|300}\r\n
## {ID|TYP|DATE|TIME|LAT|NS|LONG|EW|ALT|SPEED|
##     DIRECTION|NSAT|HDOP|BATT|TIME_INT|DIST_INT}\r\n
## Currently only POSITION messages are generated.
## The BATT (battery level) field always contains 99.

## Enable creation of AnyTracker messages
atmsg_en = no

## Value to place into the ID filed of the message.
## Use <IMSI> or <IMEI> to use IMSI or IMEI from modem 1,
## <IMSI1> or <IMEI1> to read these values from modem 2.
atmsg_id = <IMSI>

## Early versions of AnyTracker contained a bug, where the time field
```

```
## was sent as hh.mm:ss instead of hh:mm:ss.
## If set to yes, the buggy format will be sent.
atmsg_buggytime = no

## Intervals to create AnyTracker messages. Format:
## atmsg_interval = time, dist [, profile]
## A message is created every time seconds, or when the device has
## traveled more than dist meters since the last message.
atmsg_interval = 10, 300

## Use UTC time in the messages [yes|no] (default: No)
## If set to no, local time zone is used
atmsg_use_utc_time = yes

## Use a script for message assembly
## The output from stdout will be sent to the server
## Messages from this script are generated in addition to
## the aforementioned AnyTracker message
##
## Environment variables that can be used in the script:
## ID:             atmsg_id (max 60 characters)
## IMEI:           IMEI of ppp0
## IMSI:           IMSI of ppp0
## TIME_UNIX:      Time in s since 01.01.1970
## TIME:           Formatted time (hh:mm:ss) incl. buggytime
## DATE:           Formatted date (dd.mm.yyyy)
## LATITUDE:       Latitude from GPGGA
## NS:             "N" or "S"
## LONGITUDE:      Longitude from GPGGA
## EW:             "E" or "W"
## ALTITUDE:       Altitude from GPGGA (precision: 0.1)
## SPEED:          Speed in km/h (precision: 0.1)
## DIRECTION:      Heading from GPRMC (precision: 0:1)
## NSAT:           Number of satellites from GPGGA
## HDOP:           HDOP from GPGGA
## ATMSG_INTERVAL: Time interval in s
## ATMSG_DISTANCE: Distance in m
## GPGGA:          Full GPGGA or GNGGA string
## GPRMC:          Full GPRMC or GNRMC string
#atmsg_script = /etc/scripts.d/atmsg_ruag.sh

## Intervals to create script messages. Format:
## atmsg_script_interval = time, dist, profile, pattern
## A message is created every time seconds, or when the device has
## traveled more than dist meters since the last message.
## profile: Identifier for multiple filter rules
## pattern: Define a pattern which matches the first characters
##          of the generated message so the filter is able to
##          distinguish the generated message from other messages.
##          Note: Always use the same pattern in all profiles.
#atmsg_script_interval =  10, 300, 0, POST

## Configure optional GPTXT messages.
## These messages can contain arbitrary information
## Syntax: gptxt = interval, action
## A GPTXT message is sent every interval seconds along with
## all GPXYZ messages from the GPS receiver to all configured targets.
## action can be the name of an executable (including path),
## or one of GPI, DIP or WLAN.
## When set to GPI, information about the GPI pins is sent in this
## format:
## GPTXT,IO,<ign>,<res>,<val>,<val>[,...]*
## example: $GPTXT,IO,0,1,0,0,0,0*64
## where <ign> is the state of the ignition signal (0 or 1),
## <res> is the state of the reset button (0 or 1),
## <val> are the values of all GPI pins.
## When set to DIP, the position of the DIP switches 1-6 are sent:
## $GPTXT,DIP,0,0,0,1,0,0*3f
## When set to WLAN, information about visible access points is
## sent, if the wlan card is configured as client:
## GPTXT,WLAN,<ssid>,<mac>,<channel_freq>,<signal>*
## example: $GPTXT,WLAN,guestWLAN,00:0d:ed:87:c9:f2,2412,-72*50
## Creating the WLAN information does not trigger a scan, but prints
## the cached values. When idle, the system only starts a scan for
## access points every 140 seconds or so, and then keeps these values
## cached until the next scan.
##
## When specifying an executable, the stdout of the program is sent
## via GPTXT in the form
## GPTXT,<STDOUT>*ck
## If the output contains newline characters or is longer than
## 70 bytes, it is split into multiple GPTXT messages.
## (NMEA specifies that messages must not be longer than 80 bytes).
#gptxt = 4, GPI
#gptxt = 5, /etc/scripts.d/some_fancy_script.sh
##
## Lines for proper operation of AnyControl.
#gptxt = 15, GPI
#gptxt = 60, /etc/scripts.d/gptxt_handlers.sh adc
#gptxt = 20, /etc/scripts.d/gptxt_handlers.sh m1 modem_at_all
#gptxt = 20, /etc/scripts.d/gptxt_handlers.sh m2 modem_at_all
#gptxt = 20, /etc/scripts.d/gptxt_handlers.sh int_traffic ppp0
#gptxt = 20, /etc/scripts.d/gptxt_handlers.sh int_traffic ppp1
#gptxt = 20, /etc/scripts.d/gptxt_handlers.sh esfcalibration
#gptxt = 20, /etc/scripts.d/gptxt_handlers.sh mipstatus ssid
#gptxt = 20, /etc/scripts.d/gptxt_handlers.sh bondixstatus
#gptxt = 20, /etc/scripts.d/gptxt_handlers.sh m1 roamingstatus
#gptxt = 20, /etc/scripts.d/gptxt_handlers.sh m2 roamingstatus
#gptxt = 60, /etc/scripts.d/shell.sh dualmodem status

## Write current GPTXT messages to a file.
## A set of the last GPTXT messages is written regularly to a file,
## where it can be read and parsed.
## gptxt_file: name of the file to write to. Directory where to
##             write the file must exist.
##             Recommended value: /var/gps/gptxt.txt
```

```
##             The GPTXT are indexed by the first field after GPTXT,
##             and only one message for every index is stored.
##             E.g. $GPTXT,INFO,Hello*  -> key is INFO
## gptxt_writeout: write file every n seconds. (10) Set to 0 to
##                 disable this feature.
## gptxt_clean: remove messages that are older than n seconds (60).
##              They are reinserted if they appear again.
gptxt_file = /var/gps/gptxt.txt
gptxt_writeout = 10
gptxt_clean = 60

## The NMEA strings are made available to applications through fifos.
## directory of these fifos.
## default: /var/gps
directory  = /var/gps

## enable or disable gps bypass
## sends gps data directly to the external serial port. This can be
## achieved as well by adding a serial target, but the bypass is faster.
## To enable the gps bypass the serial ports have to be enabled in the
## serports section. (default: no)
gps_bypass = yes

## TTY device of GPS receiver, don't change (default: /dev/ttyS2)
device     = /dev/ttyS2
## baudrate of GPS device (default: 9600)
baudrate   = 57600

## internal signals, don't change
gpio_reset = 38
gpio_on    = 125

[sms]
###############################################################################
###  (8) SMS Console
###############################################################################

## listen for SMS?
start = yes

## phone_number: list of phone numbers that are allowed to send
## SMS commands to the System.
## If list is empty or entry is missing, all numbers are allowed.
## This check is performed for all sms commands.
#phone_number = +41790123456, +41760987654

## interval: check for new messages every n seconds
interval = 15

## SMS console
## If console = yes, the system parses SMS with the command eco.
## All other defined commands are parsed in any case.
console = no

## If enabling the console by SMS, this key must be sent.
## If set to -, the console cannot be enabled with SMS.
console_key = -

## Answers to SMS commands
## These attributes define the behavior of the system concerning
## answers to SMS commands.
## if send_answer_back is set to yes, the answer will be sent back
## to the sender.
## send_answer_to contains a list of phone numbers where the answer
## will be sent to. The numbers must not contain spaces, and they
## are separated by spaces.
## Example:
## send_answer_to = 0790123456, +41760987654
send_answer_back = yes
send_answer_to =

## Catch_all hook
## This program is called if no suitable command is found for an SMS.
## It gets the number and the text of the SMS in environment variables:
## $PHONE_NUMBER and $SMS_TEXT
## If no catch_all is specified, undefined SMS messages are just dropped.
#catch_all = /some/executable/file

## Sender id format
## If an sms is received from a defined sender like SWISSCOM, the sender
## id can be used as text or number. Default value is number.
sender_as_text = no

## TPM support for AnyGator
## If wget downloads in AnyGator scripts are supposed to use a TPM key,
## enter key ID, certificate and CA certificate here.
## Syntax:
## anygator_tpm = <KEY_ID>
## anygator_cert = <PEM file>
## anygator_cafile = <PEM file>
#anygator_tpm = ae5608cb271e417783b8eeb886e4958ea891d561
#anygator_cert = /etc/certs/ipsec/cert.pem
#anygator_cafile = /etc/certs/ipsec/ca-cert.pem

## commands:
## command_name = hash, command
## command name: is sent by SMS (spaces are converted to underscores)
##               the command name must start with a lower case letter
## hash:         0: command can be executed without hash
##               1: command must have a valid hash signature
##                  (security warning: the hash is always the same for
##                   the same command)
##               2: command requires 3-way handshake (not implemented yet)
##               If a value for hash is omitted, 0 is assumed.
## command:      The command is passed to the shell,
##               and the answer sent back by SMS (the first 160 bytes).
```

```
##              The commands described in the [gpio] section can also
##              be used here.
##
## Examples:
## ping_router: pings the next hop on the default route
## ping_client: pings the first DHCP client
## position:    returns the current GPS position (don't use cat or tail
##              on the GPS fifos!)
#ping_router = 0, ping -q -c 4 `route | awk '/^def/{print $2}'` | awk
'BEGIN{a=0}/^---/{a=1;next}a'
#ping_client = 0, ping -q -c 4 `awk '/ List /{a=1;next}a{print $1;a=0}' <
/etc/hosts` | awk '/^---/{a=1}a'
#position    = 1, head -n 1 /var/gps/gpgga.fifo

## These command allows configuration over SMS. They are quite dangerous,
## therefore they are disabled by default.
## syntax: eco conf section[:name] attribute[+|-]=value
## attribute=value replaces the first AVP with matching attribute
## attribute+=value adds the AVP at the beginning of the section
## attribute-=value removes the AVP with matching attribute and value
## Further commands:
## eco enable [password]      - enable SMS console; password must be
##                              the console_key defined above
## eco disable                - disable SMS console
## eco list section [section] - sends back requested sections
## eco restart                - reload config
## eco reboot                 - reboot system
## eco reset                  - revert to factory defaults
## eco templ name             - load config from /etc/conf.d/name
## eco save name              - save config to /etc/conf.d/name
## eco net [args]             - customer specific.
##                              This script does not do anything, it has
##                              to be customized first.
eco_%       = 0, /etc/scripts.d/eco.sh %@

[modem]
############################################################################
### (9) Modems
############################################################################

name      = modem1

## set radio band for the modem
## possible values (default = 3) for 3G modems:
## 0 = Automatic
## 1 = UMTS 3G only
## 2 = GSM 2G only
## 3 = UMTS 3G preferred
## 4 = GSM 2G preferred
## Hint: in 2G mode, the modem cannot receive SMS under load
## For LTE modems:
## 0-2: identical to 3G modems
## 3, 4: Automatic
## 5: GSM and UMTS only
## 6: LTE only (only use this option with a LTE antenna)
## 7: GSM, UMTS, LTE
## 11, UMTS and LTE Only
## 12, GSM and LTE Only
band = 3

## set roaming option.
## set parameter to yes to disable roaming
disable_roaming = no

## PIN for the SIM card. This value is only used if the SIM card
## asks for a PIN code. The code can be 4 to 6 digits.
## If the modem asks for another code (e.g. PUK, PIN2), it has to
## be fixed manually.
## It is best to use SIM cards with the PIN disabled.
#sim_pin = 1234

## IMSI checker: With this feature, the IMSI (number of the SIM card)
## can be checked before starting ppp. Different actions can be taken
## upon match or mismatch: start ppp on a different interface (e.g. ppp5)
## or do not start ppp at all.
## To find out the IMSI of the currently inserted SIM card, use one of
## these commands from the shell:
##   at at+cimi
##   id2 /dev/clhip
## If several rules are given, they are parsed in the order they appear
## in the config file. Rules have the form "<IMSI>, X", where X is a
## number in the range -1 .. 2147483648. If the IMSI matches, ppp is
## started on interface pppX. If X is negative, ppp is not started.
## The IMSI "-" matches everything, so it makes no sense to put more
## rules after one rule with IMSI -; they are never tested.
## Examples (with IMSI 228013520284438):
## To only start ppp for one specific IMSI, use this:
##   imsi = 228013520284438, 0
##   imsi = -, -1
## To start ppp for one IMSI on ppp0, and on ppp100 for every other:
##   imsi = 228013520284438, 0
##   imsi = -, 100
## To not start ppp for one particular IMSI, but for every other:
##   imsi = 228013520284438, -1
##   imsi = -, 0
#imsi = 228013520284438, 5

## Whether to wait until the SIM card signals it is ready.
## Some SIM cards need some time after entering the PIN until
## they are ready, and in some cases the system is too fast
## with starting the connection, which results in a failed
## connection attempt.
## It is recommended and should be safe to keep this set to yes.
wait_for_sim = yes

## GPIO port the modem is connected to. Don't change!
```

```
## If gpio is unset, the modem is not switched on upon startup
gpio      = 47
disable   = 48
cmd_on    = 0

## Slot where the modem is placed.
## For the CabLynx Eco / AnyRover, this is 0
slot = 0

## Get modem status informations and store them to a file
## Default value: yes
get_modem_status = yes

## Define how often modem status informations get collected.
## default value: 60 (seconds)
status_interval = 60

## Show signal RX level on external LEDs.
## When set to no, this command will not touch the external LEDs.
## This may be needed if the LEDs are used to display something else.
## Default value: yes
show_rx_led = yes

## log bad modem connection, when it is worse than the defined rx value.
#log_value = -120

## Phone calls (via cablynxctrl):
## ringtime = time in seconds the call is left in the ringing state
## if it is not answered. Call setup is included in this time, so
## it makes no sense to put values smaller than approx. 5s here.
## Default: 30
## calltime = time in seconds the call is kept open. Default: 10s
## max_call_retry: number of times the sysytem tries to call the number
## if something fails (connection not up, lockfile found, ...)
## Default: 10
#ringtime = 30
#calltime = 10
#max_call_retry = 10

## device files of the modem. Don't change unless you know what you do.
modem    = /dev/clmodem
hip      = /dev/clhip
ctrl     = /dev/clctrl
gps      = /dev/clgps

[modem]
name = modem2
band = 3
disable_roaming = no
wait_for_sim = yes
#sim_pin =
#imsi = 123456789012345, -1
gpio = 3204
disable = 3203
cmd_on = 1
slot = 1
get_modem_status = yes
status_interval = 60
show_rx_led = no
#log_value = -120
modem = /dev/clmodem
hip =   /dev/clhip
ctrl =  /dev/clctrl
gps =   /dev/clgps

[usb]
###############################################################################
### (10) USB ports
###############################################################################

## switch power on USB ports on? If set to no, only self-powered devices
## can be operated at the external USB ports. (default: no)
## The internal WLAN card is also concerned by this flag. If set to no,
## WLAN will not work.
poweron = yes

## switch power on for the three external USB ports individually. To do
## this, poweron has to be set to yes. usb1 and usb2 are the connectors
## for the optional WLAN modules, usb4 is the USB connector on the
## outside of the device.
usb1 = yes
usb2 = yes
usb4 = yes

## If this parameter is set to yes, wlan0 and wlan1 are exchanged.
switch_wlan = no

## Restart WLAN card if it produces Vendor Request Failed messages.
## It is currently recommended to set to yes.
restart_wlan = yes

##-----------------------------------------------------------------------
## SD-card management
##-----------------------------------------------------------------------

## Switch power on for the SD-card?
start_sdcard = yes

## if yes, a drive connected on the USB port or an SD-card is mounted
## automatically (default: yes)
automount = yes

## Ignore filesystem errors and continue. If set to no, the device
## will be remounted read-only.
ignore_errors = yes
```

```
## Mount points for partitions on the SD-card.
## This parameter can appear multiple times, once for each partition
## to be mounted.
## The partitions are only mounted if automount = yes.
## sdpart = part-num, mountpoint
## Example: sdpart = 1, /media/sdcard1
sdpart = 1, /media/sdcard1

[dhcp]
########################################################################
### (11) DHCP
########################################################################

## Configure the DHCP server on the AnyRover.
## The DHCP server only serves on one interface (eth0, vlanX).
## It is possible to start multiple servers for different
## interfaces, just insert multiple sections [dhcp], one
## for each interface.
## The IP addresses must correspond to the address set in the
## [system] section (for eth0), and for the address of the VLAN
## set in the [switch] section.

## interface to run dhcp server on. One of
## eth0, vlan1, vlan2, vlan3, vlan4, wlan0
## this attribute must appear first in the section
name = eth0

## whether to start dhcp server
start = yes

## Location of lease file. If not given, lease file is written to
## /var/lib/misc/udhcp.leases.<IFACE>
## which resides on a RAM disk and is lost after a reboot.
#lease_file = /etc/udhcpd/leases.eth0

## Logging. If set to syslog, the dhcp server will log its actions
## to syslog.
#log = syslog

## UDP Port to listen for DHCP requests. Default: 67
#port = 11167

## first address of dynamic range
dhcpd_start = 192.168.1.11

## last address of dynamic range
dhcpd_end = 192.168.1.12

##
## Bootp options
## These options are placed in the body of the dhcp offer
##

## next_server: IP address to be placed in the "next server" field
#next_server = 192.168.1.9

## server_hostname: Server hostname to announce to clients
#server_hostname = localhost

## boot_file: Name of the file the client uses to boot
#boot_file = kernel.img

##
## DHCP options
## These options are appended to the dhcp offer
##

## Netmask or prefix of the dynamic range.
## If both are given, the value of netmask parameter is used.
## If neither is given, the class based netmask of the start address
## is used.
#prefix = 24
netmask = 255.255.255.0

## Default router to tell the clients.
## This parameter can appear multiple times to send multiple routers.
## If not set or set to default, the IP address of the interface the
## dhcp server is running on is used as router address.
#router = default
router = 192.168.1.3

## Name servers to hand out to the clients. This parameter can
## appear multiple times.
dns = 192.168.1.3
#dns = 164.128.36.74
#dns = 164.128.36.75

## Lease time, given in seconds (default: 1 day)
## The time can be given as a number followed by one of min, hour,
## hours, day, days to give a longer timespan.
## There must be a space separating the number and the unit.
#lease = 86400
#lease = 1 day
#lease = 12 hours

## Further options available and description of parameter:
## Options with a * can appear multiple times.
## For more information check some dhcp documentation.
## timezone = 7200              #time offset to UTC in seconds
## *logsrv = 192.168.1.3        #IP address of log server
## *lprsrv = 192.168.1.5        #IP address of line printer server
## hostname = host.name.local   #hostname of the client
## bootsize = 6                 #size of boot file in 512-Byte blocks
## domain = mydomain.local      #Domain name for the client to use in DNS
## swapsrv = 192.168.1.6        #IP address of swap server
## rootpath = /root/path        #Path to client's root disk
```

```
## ipttl = 16                   #default TTL for client to use
## mtu = 1500                   #MTU for client to use on this interface
## broadcast = 192.168.1.255    #Broadcast address in client's subnet
## nisdomain = domainname       #NIS domain name for client
## *nissrv = 192.168.1.7        #IP address of NIS server
## *ntpsrv = 192.168.1.8        #IP address of NTP server
## *wins = 192.168.1.9          #IP address of WINS server
## requestip = 192.168.1.10
## dhcptype = 8
## serverid = 192.168.1.1       #IP address to send as server ID
## message = some fancy message
## vendorclass = CLASS string
## clientid = id of client
## tftp = 192.168.1.11          #IP address of tftp server
## bootfile = path/to/boot/file #File the client uses to boot
## wpad = autodiscovery         #MSIE' "Web Proxy Autodiscovery Protocol"
## vendorspec = 41:65:d:a:0     #Hex string to send as vendor specific
data
## *sipsrv = type:URL           #SIP server; type 0 = URL, 1 = IP addr
##                              # multiple options only for identical
type
## captiveport114 = URL         #Captive portal according to RFC8910
## captiveport160 = URL         #Captive portal according to RFC7710

##
## static leases
## Place any static leases here. An entry has the form
## static_lease = MAC-addr IP-addr
##

#static_lease = 01:23:45:67:89:ab 192.168.1.1

[dhcp]
name = vlan1
start = no
dhcpd_start = 172.24.34.11
dhcpd_end = 172.24.34.254
netmask = 255.255.255.0
router = 172.24.34.1
dns = 172.24.34.1

##--------------------------------------------------------------------
## DHCP Relay
##--------------------------------------------------------------------
[dhcprelay]
## Configure a dhcp relay.
## start defines whether to start the service
start = no

## client: List of interfaces (separated by comma) to listen for
## DHCP requests on.
## If left empty, listen on all interfaces.
## If the interface is preceded by a '!', it is excluded from the list,
## i.e. "client = !vlan1" means to listen on all interfaces except vlan1.
client = vlan1, vlan2

## server: List of servers (separated by comma) to forward the DHCP
## requests to. This can be IP addresses or interfaces. If an IP
## address is given, the packet is unicast to that IP address. If an
## interface is given, the packet is broadcast on that interface, and
## the interface is excluded from the list of interfaces the program
## listens on.
## The gw-addr in the DHCP header is filled with the interface the
## packet was received on.
server = 192.168.25.1

[ftp]
##############################################################################
### (12) FTP
##############################################################################

## start ftp deamon? (default: no)
start = no

## use basic configuration options? (default: no)
basic = yes

## allow anonymous logins? (default: no)
anonymous = yes

## directory for anonymous access (default: /var/ftp)
## ATTENTION! The directory /var/ is on a RAM disk, i.e. all files
## in this directory do NOT survive a reboot.
anonymous_dir = /var/ftp

## allow anonymous users to upload files? (default: no)
anonymous_write = no

## allow anonymous users to delete files? (default: no)
anonymous_delete = no

## direct configuration
## these options are directly placed into the vsftpd.conf file
## Note: vsftpd doesn't allow white space around the '=' sign
#option = hide_ids=YES

[tftp]
##############################################################################
### (13) TFTP
##############################################################################

## start tftp daemon? (default: no)
## don't forget to open the port in the firewall section
start = no

## allow file uploads? (default: no)
```

```
upload = no

## root directory of tftp daemon (default: /tftp)
rootdir = /tftp

## UDP Port to listen on (default: 69)
port = 69

[firewall]
################################################################################
### (14) Firewall
################################################################################

## This section defines firewall and packet mangling rules.
## Firewall rules only decide what to do with the packets (reject,
## accept), based on different fields in the packet.
## Packet mangling rules modify the packet. NAT or port forwarding
## are packet mangling rules.

##-------------------------------------------------------------------
## Global firewalling parameters
##-------------------------------------------------------------------

## Define whether bridged packets are seen by the firewall.
## This can only be set globally, not per bridge.
filter_bridged = yes

## Define whether vlan tagged frames on the bridge are seen by
## the firewall. This can only be set globally, not per bridge.
filter_vlan = yes

## Enable forwarding in the kernel.
## If set to no, the system will not route packets.
forward = yes

##-------------------------------------------------------------------
## Packet logging
##-------------------------------------------------------------------

## NFLOG: Special log target that can be used to trigger actions
## when certain packets appers (see nflog below).
## This parameter defines whether to start this service.
## It is still possible to define nflog rules when this is set
## to no, but then the system will not evaluate the packets.
nflog_start = no

## Script to execute when a matching packet appears. This script will
## get all relevant packet information in the environment (NFLOG_*
variables).
## The default script /etc/scripts.d/nflog.sh explains the details and
## executes all scripts found in /etc/scripts.d/nflog/
## It is recommended to place custom scripts in this directory and leave
## the default wrapper script in place.
nflog_script = /etc/scripts.d/nflog.sh

## NFLOG group. It is possible to create different groups where
## messages are sent to. This value configures the group to be used
## on the system. Possible values: 1-32
nflog_group = 7

## NFLOG payload length: Copy up to this number of bytes from UDP packets
## to variable NFLOG_PAYLOAD. If a non-printable character is encountered
## before the required number of bytes is read, reading stops.
## Payload is only copied for UDP packets.
nflog_payload_length = 64

##-------------------------------------------------------------------
## Firewall rules
##-------------------------------------------------------------------

## Define whether to start all firewall rules.
start_firewall = yes

## The firewall is implemented using Linux' iptables.
## The rules are inserted in the order they appear in the config file,
## so make sure to place them in the correct order.
## Upon booting the AnyRover, the switch is enabled only after the
## firewall rules have been set.

## Basic rules:
## - deny everything addressed to the system via policy
##   (can be overridden by rules)
## - deny everything passing through the system via policy
## - allow ICMP echo request (ping)
## - allow established connections
## - allow related connections (e.g. FTP data, ICMP errors)
basic = yes

## Allow creation of new chains
## new_chain = NAME[, CHAIN[:POS[:COND]]]
## creates the chain NAME and inserts a jump to this chain into
## chain CHAIN at position POS.
## COND can be additional iptables flags, e.g. "-i br0".
## Note: Position can only be relative to firewall rules defined above
## this line in this file. To place it after other rules, move this
## attribute further down.
## The name must not contain spaces or '_'. To create multiple chains,
## use multiple entries.
#new_chain = mychain1
#new_chain = mychain2

## Firewall rule definitions
## Syntax:
##  target = [SRC][,[!]proto[,DST]][,R:RATE][,L:prefix][,I:ICMP]
##           [,MAC:[!]ADDR]
##  SRC,DST = [[!]if] [ipsec] [[!]net][:[!]ports]
```

```
##  RATE = [rate][:burst]
##  ADDR = {MAC address}
##  ICMP = (icmp-port-unreachable|icmp-host-unreachable|
##          icmp-port-unreachable|icmp-proto-unreachable|
##          icmp-net-prohibited|icmp-host-prohibited|
##          icmp-admin-prohibited) (default: icmp-port-unreachable)
##
## target has the form rule[_chain]. If _chain is omitted, _in is assumed
## Both rule and chain must not contain spaces or '_'.
## rule can be one of
##   accept: let the packet pass
##   drop: drop the packet to the floor
##   reject: drop the packet and return icmp message to sender
##           -> use drop unless you know that you need reject
##   return: stop processing and return to parent chain
##           or apply chain policy
##   log: log packet to syslog. This rule does not stop processing!
##   nflog: log packet to netlink. This rule does not stop processing!
##   name of custom chain: processing will continue in this chain
## chain can be one of:
##   in: add rule to INPUT chain
##   fw: add rule to FORWARD chain
##   out: add rule to OUTPUT chain
##   everything else: add to custom defined chain (which must exist)
##
## if: input/output interface
##   interface: eth0, vlanX, wlan0, tunlX, tunX, ppp0, ...
##   -> Specifying an iif only makes sense in the _in or _fw rules.
##   -> Specifying an oif only makes sense in the _fw or _out rules.
##   For bridges, the physical interface can be specified:
##    [brX]>phyIF, e.g. br0>vlan1, >vlan2
##       accept = br0>vlan1,tcp,:22
## ipsec: The rule only matches if the cleartext packet arrives (SRC) or
##        leaves (DST) through an IPsec tunnel.
##        The keyword ipsec can only be used either in SRC or DST,
##        but not both (e.g. in _fw rules).
## net: source/destination IP address
## port: source/destination port; only used if proto is udp or tcp. If
##       there are more than one port, they have to be separated with a
##       colon.
## proto: protocol (tcp, udp, esp, icmp)
## rate: rate limit traffic (mostly used for log target, to prevent
##       log file flooding). E.g. 3/sec, 2/min, 7/hour, 12/day
##       do not use to limit bandwidth
## burst: maximal initial number of packets (default: 5) to match
## prefix: text to be used as log prefix; only valid for log targets.
##         The text must not contain , or ' characters. It can be
##         enclosed in double quotes ("), this is only necessary if
##         text ends with spaces.
## ICMP: ICMP message to send back; only valid for reject targets.
## ADDR: Filter based on source MAC address of packet.
## An exclamation mark (!) inverts the matching, i.e. the rule then
## matches everything except the given value.
##
## Example: rule_fw = vlan1 192.168.1.0/24,tcp,vlan2 10.0.0.0/8
##    will accept packets originating in the net 192.168.1.0/24, entering
##    on interface vlan1, destined for 10.0.0.0/8 and leaving on vlan2

#accept = ,tcp,:21              # allow ftp connections
accept = ,tcp,:22               # allow ssh connections
#accept = eth0,tcp,:23          # allow telnet from inside
#accept = ppp0 ipsec,tcp,:23    # allow telnet through IPsec tunnel
accept = eth0,udp,:53           # allow DNS
accept = eth0,udp,:67:68        # allow DHCP
#accept = eth0,tcp,:80          # allow access to the Webserver
accept = eth0,udp,:123          # allow for local NTP clients
#accept = ppp0,udp,:123         # allow for remote NTP clients
#accept = ,udp,:500             # allow IKE (IPsec)
#accept = ,esp,                 # allow IPsec
#accept = ,udp,:4500            # allow IPsec NAT-T
#accept = ,udp,:1194            # allow OpenVPN
#accept = eth0,tcp,:2947        # allow access to gpsd daemon
#accept = ,tcp,:13180           # allow access to GPS server
#accept = ,tcp,:18080           # allow access to bonding proxy
# log attempts to access via telnet on ppp0
#log    = ppp0,tcp,23,R:3/min,L:"TELNET ON PPP0: "
#drop   = ppp0,tcp,23           # block telnet traffic on ppp0
#reject = ,tcp,:113             # block ident request
#accept_fw = vlan1,,ppp0        # allow all traffic from vlan1 to ppp0

## direct rule definition, value is directly passed to iptables:
## (For more information, see documentation of iptables, e.g.
##  http://www.netfilter.org)
##   rule = {iptables parameter}
##   rule = -A INPUT -i eth0 -p udp --dport 123 -j REJECT

##------------------------------------------------------------------
## Packet mangling rules
##------------------------------------------------------------------

## Define whether to start packet mangling rules
start_mangle = yes

## Network Address Translation
## list of interfaces to perform NAT on, i.e. all packets leaving on
## one of these interfaces has its source address set to the address
## of the outgoing interface.
#nat = vlan1, vlan2, wlan0
nat = ppp0

## Allow creation of new chains
## new_natchain = NAME[, CHAIN[:POS][, CHAIN[:POS]]]
## creates the chain NAME in the NAT section and inserts a jump to
## this chain into chain CHAIN at position POS.
## For position, see attribute new_chain.
## The name must not contain spaces or '_'. To create multiple chains,
```

```
## use multiple entries.
#new_natchain = mynatchain

## Port forwarding
## To forward a port to a different host:
## portfw = [proto],(tarip|iface)[:tport],dstip[:dport][,srcnet]
## proto: protocol (tcp, udp, ...); if left blank, all packets that match
##                                  the rest of the rule are forwarded
## tarip|iface: IP address or interface that is the target of the packet.
##              Can be an address range (e.g. 192.168.2.0/24).
## tport: port the packet is addressed to. Can be left blank
## dstip: IP address where the packet should be sent to. Mandatory
## dport: port where the packet should be sent to. If left blank, the
##        original port number is taken.
## srcnet: Network where the packet comes from. Can be an address range.
##
## Examples:
## forward all connections to port 80 (http) to webserver with address
## 172.24.17.42 and change to https (port 443)
#portfw = tcp,ppp0:80,172.24.17.42:443
#portfw = ,192.168.4.0/24,192.168.5.1,10.1.1.0/24

## Source NAT and Destination NAT
## Syntax (similar to firewall rules above):
## snat[_chain] = [SRC],[proto],[DST],T:target
## dnat[_chain] = [SRC],[proto],[DST],T:target
## SRC,proto, and DST are used to match packets. When a match occurs,
## the source/destination address/port is changed to target.
## Input interface matching is not possible for snat, output interface
## matching for dnat.
## Ranges are supported in targets, both for ports and IP addresses. The
## system will choose an appropriate value from the range automatically.
## SNAT is only applied to packets leaving the system, after a routing
## decision has been made, but before the packet is checked for IPsec
## encryption.
## DNAT is applied to packets entering or passing the system, before
## a routing decision is taken.
## DNAT is essentially the same as portfw above, but with different
## rule syntax (similar to all other rules).
## Use case: syslog cannot be configured with a source address, it always
## takes the address of the interface on the direct route to the
destination.
## To send syslog traffic into an IPsec tunnel, use an snat rule like the
## first example below.
## 10.11.12.13 is the syslog server, 192.168.1.3 our internal address.
## Example:
#snat = ,udp,10.11.12.13:514,T:192.168.1.3
#dnat = ,tcp,192.168.1.0/24,T:10.1.2.3:8000-8020
#snat = 10.11.12.0/24,tcp,192.168.1.24,T:10.1.2.1-10.1.2.5
## The same rules as the portfw examples above:
#dnat = ,tcp,ppp0:80,T:172.24.17.42:443
#dnat = 10.1.1.0/24,,192.158.4.0/24,T:192.168.5.1

## TCP MSS modification
## These rules can be used to alter the MSS of TCP SYN packets.
## This can be useful when traffic has to pass through a tunnel and
## automatic detection does not work for some reason.
## MSS mangling can be placed in one of the five chains INPUT,
## OUTPUT, FORWARD, PREROUTING, and POSTROUTING:
## - INPUT: for packets destined to the system
## - OUTPUT: for packets from the system
## - FORWARD: for packets passing through the system
## - PREROUTING: for all incoming packets, before a routing decision
##               is taken, i.e. INPUT and FORWARD. No output interface
##               can be used in the PREROUTING chain (no routing
##               decision taken, output interface not known yet).
##               A destination network is allowed though.
## - POSTROUTING: for all outgoing packets, after routing, e.g.
##                for FORWARD and OUTPUT. No input interface must
##                be used in the POSTROUTING chain.
##                A source address is allowed though.
## Syntax (identical to rules above), proto must be set to tcp:
## tcpmss_chain =  [SRC],proto,[DST],M:MSS
## Examples:
# tcpmss_in = ,tcp,10.10.0.0/16,M:1300
# tcpmss_fwd = 192.168.1.0/24 vlan1,tcp,vlan2,M:1374
# tcpmss_out = ,tcp,wlan0,M:1356
# tcpmss_POSTROUTING = ,tcp,192.168.17.0/24,M:1420
# tcpmss_PREROUTING = tunl0,tcp,192.168.17.0/24,M:1444

[dyndns]
################################################################################
### (15) DynDNS
################################################################################

## start dyndns client?
start = no

## username and password for dyndns
username = user
password = pass

## dynamic hostname(s)
## this attribute can appear multiple times
hostname = myhost.dyndns.org

## further options for the inadyn program
## don't remove the option syslog unless you know what you do
option = syslog
#option = update_period_sec 60

[ppp]
################################################################################
### (16) PPP
################################################################################
```

```
## name of the modem section
## must be the first parameter in this section
modem = modem1

## start ppp daemon?
start = yes

## Username for 3G access
user =

## Password for 3G access
password =

## Set the ppp connection as the default route?
defaultroute = yes

## Set the metric of the default route (default: 0)
defaultmetric = 10

## Use DNS entries sent by peer?
usedns = yes

## Debug: prints detailed information about the dial-in process
## into the log file. Default = no
debug = no

## configure connection through the modem
## basic = yes takes the default ppp config file (default: no)
basic = yes

## chat_verbose: if set to yes, chat logs the execution state as well
## as all text sent and received during dialling.
## Only has an effect if basic=yes
chat_verbose = yes

## chat_script designates the chat script-section to use
## if the name is basic, the template is taken and the parameter 'apn'
## replaced in the basic config file
chat_script = basic

## restart modem when ppp goes down? (default: yes)
restart = yes

## Time [seconds] to wait until modem is switched on again.
## Must be >0 (default: 2)
timeout = 2

## If set to yes, do not restart when connection fails with NO CARRIER.
## This leads to faster reconnect times after losing connection because
## of no reception in dead zones.
#hold_nocarrier = yes

## filter: packets that match this filter trigger dial on demand
## and reset the idle counter. If not set, all packets match.
## Syntax is similar to tcpdump, see tcpdump man-page for further
details.
## Expressions that are inappropriate for ppp link such as ether and arp
## are not permitted.
## Syntax:
## [(] [not] expr [)] [and|or] [[(] [not] expr [)]]
## if src and dst are omitted, both directions match
## [src|dst] host HOST
## [src|dst] net NET [mask MASK]
## [src|dst] port PORT
## [src|dst] portrange RANGE
## ip proto \\(icmp|ah|esp|tcp|udp)
## (inbound|outbound)
## expr RELOP expr
## RELOP is one of <, >, <=, >=, =, !=
## expr can contain integers, +, -, *, /, &, |, <<, >> (as in C)
## PROTO[expr:size]
## Examples:
## filter = outbound and not icmp[0] != 8 and not tcp[13] & 4 != 0
## filter = outbound and not ((tcp[13] & 4 != 0) or (icmp[0] = 3))
## filter = outbound and ip proto \\esp
filter =

## options are directly added to the ppp config file. For details see
## man pppd(8)
## Some useful options:
## - demand:    enable dial on demand.
##              The device is created and routing set up, but
##              the connection is only established upon demand
## - persist:   pppd doesn't terminate when the connection goes down,
##              but waits for the next connection request
## - idle n:    if the link is idle for n seconds (no traffic),
##              it is taken down
## - holdoff n: wait for n seconds before re-initiating the link
##              after it terminates. Does NOT apply if the link
##              goes down because it was idle
##              Use more than 10s if modem restart is enabled,
##              because the modem needs roughly 10s to come back.
#option = demand
#option = persist
#option = idle 300
#option = holdoff 15

[ppp]
modem = modem2
start = no
user =
password =
defaultroute = no
defaultmetric = 20
usedns = yes
debug = no
```

```
basic = yes
chat_verbose = yes
chat_script = basic
restart = yes
timeout = 2

[chat_script]
###========================================================================
### (16a) chat scripts
###========================================================================
## The chat script prepares the modem and dials the ISP. It consists
## of a series of AT commands for the modem.
## This sections does not allow for comments on the same line as
## config directives (since the dial command contains a '#' character)

## chat script for pppd.
## basic: set APN for basic chat script
name = basic
apn = gprs.swisscom.ch
#apn = internet

##------------------------------------------------------------------------
[chat_script]
## chat script for pppd
## the script is referenced in the ppp section by its name (name=...)
## all script parameters are placed in the chat script
name = anyweb
script = '' AT
script = OK ATZ
script = OK 'AT+CGDCONT=1,"IP","my.apn"'
script = OK ATD*99#
script = CONNECT ''

[wan]
###========================================================================
### (16b) WAN
###========================================================================

## Start WAN connection. This is only active if [ppp] start = no
## for the same modem.
start = no

## Name of the corresponding modem section.
modem = modem1

## APN to use for connection.
apn = gprs.swisscom.ch

## Username for 3G/4G access
user =

## Password for 3G/4G access
password =

## IP parameters. See ipaddr in [system] section for description.
ipaddr = dhcp default nolinklocal dns

## Radio Access Technology (RAT). Define which technologies to use.
## Possible values:
## 0, 3, 4: Automatic
## 1: UMTS 3G Only
## 2: GMS 2G Only
## 5: GSM and UMTS Only
## 6: LTE Only
## 7: GMS, UMTS, LTE
## Default value (if not specified): 3
#radio_access = 5

## If set to yes, all chat messages are logged in the log file.
chat_verbose = yes

[ipsec]
###########################################################################
### (17) IPsec
###########################################################################

## configure IPsec connections
## This section defines one IPsec tunnel between the system
## and one peer. Through this tunnel, different local and remote
## networks can be connected.
## To define multiple tunnels with different peers, insert
## one ipsec section for each tunnel.

## start = [yes|no] defines whether to start IPsec (default: no)
start = no

## Name of the connection
## If the connection has multiple local or remote subnets, a
## number is appended to the name.
## If no name is given "ipsecX" is used, with X the number of the
## ipsec section.
name = net-10

## What to do when ipsec is started. Possible values:
## start: bring up the connection
## route: load connection, start as soon as traffic wants to use it.
## add:   set everything up, but do not initiate -> wait for peer
setup = start

## IKE version to use (1 or 2). Default: 2
ike = 2

## Define IPsec mode: tunnel or transport.
## With tunnel mode, routers securely connect subnets on both sides.
## With transport mode, host to host communication is secured.
## If set to tunnel, both local_net and remote_net must be defined.
```

```
## If set to transport, local_net and remote_net are ignored, but
## local and remote must be set.
## Main use for transport mode: when using bondix tunnel, create a
## transport mode encrypted channel for every uplink, and then run
## the bondix tunnel over these links. This way, the bondix client
## sees plain text traffic and may use the TCP proxy for speedup.
## When running IPsec over Bondix, the client only sees UDP traffic
## and can not optimize.
## Default mode is tunnel.
#mode = transport

## IKE fragmentation for IKEv2
## If set to yes, large IKE messages will be sent in fragments.
## If set to no (default if not set), IP fragmentation might be applied
if
## IKE messages are larger than 1500 bytes.
fragmentation = yes

## MobIKE for IKEv2
## Enable the MobIKE extension (RFC 4555).
## With MobIKE, the AnyRover can renegotiate a tunnel if the local
## IP address of the tunnel endpoint changes.
## Hint: if MobIKE is enabled, tunnel setup (IKE protocol) will use both
## UDP ports 500 and 4500. Without MobIKE, only port 500 will be used;
## port 4500 is then only used when NAT-T is needed.
## MobIKE is not meant to dynamically switch to another interface for
## use as tunnel endpoint, only to handle changing addresses (and thus
## attachement points to the Internet) on a single interface.
mobike = no

## scripts to perform actions on certain states must be placed in
## /etc/scripts.d/ipsec-hooks , and they must be called one of
## up-host down-host up-client down-client
## This path can either be an executable, which is run,
## or a directory. Then all executables within named *.sh
## are executed.
## These scripts can be defined using the scripts section in this
## config file.

##---------------------------------------------------------------------
## Addressing
##---------------------------------------------------------------------
## remote: peer for the IPsec tunnel (ip address, hostname in quotes ("))
## If using a hostname, make sure it can be resolved.
## To allow road warriors with unknown IP addresses to connect, specify
## remote = any. To limit to certain IP addresses, see remote_range
below.
## Must be set if mode is set to transport.
remote = 192.168.17.42
#remote = "vpn.example.org"
#remote = any

## local: defines which interface the IPsec service listens on
## possible values: eth0 (local), ppp0 (3G), vlanX
## If left blank, the IPsec service listens on the interface that
## is the direct path to the remote host.
## Must be set if mode is set to transport.
#local = ppp0

## local_within: if route to remote goes over one of the interfaces
## listed here, then this IPsec connection is started.
## If interface is not listed, this IPsec connection is not started.
## If list is empty or parameter not defined, the IPsec connection is
started.
#local_within = vlan1 vlan2

## local_net: list of local networks, separated by space. The list can
##            contain IP addr/prefix pairs as well as interface names.
##            If not set, the address of the first configured interface
##            of eth0, vlanX is taken.
## Value is ignored if mode is set to transport.
# local_net = eth0 br0
# local_net = 192.168.1.0/24
local_net = vlan1: vlan2

## Limit IPsec tunnel to a single protocol and/or port. Both protocols
## and ports can be specified by name or number as given in
## /etc/protocols and /etc/services.
## Syntax: [proto][,[sport][,dport]].
## Port numbers are given for traffic to the peer; for traffic from the
## peer, the port numbers are exchanged.
## Examples:
#protocol = tcp, http
#protocol = udp
#protocol = , 443
#protocol = udp, 67, 68

## remote_net: list of remote networks, separated by space. The list
##             contains IP addr/prefix pairs. If set to any, the server
##             takes the remote net as advertised by the client. This
##             is used for road warrior setups.
## Value is ignored if mode is set to transport.
# remote_net = 192.168.18.0/24
# remote_net = 192.168.18.0/24 172.23.0.0/16
remote_net  = 0.0.0.0/0

## remote_range: when configuring remote_net as "any", this parameter
## limits the range of networks the remote host can advertise, as the
## network must be a subnet of the network given here.
#remote_range = 192.168.0.0/16

## remote_address: The internal source IP address to use in a tunnel
## for the remote peer. This is needed for example when the peer is
## a Cisco VPN Client.
## This can be set to %config, then it will echo back the address
## proposed by the remote peer.
```

```
#remote_address = 192.168.21.1

## tunnel: if configuring multiple local and remote nets, it is
## possible to define which local nets can talk to which remote
## nets. If this attribute is omitted, all local nets can talk to
## all remote nets.
## All local nets are labeld with a number, starting from 1.
## All remote nets are labeled with a letter, starting from a.
## The tunnel attribute contains all number-letter pairs of allowed
## connections, separated by space.
## If the list starts with an '/', the listed connections define the
## forbidden ones, with all others allowed.
## This parameter can also be used to define source policy routes for
## the connections. If a pair is followed by a colon and optionally
## an IP address or interface, then upon completion of the tunnel
## a route is set to the remote net with the IP address or interface
## as source. If no source is given, the IP address of the interface
## on the local subnet is used.
## This route allows processes on the system to use the tunnel.
## To set the source policy route on all connections, put the colon
## as the first character in the string (even before a possible "/").
## Example:
## local_net = loc1 loc2
## remote_net = rem1 rem2
## These following two lines are identical:
##  - loc1 can connect to both rem1 and rem2
##  - loc2 can connnect to rem2
## tunnel = 1a 1b 2b
## tunnel = /2a
## These examples show the usage of the source policy routing:
## tunnel = 1a:eth0 1b:192.168.15.1 2a: 2b
## tunnel = :1a 1b 2b
## tunnel = :/2a
tunnel = :/

##---------------------------------------------------------------------
## Tunnel options
##---------------------------------------------------------------------
## NAT traversal is automatically detected.

## Dead peer detection. The kernel sends echo request packets to
## find out when the peer is no longer available. Three options are
## available:
## dpd_delay: send DPD packets every n seconds (0=off, default: 5)
## dpd_retry: time in seconds until DPD packet is considered failed (5)
## dpd_max:   number of consecutively failed packets until peer is
##            considered dead (default: 5)
## format: dpd = dpd_delay, dpd_retry, dpd_max
## e.g. dpd = 5,5,5
dpd = 5,5,5

## Action to take when the tunnel is found to be dead. Possible values:
## restart: Try to reestablish the tunnel
## clear: clear and unroute the connection; it cannot be reestablished
## hold: hold the connection
## Default: restart
dpdaction = restart

## How many attempts should be made to negotiate a connection, or a
## replacement for one (DPD), before giving up.
## Can be a positive integer value, or 0 for forever.
## Default (if not set): forever
tries = 0

## Action to take if tunnel is properly closed by peer. Possible values:
## restart: try to reestablish the tunnel (not recommended for VPN-GW)
## clear: close connection with no further action (peer may reestablish)
## hold: install trap that tries to reestablish upon traffic
## Default: restart
closeaction = restart

##---------------------------------------------------------------------
## Identification
##---------------------------------------------------------------------
## my_identifier*    = [address|fqdn|asn1dn], [string]
## peers_identifier* = [address|fqdn|asn1dn], [string]
## When using address as identifier, string can either be an
## IP address, or one of ppp0, eth0, vlanX, which will be replaced
## with the IP address of the respective interface.
## use fqdn, NAME for PSK systems, and asn1dn for certificates
## Note:
## When using address or fqdn with certificate based authentication,
## the address or fqdn must be present as subjectAltName in the
certificate,
## otherwise the AnyRover cannot match the certificate to the ID.
## This applies both to our and the peer certificate.
my_identifier = address, ppp0
#my_identifier = fqdn, client.example.org
#my_identifier = asn1dn, C=ch, ST=zh, L=zh, O=AW, OU=AR, CN=srv, E=em@i.l
peers_identifier = address, 192.168.17.42
#peers_identifier = fqdn, server.example.org
#peers_identifier = asn1dn, C=ch, ST=zh, L=zh, O=AW, OU=AR, CN=clt,
E=em@i.l

##---------------------------------------------------------------------
## Authentication
##---------------------------------------------------------------------
## auth_method (use for IKEv1)
## local_auth, remote_auth (for IKEv2)
## Possible values are
## any         (default for remote_auth if not set)
## psk         pre shared key (default for local_auth)
## pubkey      certificates
## cert        synonym for pubkey for IKEv1
## xauth-psk   XAUTH with PSK (only for IKEv1)
## xauth-cert  XAUTH with certificates (use for Cisco VPN clients, IKEv1)
```

```
#auth_method = psk
local_auth = psk
#remote_auth = any

##.....................................................................
## Pre-shared Key
##.....................................................................
## psk = key   pre-shared key, string or hex-number (prefixed with 0x)
psk = mysupersecretkey

##.....................................................................
## XAUTH
##.....................................................................
## xauth: specify role in XAUTH authentication. Possible values are
## server and client.
## This is only relevant if auth_method is xauth-psk or xauth-cert.
#xauth = server

## xauth_id: Username and password for XAUTH authentication.
## This parameter can appear multiple times to define several
## username/password pairs.
## Syntax: xauth_id = username:password
#xauth_id = very:secret

##.....................................................................
## Certificates
##.....................................................................
## the entries reference a [cert] section
## these entries are only evaluated if auth_method=cert
## * the root certificate is given in root
## * the client certificate is given in cert
##   if cert has the syntax:
##   cert = pkcs11:<key_id>, <PKCS#11 user pwd>
##   the certificate is read from the PKCS#11 storage and saved in
##   /etc/ipsec.d/certs/<key_id>.pem
##   No [certificate] section is read in this case.
## * the private key is given in key
## * if private key is in PKCS#11 storage (TPM), give key id
##   and PKCS#11 user password.
##   pkcs11 = file:<filename>, <PKCS#11 user pwd>
##   pkcs11 = cert:<sectname>, <PKCS#11 user pwd>
##   pkcs11 = id:<key_id>, <PKCS#11 user pwd>
##      <filename>: file containing hex ID of PKCS#11 key
##      <sectname>: name of [certificate] section referencing file
##                  containing hex ID of PKCS#11 key
##                  [scep] section with key_type = pkcs11 will create
##                  such a file.
##      <key_id>: hex ID of PKCS#11 key
## * the certificate revocation list is given in crl
## IPsec does not support certificates in .p12 format
#cert = ipsec-cert
#root = ipsec-root
#key  = ipsec-key
#pkcs11 = cert:ipsec-key, abc123
#crl  = ipsec-crl

##.....................................................................
## Security
##.....................................................................
## Security parameters
## (ph1|ph2)_encryption = (aes|twofish|blowfish|3des) [keylen]
## encryption algorithm to use (default: aes) for phase 1 and phase 2
## The desired key length in bit can be given after the algorithm.
## Make sure to use a key length that is supported by the algorithm
## (key length must be a multiple of 8):
## aes, twofish: 128 (default), 192, 256
## blowfish: 40-448 (default: 128)
## 3des: 168 (fix)
## 3des should not be used anymore
ph1_encryption = aes 256
ph2_encryption = aes 256

## (ph1|ph2)_hash_alg = (md5|sha1|sha256|sha384|sha512)
## hash algorithm to use (default: sha256) for phase 1 and phase 2
## md5 and sha1 are not considered secure anymore
ph1_hash_alg = sha256
ph2_hash_alg = sha256

## ph1_prf = (md5|sha1|sha256|sha384|sha512|aesxcbc|aescmac)
## It is possible to explicitely define the PRF algorithm for phase 1.
## If not configured, the same algorithm as for hashing is used.
#ph1_prf = sha1

## ph(1|2)_strict = (yes|no)
## If set to yes (default if unset), then only the algorithms defined
## above will be accepted for the tunnel. If set to no, all supported
## algorithms will be accepted if proposed by peer.
## Only use this for debugging purposes, e.g. if connecting to a peer
## fails with "NO PROPOSAL CHOSEN". Setting *_strict to no will then
## allow the tunnel to come up, so the correct settings can be found.
#ph1_strict = no
#ph2_strict = no

## (ph1|ph2)_lifetime = time VALUE UNIT
## specify life time of the ISAKMP/IPsec SA
## VALUE is a number, UNIT can be one of sec,min,hour
## default values:
##    phase 1: time 24 hour
##    phase 2: time 1 hour
ph1_lifetime = time 24 hour
ph2_lifetime = time 1 hour

## dh_group and pfs_group denote the Diffie-Hellman group for the
## key exchanges. The DH group defines the size of the prime numbers
used.
## The groups must be defined identically on the other end of the tunnel.
```

```
## dh_group is for phase 1, pfs_group for phase 2.
## If you do not use PFS (perfect forward secrecy), just leave pfs_group
## blank, i.e. pfs_group =
## The default values for dh_group is 2, for pfs_group blank
## dh_group  = [1|2|5|14-18]
## pfs_group = [|1|2|5|14-18]
##   ============================================
## || group | group name   || group | group name   ||
## ||-------|--------------||-------|--------------||
## ||    1  |   modp768    ||   21  | ecp521       ||
## ||    2  |   modp1024   ||   22  | modp1024s160 ||
## ||    5  |   modp1536   ||   23  | modp2048s224 ||
## ||   14  |   modp2048   ||   24  | modp2048s256 ||
## ||   15  |   modp3072   ||   25  | ecp192       ||
## ||   16  |   modp4096   ||   26  | ecp224       ||
## ||   17  |   modp6144   ||   27  | ecp224bp     ||
## ||   18  |   modp8192   ||   28  | ecp256bp     ||
## ||   19  |   ecp192     ||   29  | ecp384bp     ||
## ||   20  |   ecp384     ||   30  | ecp512bp     ||
##   ============================================
## groups 1,2,5 are not considered secure anymore.
dh_group  = 5
pfs_group = 5

###====================================================================
### (17a) IPsec Certificates
###====================================================================
## There is no difference between IPsec and OpenVPN certificates.
## The placement in different regions of the config file is solely for
## the convenience of the user. The scripts check all available
## [certificate] sections in the config file and identify the correct
## one based on the name attribute.
## The only restriction is that the appropriate certificate section
## has to be after the reference in the ipsec or openvpn section.
##
## IPsec does not support certificates in .p12 format.
[certificate]
name = ipsec-cert
type = pem
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----

[certificate]
name = ipsec-root
type = pem
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----

[certificate]
name = ipsec-key
type = pem
-----BEGIN RSA PRIVATE KEY-----
-----END RSA PRIVATE KEY-----

[certificate]
name = ipsec-crl
type = pem
-----BEGIN X509 CRL-----
-----END X509 CRL-----

[openvpn]
###########################################################################
### (18) OpenVPN
###########################################################################
## configure openvpn tunnel to server

## start = [yes|no] defines whether to start openvpn at all
## start_server for the OpenVPN server
## start_client for the OpenVPN client
start_server = no
start_client = no

## Some basic configuration options.
## Common: port 1194, proto udp, dev tun, verb 0, (logging)
## for the client: client, explicit-exit-notify
## for the server: client-to-client, client-config-dir, keepalive, dh,
##                 ifconfig-pool-persist, management
## (default: no)
basic_server = yes
basic_client = yes

##---------------------------------------------------------------------
## Server options
##---------------------------------------------------------------------
## server_net defines the net addr and mask of the virtual network
## The OpenVPN server uses the first address of the range for itself,
## and hands the others out to connecting clients
server_net = 192.168.0.0 255.255.255.0

## server_remote_net defines the networks of the peer. A route to
## these networks is defined on the host.
## The list contains network/prefix pairs separated by space.
## server_remote_net = 192.168.2.0/24 192.168.3.0/24
server_remote_net = 192.168.2.0/24

## push_local_net: a list of network/prefix pairs separated by space.
## Routes to these networks are pushed to the client.
push_local_net = 192.168.1.0/24

## push_default: set the default route on the client to the tunnel
push_default   = yes

##---------------------------------------------------------------------
## Client options
##---------------------------------------------------------------------
## remote = SERVER[:port] default port is 1194
```

```
##          SERVER can be a hostname or an IP address
## remote = vpnserver.example.org
## remote = vpnserver.example.org:1194
remote = vpnsrv.example.org

## client_remote_net defines the networks of the peer. A route to
## these nets is defined on the host. (Note: this routes can also be
## pushed by the server, cf. push_local_net above).
## The list contains network/prefix pairs separated by space.
#client_remote_net = 192.168.1.0/24 192.168.0.0/24

##-----------------------------------------------------------------
## Authentication
##-----------------------------------------------------------------
## auth_method = [psk|cert]  pre-shared key or certificates
server_auth_method = cert
client_auth_method = cert

## Use additional key for TLS authentication. This helps to prevent
## DOS attacks that open numerous connections.
## Syntax:
## tls_auth = tlskey[, direction]
##   tlskey references a [certificate] section
##   direction = 0|1. If given, the key must be 2048 bit, and the
##                    peer must have the other value.
##                    This leads to different keys for both directions
#server_tls_auth = server-tls-auth
#client_tls_auth = client-tls-auth

##-----------------------------------------------------------------
## Encryption
##-----------------------------------------------------------------
## Cipher to use. Default (if not specified) is BF-CBC.
## However, this is no longer recommended. For better security,
## use AES-128-CBC.
## Earlier versions of AnyRover used BF-CBC.
## To see all available ciphers, call
## openvpn --show-ciphers
#server_cipher = BF-CBC
server_cipher = AES-128-CBC
#client_cipher = BF-CBC
client_cipher = AES-128-CBC

## Hash authentication algorithm to use. Default is SHA1.
## SHA1 is no longer recommended, for better security, use SHA256.
## To see all available hash algorithms, call
## openvpn --show-digests
server_auth = SHA256
client_auth = SHA256

##.................................................................
## Pre-shared Key
##.................................................................
## OpenVPN pre-shared keys are saved in files and look like
## x509 certificates. They are created with the command
## openvpn --genkey --secret file
## Here, the psk entry refers to a certificate section
server_psk = ovpn-psk
client_psk = ovpn-psk

##.................................................................
## Certificates
##.................................................................
## cert,root,key define the certificates
## - when using one p12 file, place the name into cert
## - otherwise, place the certificate into cert,
##   the root certificate into root and the key into key
## the cert section is mandatory, the other two can be omitted
## if a p12 file is given as certificate
server_cert = server-cert
server_root = server-root
server_key  = server-key
client_cert = client-cert
client_root = client-root
client_key  = client-key

##-----------------------------------------------------------------
## Additional options (server and client)
##-----------------------------------------------------------------
## add additional options to the openvpn config file
## When using BF-CBC, inserting this line is recommended to counter
## SWEET32 attacks: https://community.openvpn.net/openvpn/wiki/SWEET32
## client_option = reneg-bytes 64000000
# server_option = WHATEVER
# client_option = WHATEVER

[clientconfigfile]
###=================================================================
### (18a) Custom client config files
###=================================================================
## With this section, custom client config files can be placed in the
## --client-config-dir directory
## The section takes 1 argument: file. It must be
## present at the head of the section.
## The rest of the section is directly copied to the indicated file.
## Lines in the script cannot begin with '[', as this is interpreted
## as the beginning of the next section.
## Lines in the script can begin with a '#' sign, since after the
## argument line, all lines up to the next section are copied.
## file: name of the file to write. The file is placed under
##       /etc/openvpn/ccd
## To prevent additional lines of the config file (like the EOF mark)
## from appearing in the file, a new section header, e.g. [nofile],
## can be placed there.
#file = client01
#iroute 192.168.33.0 255.255.255.0
```

```
[certificate]
###====================================================================
### (18b) OpenVPN Certificates
###====================================================================
## There is no difference between IPsec and OpenVPN certificates.
## The placement in different regions of the config file is solely for
## the convenience of the user. The scripts check all available
## certificate sections in the config file and identify the correct
## one based on the name attribute.
## The only restriction is that the appropriate certificate section
## has to be after the reference in the ipsec or openvpn section.

## When using a p12 file (binary), the file has to be copied to the
## AnyRover manually, and the path entered into the 'cert' option.
## Alternatively, the certificates can be placed into the config file
## in pem format.
## Currently, it is not possible to use encrypted certificate files.

## To generate the pem files from a p12 file, use these commands:
## openssl pkcs12 -clcerts -nokeys -in file.p12 -out cert.pem
## openssl pkcs12 -cacerts -nokeys -in file.p12 -out root.pem
## openssl pkcs12 -nocerts -nodes  -in file.p12 -out key.pem
## (the -nodes option saves the key unencrypted)
## To generate a p12 file from pem certificates:
## openssl pkcs12 -export -in cert.pem -inkey key.pem \
##                -certfile root.pem -out file.p12

##--------------------------------------------------------------------
name = ovpn-psk
type = pem
-----BEGIN OpenVPN Static key V1-----
-----END OpenVPN Static key V1-----

[certificate]
##--------------------------------------------------------------------
name = server-key
type = pem
-----BEGIN RSA PRIVATE KEY-----
-----END RSA PRIVATE KEY-----

[certificate]
name = server-cert
type = pem
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----

[certificate]
name = server-root
type = pem
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----

[certificate]
name = server-tls-auth
type = pem
-----BEGIN OpenVPN Static key V1-----
-----END OpenVPN Static key V1-----

[certificate]
##--------------------------------------------------------------------
## name identifies the certificate.
name = client-key

## type = [pem|p12|file]
## (file = FILENAME)
## - if type is p12, the file must be specified,
##   and the section referenced as "cert" in the openvpn section
## - if type is pem, the rest of the section is interpreted
##   as a pem file, the file attribute is not necessary in this case
## - if type is file, the parameter file specifies the location of
##   the certificate file to use. This parameter is used if the
##   certificates in pem format are kept outside of the config file,
##   e.g. because they are renewed by some mechanism (e.g. SCEP).
## The certificate is identified by the lines beginning with
## -----BEGIN
## and
## -----END
## everything in the cert file outside these markers can be omitted
type = pem
# file = /etc/openvpn/cert.p12
-----BEGIN RSA PRIVATE KEY-----
-----END RSA PRIVATE KEY-----

[certificate]
name = client-cert
type = pem
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----

[certificate]
name = client-root
type = pem
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----

[certificate]
name = client-tls-auth
type = pem
-----BEGIN OpenVPN Static key V1-----
-----END OpenVPN Static key V1-----

[tunnel]
###########################################################################
### (19) Tunnel
###########################################################################
```

```
## This section is used to configure tunnels.
## Available are: IP in IP tunnel, GRE tunnel and SIT tunnel.
## IPIP: IPv4, no multicast
## GRE: IPv4, multicast
## GRETAP: include ethernet header of tunneled packet
## SIT: IPv6, multicast
## If multiple tunnels are needed, several tunnel sections can be
## defined.
##
## Name: this string is used as the name for the tunnel interface
## Use gre1, gre2, ... for GRE tunnels, tunl1, tunl2, ... for IPIP
tunnels
## gretap1, gretap2, ... for GRETAP tunnels
name = tunnel0

## start: tunnel is only created if start = yes
start = no

## type: defines the type of the tunnel: ipip, gre, sit, gretap
type = gre

## local: IP address or interface of the local tunnel endpoint. The
## remote endpoint is contacted exclusively over this interface.
## If the route to the other endpoint shows through a different
## interface, the peer is not reachable.
local = 192.168.1.3

## remote: IP address of the other tunnel endpoint
remote = 192.168.17.42

## remote_net: networks that are reachable through the tunnel
## several networks are separated by space
remote_net = 192.168.42.0/24

## vlocal: IP address (with netmask) of the virtual tunnel network
vlocal = 10.1.1.1/30

## vremote: IP address of the peer in the virtual network
vremote = 10.1.1.2

[bridge]
########################################################################
### (20) Bridge
########################################################################
## Define a bridge. This section can appear multiple times.
## Rules for bridges:
##   - An interface can only be part of at most one bridge.
##   - If an interface is part of a bridge, it cannot be used
##     directly anymore

## name: will be the name of the bridge interface. This name can be
##       used in the firewall section. It must not collide with some
##       other interface name. Best is to use brX with X = 0,1,2,...
name = br0

## start: set to yes to use this bridge. If set to no, this section
## is ignored.
start = no

## ipaddr: IP address/netmask of the bridge
## The Syntax is identical to the parameter ipaddr in the system section.
ipaddr = 192.168.3.1/24

## iface: space separated list of interfaces to be added to the bridge.
## Possible interfaces are eth0, vlanX, wlan0
iface = vlan2 vlan3

## stp: set to yes to enable spanning tree protocol (STP)
## If set to no, all following parameters are ignored.
stp = no

## prio: priority of the bridge in the spanning tree root negotiations
prio = 32768

## portprio: list of ports and their respective priority
## portprio = vlan2:48 vlan3:99
portprio =

## hello: timer for the STP hello packets
hello = 1

## age: timer for the STP ageing
age = 4

## fw_delay: forward delay timer
fw_delay = 4

## cost: list of ports and their path cost.
## cost = vlan2:45 vlan3:77
cost =

[banner]
########################################################################
### (21) Message of the day
########################################################################
## Message of the day.
## If start = yes, all text between the start attribute and the next
## line starting with '--- END MOTD ---' are placed in the file
## /etc/motd and show upon login, no matter whether this being via
## console, telnet, or ssh.
start = yes
--- END MOTD ---

[daemons]
########################################################################
### (22) User daemons
```

```
#########################################################################
## Define user programs to be started upon boot or shutdown.
## This section can reference a script defined in a script section.
## The script sections are copied to files before this deamon section
## is evaluated.
## Do not start long running commands on shutdown, as the system
## will not wait for them to terminate but proceed shutting down.
## start defines scripts to be run on boot.
## stop defines scripts to be run on shutdown.
## start = /path/to/script/file
## stop = /etc/scripts.d/led.sh clear

[script]
#########################################################################
### (23) User scripts
#########################################################################
## With this section, user scripts can be placed in the system
## The section takes 3 arguments: name, file, mode. They must be
## present at the head of the section.
## The rest of the section is directly copied to the indicated file.
## Lines in the script cannot begin with '[', as this is interpreted
## as the beginning of the next section.
## Lines in the script can begin with a '#' sign, since after the
## 3 argument lines, all lines up to the next section are copied.
## name: currently not used, reserved for later
## file: name of the file to write. If the name starts with a '/',
##       the path is taken absolute, else it is placed under
##       /etc/scripts.d/
##       Non-existing directories are created.
## mode: the file mode of the file in octal notation, e.g. 755.
##       The mode parameter must appear after the file parameter.
##       If mode is "Link:FILENAME", then a symlink from FILENAME to
##       file is created, and the rest of the section is ignored.
##       Example:
##         file = /link/to/file
##         mode = Link:/original/file
## To prevent additional lines of the config file (like the EOF mark)
## from appearing in the script, a new section header, e.g. [noscript],
## can be placed there.

##!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
## HINT: files placed in /etc/scripts.d/ are deleted and recreated
## upon system start. All other files are _not_ deleted automatically,
## especially not if a section is removed from the config file.
## It is thus not recommended to create files with script sections
## outside of the /etc/scripts.d/ directory, as this can have hard
## to find side effects if the configuration is changed.
##!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

#name = myscript
#file = /etc/scripts.d/local/test.sh
#mode = 755

[webserver]
#########################################################################
### (24) Webserver
#########################################################################
## enable the webserver?
start = no

## Port to listen on. Don't forget to open this port in the firewall.
## Default: 80
port = 80

## Interface to listen on. Can be one of eth0, ppp0, vlanX;
## or an IP address.
## If not set or set to all, listen on all interfaces.
## Currently, only one interface is supported. If you need to listen
## on multiple interfaces, you have to leave this empty.
## You can then block access to non-required interfaces with
## corresponding firewall rules.
interface = all

## Set the document root.
## default: /usr/share/www
document_root = /usr/share/www

## Run boa webserver as specified user. If not given, user nobody
## any group nogroup are used.
#user = root
#group = root

## Location of the log files. If no path is given, they are placed in
## /var/log/boa/
## Default: /var/log/boa/access_log and /var/log/boa/error_log
access_log =
error_log =

## The MIME type of files is determined according to file extension.
## For missing or unknown extensions, the type can be specified here.
## Default value: text/plain
## Example: text/html, application/x-httpd-cgi (for cgi scripts)
#default_mime = application/x-httpd-cgi

## Add any further options to webserver config file (boa.conf)
#option = ScriptAlias /cgi-bin/ /etc/scripts.d/local/cgi-bin

## Define whether to enable AnyGator scripts.
anygator = no

[wlan]
#########################################################################
### (25) WLAN
#########################################################################
## start the wlan card?
## If set to yes, make sure to enable power on the USB bus in the
```

```
## usb section.
start = no

## mode: operating mode for the WLAN card:
## ap (access point), client, mesh (IEEE 802.11s)
mode = client

## device defines the network device to run on. For the internal
## wireless LAN card, this is wlan0.
## This parameter can be set to none for a standalone Radius server,
## when running in ap mode.
## If set to none when running in client mode, the configuation files
## will be created, but wpa_supplicant will not be started.
device = wlan0

## Scripts to be run on (dis)connect events must be placed in
## /etc/scripts.d/wlan-ap-hooks/ and /etc/scripts.d/wlan-client-hooks/
## They have 2 or 3 parameters: interface cmd [clientMAC]
## interface defines the WLAN interface the event occurred on, cmd is
## CONNECTED or DISCONNECTED for client interfaces, and
## AP-STA-CONNECTED or AP-STA-DISCONNECTED for AP interfaces.
## On AP interfaces, the MAC address of the client is passed as
## the 3rd parameter.

##---------------------------------------------------------------------
## Common options
## country, channel and ipaddr are for the client, ap and mesh modes,
## all other common options are only for client and ap modes.
##---------------------------------------------------------------------
#
## Country: CH, US, ...
country = ch

## channel: channel number to use.
## For ap and mesh mode, this is the number of the channel to use.
## For client mode, this can be a list of channels to scan, e.g.
## channel = 1,7,13 for 2.4 GHz
## channel = 36,40,44,48,52,56,60,64 for 5 GHz (V2 only), only for
## indoor use.
## If not set, channel 1 is used for AP and mesh, and all for client.
channel =

## Set the ip address of the WLAN interface.
## The syntax is identical to the ipaddr parameter in the system section.
## dhcp does not work if the card is running as access point.
ipaddr = dhcp default nolinklocal noarp

##---------------------------------------------------------------------

## Set the SSID. This can be either an ASCII string, or a hex value.
## Start with 0x if giving a hex value. If the ASCII string starts with
## the characters 0x, enclose in quotes (").
ssid = SSID

## Key management protocol. Possible values are
## WPA-PSK, WPA-EAP, FT-PSK, FT-EAP for AP mode,
## WPA-PSK, WPA-EAP, FT-PSK, FT-EAP, IEEE8021X, NONE for client mode.
## FT-* for Fast BSS Transition (IEEE 802.11r)
## Multiple values can be given separated by space.
key_management = WPA-EAP

## List of accepted pairwise (unicast) ciphers for WPA. Possible values
are
## CCMP, TKIP, WEP104, WEP40 for client mode
## CCMP, TKIP for AP mode.
## Default (if not set) is all.
pairwise = CCMP

## WEP keys. Enter up to 4 WEP keys. The keys can be ASCII text or a hex
## value (starting with 0x).
#wep_key = 0x11223344556677889900112233
#wep_key = 0x12345678901234567890123456
#wep_key = 0x09876543210987654321098765
#wep_key = 0x00998877665544332211009988

## Select the default WEP key. Can be a value from 0 to 3.
## 0 means the first wep key in the config is used as default key.
#wep_default_key = 0

## By default, EAPOL version 2 is applied. But since many APs only
## support version 1, it can be set here.
eapol_version = 1

##---------------------------------------------------------------------
## Client specific options
##---------------------------------------------------------------------

## Scan with SSID-specific frames. This is needed when dealing with
## access points that do not broadcast their SSID.
## Do not enable if not needed, since it will add latency to the
## SSID scanning process.
scan_ssid = no

## Scan for APs when not connected to an AP. Time in seconds.
scan_interval = 30

## Define regular rescans when connected to an AP.
## Note: a scan takes approx 15s, so it will not work if the intervals
## are set to less than these 15s.
## scan_short: interval in seconds when signal level below threshold
## scan_long: interval in seconds when signal level above threshold
## scan_threshold: signal level threshold in dB (negative value)
## Set scan_short to -1 to disable regular rescans.
scan_short = 30
scan_long = 60
scan_threshold = -75
```

```
## If key_management is set to WPA-PSK, the pre-shared key is entered
## here. The key can be ASCII text or a hex value (starting with 0x).
## If the ASCII text starts with the characters 0x, is has to be enclosed
## in quotes (").
pre_shared_key =

## Space-separated list of accepted EAP methods. Possible values are
## MD5, MSCHAPV2, OTP, GTC, TLS, PEAP, TTLS
eap = PEAP

## List of accepted group (broadcast/multicast) ciphers for WPA. Possible
## values are:
## CCMP, TKIP, WEP104, WEP40.
## Default (if not set) is all.
group = CCMP

## Identity string for EAP phase 1 authentication.
#anonymous_identity = anonymous

## Identity string for EAP phase 2 authentication.
identity = userid

## Password string for EAP phase 2 authentication.
password = password

## Root certificate to use for cert based authentication.
## References a [certificate] section.
#root = wlan-root

## Certificate to use for cert based authentication.
## References a [certificate] section.
#cert = wlan-cert

## Key for certificate.
## References a [certificate] section.
#key = wlan-key

## Key for certificate in PKCS#11 storage (TPM). Use instead of key=
## pkcs11 = file:<filename>, <PKCS#11 user pwd>
## pkcs11 = cert:<sectname>, <PKCS#11 user pwd>
## pkcs11 = id:<key_id>, <PKCS#11 user pwd>
##   <filename>: file containing hex ID of PKCS#11 key
##   <sectname>: name of [certificate] section referencing file
##   <key_id>:   hex ID of PKCS#11 key
#pkcs11 = cert:wlan-key, abc123

## Phase 1 (outer authentication, i.e. TLS tunnel) parameters.
## This is a string with field-value pairs, e.g.
## peapver=0
phase1 =

## Phase 2 (inner authentication with TLS tunnel) parameters.
## This is a string with field-value pairs, e.g.
## auth=MSCHAPV2
phase2 = auth=MSCHAPV2

##-------------------------------------------------------------------
## Mesh network specific options
##-------------------------------------------------------------------

## Mesh ID. All stations that want to participate in the mesh
## must have the same ID. The ID is an arbitrary string.
mesh_id = mymeshid

##-------------------------------------------------------------------
## Server (Access point) specific options
##-------------------------------------------------------------------

## WPA: enable WPA:
## wpa, wpa2
wpa = wpa2

## Broadcast SSID
## If set to no, the SSID will not be broadcast.
broadcast_ssid = yes

## Advertise regulatory domain according to IEEE 802.11d?
## Default: no
ieee80211d = yes

## Use IEEE 802.11n
## If set to yes, set hw_mode = g for a 2.4GHz access point
## or hw_mode = a for a 5GHz access point (V2 only)
## Default: no
ieee80211n = no

## hw_mode: operation mode.
## a = IEEE 802.11a, b = IEEE802.11b, g = IEEE802.11g
hw_mode = g

## transmit power
## tx power can be reduced here by up to 20 dBm.
## Default value is 20dBm, the parameter can take values from 0..20.
#txpower = 15

##:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
## MAC address filtering
##:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
## macacl: enable MAC address access list
##   no: disabled, all clients can connect
##   accept: allow client unless MAC address is in deny list
##   deny: deny client unless MAC address is in accept list
#macacl = deny
macacl = no
```

```
## MAC ACL access and deny lists. Use one line for every MAC address.
## acl_accept: Write MAC address in access list. These clients can
##             connect if macacl is set to deny.
## acl_deny: Write MAC address in deny list. These clients cannot
##             connect if macacl is set do accept.
#acl_accept = 00:11:22:33:44:55
#acl_deny = 55:44:33:22:11:00

##::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
## IEEE 802.11n Capabilities
##::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
## High throughput mode (greenfield mode).
## Only enable if no 802.11a/b/g clients are around, otherwise
## the network will not work reliably.
cap_htgf = no

## Support for 40MHz channels.
##
## [HT40-] = both 20 MHz and 40 MHz with secondary channel below
## the primary channel
## available channels: 2.4 GHz: 5-13, 5 GHz (V2 only): 40,48,56,64
##
## [HT40+] = both 20 MHz and 40 MHz with secondary channel above
## the primary channel
## available channels: 2.4 GHz: 1-7, 5 GHz (V2 only): 36,44,52,60
##
## possible values (don't set to use 20 MHz channels):
## cap_40mhz = 40- for [HT40-]
## cap_40mhz = 40+ for [HT40+]
cap_40mhz = 40+

## Support for Short Guard Interval
## Can provide an increase of 11% on data rate at the cost of less
## stable network and more packet collisions.
## Only use if maximum data rate is of utmost importance.
cap_short_gi = no

## Enable multiple receiving channels. Possible values: 1, 2
## Depends on the number of antennas attached to the device.
cap_rx_stbc = 1

## Enable frame aggregation.
## Results in an increased user level data rate.
cap_amsdu = no

##::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

## WPA pre-shared key
## Defines the pre-shared key for key_mgmt=WPA-PSK
## Either define wpa_psk here valid for all clients, or give one
## wpa_psk_entry for every MAC address. wpa_psk_entry can appear
## multiple times. Syntax: wpa_psk_entry = MAC KEY
## The MAC-Address 00:00:00:00:00:00 is for all clients.
## If wpa_psk is set, wpa_psk_entry lines are ignored.
## The PSK can be an ASCII string (8..63 characters) or
## a hex key (64 hex digits) prefixed with 0x
#wpa_psk = secretwlanwpapskpresharedkey
#wpa_psk =
0x0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef
#wpa_psk_entry = 00:11:22:33:44:55 keyforclient1
#wpa_psk_entry = 00:22:44:66:88:aa keyforclient2
#wpa_psk_entry = 01:23:45:67:89:0a
0x0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef
wpa_psk =
#wpa_psk_entry =

## enable 802.1x
ieee8021x = no

##--------------------------------------------------------------------
## Use internal authentication server
##--------------------------------------------------------------------

## Reference to an authentication section to be used as EAP server.
## This references the name attribute of the authentication section.
authentication = eap_server

##--------------------------------------------------------------------
## external Radius Server (Access point) specific options
##--------------------------------------------------------------------

## Use an external Radius server for authentication.
## If enabling this, don't use authentication above.
## Otherwise, it won't work...
use_radius_server = no

## Define IP address to use as source for communication with
## radius server.
## Default: use address according to routing table.
#source_addr = 192.168.1.3

## The IP address of the access point (used as NAS-IP-Address)
## If not given, the system uses the IP address of the WLAN card.
radius_ipaddr = 192.168.59.62

## IP address and port of the Radius server. If port is not given,
## the default port 1812 is used.
## Multiple Radius and Accounting servers can be configured by repeating
## these two statements. They are used if the first one does not reply.
radius_server = 192.168.155.45:1812

## The shared secret used for accessing the Radius server.
radius_secret = thisisverysecret

## IP address and port of the Accounting server. If port is not given,
## the default port 1813 is used.
```

```
radius_accounting = 192.168.201.98:1813

## The shared secret used for accessing the Accounting server.
radius_acct_secret = thisisevenmoresecret

## The interval (in seconds) to try and return to first radius server.
## If set, the system will try to return to the first server even if the
## current server still works.
## If not set, the system will try the given Radius servers consecutively
## and stays with a working server until it fails.
#radius_retry = 600

###====================================================================
### (25a) WLAN client Certificates
###====================================================================
[certificate]
name = wlan-root
type = file
file = /etc/certs/wlan/ca.pem
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----

[certificate]
name = wlan-cert
type = file
file = /etc/certs/wlan/cert.pem
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----

[certificate]
name = wlan-key
type = file
file = /etc/certs/wlan/key.pem
-----BEGIN RSA PRIVATE KEY-----
-----END RSA PRIVATE KEY-----

[authentication]
##########################################################################
### (26) Authentication (EAP/Radius server)
##########################################################################

## Name of the section. This parameter must be first in the section.
name = eap_server

## This section is only evaluated if start is set to yes.
start = no

## Defines whether this authentication server runs as a standalone
## RADIUS server ("yes"), or is referenced from a WLAN section ("no").
standalone = yes

## Username/password pair for EAP phase 1 authentication.
## Syntax: eap_phase1_id = TYPE [user[:password]]
## type can be one of: PEAP TTLS
## If username and/or password are omitted, no checking occurs
## in phase 1 negotiation.
eap_phase1_id = PEAP

## Username/password pair for EAP phase 2 authentication.
## Syntax: eap_phase2_id = TYPE [user[:password]]
## type can be one of: MSCHAPV2 (for PEAP), TTLS-MSCHAPV2 (for TTLS)
eap_phase2_id = MSCHAPV2 fancyuser:verysecretpassword

## CA certificate. This references a [certificate] section.
root_cert = eap_ca_cert

## Server certificate. This references a [certificate] section.
server_cert = eap_server_cert

## Server key. This references a [certificate] section.
server_key = eap_server_key

## Run RADIUS server? This is not needed if this authentication section
## just serves a WLAN AP running on this host.
radius_start = no

## The following attributes are only used if radius_start is set to yes.

## IP address the RADIUS server and RADIUS accounting server listen on.
## If not set: 0.0.0.0
#radius_addr = 192.168.1.3
#radius_acct_addr = 192.168.1.3

## UDP port the RADIUS server and RADIUS accounting server listen on.
## Default is 1812 for RADIUS server and 1813 for accounting server.
radius_port = 1812
radius_acct_port = 1813

## IP addr/network and secret key pairs.
## If several clients share the same password, the client addresses
## can be listed on one line separated by spaces.
## These attributes can appear multiple times for multiple clients.
## The secret always belongs to the last client that was defined
## prior to the secret.
radius_client = IP/prefix
radius_secret = thisisaverysecretsharedsecret

###====================================================================
### (26a) Server Certificates
###====================================================================
[certificate]
name = eap_ca_cert
type = pem
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
```

```
[certificate]
name = eap_server_cert
type = pem
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----

[certificate]
name = eap_server_key
type = pem
-----BEGIN RSA PRIVATE KEY-----
-----END RSA PRIVATE KEY-----

[ospf]
################################################################################
### (27) OSPF
################################################################################

## Start OSPF?
start = no

## OSPF router id. This can be an interface name, an IP address or
## a single number. If an interface has multiple IP addresses, the
## first one is taken (which means that if interface lo is given,
## the router id will be 127.0.0.1)
router-id = vlan1

## Default route. If set to yes, the default route is advertised
## through OSPF.
insert-default = yes

## Area definition. The definition contains the area number and
## a list of interfaces or networks that belong to this area,
## separated by comma.
## This parameter can appear multiple times.
#area = 0, vlan1, vlan2

## Stub area definition. Identical to area definition, but the area
## is marked as stub area.
## This parameter can appear multiple times.
#stub = 1, vlan2, vlan3

## Passive interfaces. The network of these interfaces is advertised
## through OSPF, but the protocol is not run on them.
## This value must be a (list of space separated) interface name(s).
#passive = ppp0

## Authentication options. The authentication options contain area
## number, authentication type, and a list of interface:key tuplets
## or interface:id:key triplets.
## Authentication type can be key or md5
## The key is defined as interface:key for type key, and
## interface:id:key for type md5.
## The id must be consistent across routers on a link.
#auth = 0, key, vlan1:verysecret, vlan2:evenmoresecret
#auth = 1, md5, vlan2:2:unbreakable

## Route summarization. Only valid on ABRs. If networks in an area
## are contiguous, the router can advertise a route summary into other
## areas.
## The definition contains the area id, and a list of summarized networks
## to advertise into other areas.
range = 1, 192.168.64.0/22

[snmp]
################################################################################
### (28) SNMP
################################################################################

## Start SNMP?
start = no

## Listen for SNMP requests
## listen = [proto:][interface/address:][port],...
## Define where to listen for SNMP requests. proto is one of tcp, udp.
## Multiple listen lines are possible to have the daemon listen on
## different interfaces.
## Default is to listen on udp:0.0.0.0:161
listen = udp:161

## SysDescription. Can be an arbitrary string. If not set, value will
## be the output of "uname -mnrsv"
sysdescr = AnyRover v3

## Location information. Can be an arbitrary string.
location = here

## SNMP contact information. Can be an arbitrary string. Usually
## an email address or phone number.
contact = mail@example.com

## List of services on this system.
## Possible values: physical, datalink/subnet, internet, endtoend,
application
## Can also be a number: 1, 2, 4, 8, 64 correspond to the above names
services = physical, datalink, internet

##--------------------------------------------------------------------

## Views
## Used to restrict users to certain subtrees of an OID.
## Views must be defined before the user below. Use second argument
## included to add this OID to the view, exluded to remove it.
## Multiple view lines can be used to add several OIDs to one view.
## view = name, [include|exclude], OID
#view = ping, included, .1.3.6.1.4.1.8072.1.3.2.2.1.3
#view = ping, included, .1.3.6.1.4.1.8072.1.3.2.2.1.7
```

```
## User management
## SNMP version 1 and 2c
## user = SNMP version,{ro|rw},community[,source[,OID|View]]
## SNMP version 3
## user = SNMP version,{ro|rw},user:pw1[:pw2][,hash:enc[,priv[,OID|
View]]]
## A user with read only (ro) or read/write (rw) capabilities is
## created for SNMP version (1, 2 or 3).
## If source is specified, only requests from this source are accepted.
## Source can be an IP address, a hostname, or a net address, e.g.
## 192.168.43.12, mysnmphost, 192.168.67.0/24
## If OID is specified, access is limited to the subtree rooted here.
## Multiple OIDs can be specified using views. Only views defined above
## may be used here.
## If only one password is given, it is used both for authentication and
## privacy, otherwise pw1 is used for authentication, and pw2 for
privacy.
## All passwords must have 8 or more characters.
## hash and enc specify the algorithms used for authentication and
privacy.
## Possible values: hash = MD5|SHA, enc = DES|AES
## Default are MD5 and AES.
## The flag priv may contain the values priv, auth or noauth.
## Setting the flag enforces the use of the requested encryption or
## authentication schemes. If not given, the usage is optional.
#user = 3, ro, user:verysecret:evenmoresecret, SHA:AES
user = 3, ro, admin:adminpass

## Monitor processes
## process = name [, max [, min]]
## Process name must be present between min and max times
#process = gpio_daemon, 2, 2

## Execute arbitrary scripts
## exec = [OID,] name, path [,arg [,arg]]
## sh = [OID,] name, path [,arg [,arg]]
## extend = [OID,] name, path [,arg [,arg]]
## When queried, the program path is executed and its output and status
## returned. If OID is specified, the output will be rooted at this point
## in the OID tree and the full output of the program is returned.
## Otherwise, only the first line of the output is returned in
## the extTable.
## Use exec for binary programs and sh for shell scripts.
## Apart from that, they are identical.
## Extend is an improved form of exec, where the results are returned
## in two tables, once the full output as a single string, and once
## every line separately.
## Extend works for both binaries and shell scripts.
## Use extend unless you have some good reason not to.
#extend = SomeFancyName, /path/to/my/binary, argument1, argument2

##-------------------------------------------------------------------
## Traps
##-------------------------------------------------------------------

## Default community for traps.
trapcommunity = public

## Default username for the trap agent. This must be a valid SNMP v3
## user that the agent uses to poll the information needed to check
## the values.
trapagent = admin

## Destination address for traps (SNMP version 1 and 2)
## trapsink = [tcp|udp:](ip address|hostname)[:port][, community]
## trap2sink = [tcp|udp:](ip address|hostname)[:port][, community]
## Send all traps to this destination. If community is not given,
## the value from trapcommunity is used.
## Default protocol is udp, default port is 162
#trapsink = 192.168.1.1

## Enable or disable sending authentication failure traps.
## This only triggers for SNMP authentication failure, not user login.
authfail = no

## Enable or disable sending interface up/down traps.
## Note: interface status is checked only once per minute, so it can
## take some time after an up/down event until a trap is sent.
updown = no

## Monitor MIB object
## monitor = name, expr [, action [, user [, freq [, oid [, oid]]]]]
## The name must be unique for every monitor.
## expr is of the form: OID | !OID | !=OID | OID OP value | OID min max
## OP can be one of ==, !=, <, <=, >, >=
## action is the name of an action attribute (see below). If action is
## omitted, a notification event is generated (i.e. a trap sent).
## freq defines the interval for checking the expression (default: 600)
## Further oids are appended if action is a notification event
#monitor = Interface UP, ifOperStatus != 2, linkUpTrap, admin, 60

## Action to perform when a monitor triggers
## action = name, type, value [, oid [, oid ]]
## The name is used to identify the action (see monitor above).
## type can be one of: set, notify
## If type is set, value is of the form: oid = value
## If type is notify, value is the notification type, one of:
##   coldStart, warmStart, linkDown, linkUp, authenticationFailure,
##   egpNeighborLoss, enterpriseSpecific
## If set is notify, additional OIDs can be specified that are sent
## in the trap message.
#action = linkUpTrap, notify, linkUp, ifIndex, ifAdminStatus,
ifOperStatus

[dns]
```

```
################################################################################
### (29) DNS
################################################################################

## DNS Proxy
## Start the DNS proxy?
## Make sure to open the necessary port(s) in the firewall section.
## By default, DNS queries are on UDP:53.
start_proxy = yes

## Use some basic properties? If set to yes, some useless Windows
## queries are blocked so they don't generate upstream traffic
## (disable this if you use Kerberos, SIP, XMMP or Google-talk),
## and addresses in the private address space or plain names
## (without dot in the domain part) are not forwarded.
proxy_basic = yes

## List of interfaces to listen on for queries, separated by comma.
## If this is not set, the proxy listens on all local interfaces.
## If the list starts with a '/', it specifies the interfaces
## that are not used.
## Either use this or proxy_address below, but not both.
proxy_interface = /ppp0

## List of IP addresses to listen on, separated by comma.
## Either use this or proxy_interface above, but not both.
#proxy_address = 192.168.1.3

## Port to listen on for DNS queries. If not defined, the default
## port 53 is used.
#proxy_port = 53

## Domain name to append to simple names for DNS lookup.
## Example: if set to example.org, and a client tries to look up
## myhost, then the proxy will send myhost.example.org
## to the name server.
#proxy_domain = localdomain

## More parameters can be put here. Some useful parameters are
## - strict-order: query the name servers strictly in the order they
##   appear in the /etc/resolv.conf file.
## - all-servers: query all servers at the same time. If not set,
##   they are queried one after the other until one answers.
proxy_param = strict-order
#proxy_param = all-servers

## Add static host entries to prevent DNS lookups.
#static_host = google-public-dns-a.google.com, 8.8.8.8

[serports]
################################################################################
### (30) Serports
################################################################################

## enable serports
enable = yes

[openconnect]
################################################################################
### (31) OpenConnect VPN
################################################################################

## OpenConnect is a client for Cisco's AnyConnect SSL VPN.
## OpenConnect is not officially supported by, or associated in any
## way with, Cisco Systems. It just happens to interoperate with
## their equipment.

## Start openconnect?
start = no

## Address of remote server
## Format: https://server.example.org or https://192.168.20.12
remote = https://server.example.org

## Username to connect on the remote server.
username = user

## Password for login on the remote server.
password = verysecret

## openconnect complains and asks for confirmation if it cannot
## verify the server certificate. Setting this parameter to no
## prevents this check.
check_certificate = no

[mobileip]
################################################################################
### (32) Mobile IP
################################################################################
## Abbreviations: HA = home agent, MN = mobile node, HoA = home address

## Start Mobile IP?
start = no

## Mode: mn (mobile node) or ha (home agent, not supported yet)
## foreign agent is not supported.
mode = mn

##--------------------------------------------------------------------
## Addressing
##--------------------------------------------------------------------
## IP Address of the HA. If the HA has multiple IP addresses, they
## can be given separated by comma. The MN will use the first address
## to contact the HA, and the rest to identify the HA from agent
## advertisements (when the MN is at home).
ha = 192.0.2.56
```

```
## IP address of the MN in the home network. Set to 0.0.0.0 to get
## address through AAA infrastructure (not supported yet).
hoa = 10.62.1.23

## List of interfaces over which the MN will not try to contact the HA.
## The interfaces lo, tunl0, and gre0 are ignored by default. To enable
## them, list them with a leading '/'.
## Example:
#ign_interface = wlan0, wlan1
#ign_interface = wlan0, /gre0
ign_interface = eth0, vlan1

## Define what kind of routing will be set up once the tunnel is
## established. Possible values: default, none, {network}.
## default: a default route will be set to the tunnel
## none:    no routing is set up, it must be done using some external
##          scripts, e.g. in /etc/scripts.d/mip-hooks/
## If the value is a network address, then routing to this network
## is set over the tunnel.
## Example:
#routing = default
#routing = 10.0.0.0/8
#routing = none
routing = default

##---------------------------------------------------------------------
## Security parameters
##---------------------------------------------------------------------
## SPI: Security Parameter Index. Defines the security association on
## the HA. Given either in hexadecimal (prefixed with 0x) or decimal.
## Example:
#spi = 0x10a
spi = 266

## Authentication algorithm. Possible values:
## md5-prefix-suffix, hmac-md5, sha1, hmac-sha1
## Do not use md5-prefix-suffix, it has known weaknesses and does
## not work with Cisco HA devices.
auth = hmac-md5

## Shared secret for authentication with the home agent.
## RFC2002 compliant secrets have 16 bytes or 32 hex digits; but
## other lengths are also supported.
## Format: hex number (prefixed with 0x) or string.
## Example:
#secret = ABCDE
#secret = 0x4142434445
secret = AnyRoverSecret

## Replay protection. Possible values: none, timestamp, nonces
replay = timestamp

##---------------------------------------------------------------------
## Tunnel parameters
##---------------------------------------------------------------------
## Tunnel life time: Time until next re-registration in seconds.
## Values >=65535 mean infinite (i.e. never send re-registration).
lifetime = 3600

## UDP port to send registration requests to.
## Default: 434
udpport = 434

## UDP port to use as source in the communication with the HA.
### If not set, a random port is used.
#udpsrcport = 435

## Tunnel keepalives. An active tunnel is probed regularly to check
## availability. This parameter defines the minimum interval between
## keepalive pings (in milliseconds).
interval = 200

## A link is considered to be down after this amount of lost keepalive
## pings.
linkdown = 3

## Initial keepalive round trip time (in milliseconds). The round
## trip time is constantly updated according to current values.
## See parameter percentage.
tunnel_rtt = 500

## Ping timeout: if a reply is not received withing this precentage
## of the average round trip time (tunnel_rtt), it is considered lost.
percentage = 120

##---------------------------------------------------------------------
## Dynamic switching
##---------------------------------------------------------------------
## Link priority: The MN keeps a list of all default routes sorted
## by routing metric.
## If link priority is enabled, it will constantly check all routes
## with lower metric than the currently used, and switch to a
## better one as soon it is available.
## If not set, the MN only changes route if the currently used
## route disappears.
link_priority = yes

## Define how to check higher priority routes. Currently, there are
## three possibilities: ICMP echo request, MIP RegReq valid and invalid
## ICMP echo request sends ICMP echo request messages to the HA,
## while MIP RegReq sends MobileIP registration requests.
## When using valid RegReqs, the tunnel must be switched after the first
## successful message, and the next parameters have no effect.
## When using invalid RegReqs, the id field which contains the current
## time is modified to some point in the past, which causes the HA
```

```
## to respond with "authentication failed" messages. This way, it is
## possible to wait for several failure messages until the tunnel is
## switched.
## If link_prio_icmp is set, then link_prio_regreq_valid is ignored.
## If none of these attributes are set, link_prio_icmp=yes is assumed.
link_prio_icmp = yes
link_prio_reg_valid = no

## This parameter defines the number of successful answers from the HA
## until the MN switches to this route.
link_count = 2

## This parameter defines the interval between consecutive hello
## messages (in seconds). Together with link_count, this defines how
##fast the MN will switch to a better link after it is available.
link_interval = 2

[scep]
############################################################################
###  (33) SCEP
############################################################################

## This section describes the parameters for automatically enrolling
## certificates using SCEP (Simple Certificate Enrollment Protocol).
## Using SCEP, expiring certificates are automatically renewed with
## the SCEP server.
## This section can appear multiple times, to renew several sets of
## certificates

## Hook scripts:
## Upon completion of the SCEP process, a hook script is called
## which then calls all scripts in /etc/scripts.d/scep-hooks/ and
## /etc/scripts.d/scep-hooks/<name>/ where <name> is the value
## of the name parameter in this section.
## In these scripts, several environment variables are set:
##   SCEP_NUMCERT: number of certificates to renew
##   SCEP_SUCCESS: number of certificates that were successfully renewed.
##   SCEP_TIMEOUT: number of certificates where server timeout occurred.
##   SCEP_SKIPPED: number of certificates that are not expiring yet.

## Name of the section. This is used as name of the config file,
## and then passed to the hook scripts.
## This parameter must appear first in the section.
name = ipsec-cert

## Start SCEP client
start = no

## Time table for checking the certificate. This entry will create an
## entry in the cron table and will automatically enable cron daemon,
## even if it is disabled in the [cron] section.
## This parameter can appear multiple times, it will then check at every
## of the specified times.
## The SCEP client contains protection against running multiple times
## at the same time.
## Syntax:
## - same as for cron entries:
#check = 30 21 * * *
## - weekly DAY TIME
#check = weekly thursday 19:00
## - daily TIME
#check = daily 21:30
## - for certain events: on EVENT [arg]
##   possible events:
##      ppp-up:      3G/4G connection established ([ppp] only)
##      mip-up:      MobileIP connected (first connect only)
##      ipsec-up:    IPsec tunnel established
##      dhcp <if>:   interface <if> has obtained an IP-address
##      boot:        after system boot
##      wlan <if>:   wlan <if> has connected (wlan as client)
#check = on ppp-up
#check = on boot

## Actions to take upon successful enrollment.
## Further action can be defined using hook scripts (see above).
##
## This parameter defines fundamental actions. Currently defined:
## - ipsec: reload IPsec connections (all active)
## - bondix: restart bondix tunnels
#action = ipsec

##============================================================================
## global options
##============================================================================

## Debug: write lots of information to log file.
#debug = yes

## Directory where the certificate files are saved.
## It is possible to start the SCEP client with this directory empty.
## The directory is created if it doesn't exist yet.
directory = /etc/certs/ipsec

## Number of days before certificate expiration, when the SCEP client
## is to try and renew the certificate.
days = 14

## Size of private key to generate if no key is present.
## Values: 768, 1024, 2048, 4096
## Use 2048 for PKCS#11 (TPM) based keys.
key_size = 2048

## Type of Private Key to generate:
## rsa: create normal RSA key that is stored in a file
## tpm: create migrateable key stored in TPM (not suitable for IPsec)
## pkcs11: create private key in PKCS#11 store (i.e. the TPM)
```

```
key_type = pkcs11

## Label of key in PKCS#11 store.
## This attribute has no effect if key_type != pkcs11
#label = Production

## Algorithm to use for key signature.
## One of md5, sha1, sha224, sha256, sha384, sha512.
signature = sha256

##====================================================================
## CA options
##====================================================================

## URL to contact on the SCEP server.
## For MS servers, this has the form
## http://<server>/certsrv/mscep/mscep.dll
server = http://172.23.148.199/certsrv/mscep/mscep.dll

## Always fetch CA certificates from server.
## If set to yes, CA certificates are fetched whenever an SCEP request
## ist made. If set to no, CA certificates are only refetched if
## the stored ones are no longer valid or do not match the config.
#fetch_ca = yes

## Encryption used in communication with the SCEP server.
## Possible values: des, 3des, blowfish
encryption = des

## Name of the CA certificate file. A second file with the same name
## but prefixed with enc- is also created.
ca-file = ca-cert.pem

##====================================================================
## certificate options
##====================================================================

## Challenge password, used in communications with the SCEP server.
password = verysecret

## Use current certificate instead of password to authenticate
## with the SCEP server.
old_cert_auth = yes

## Distinguished Name of the CA-certificate.
CA-DN = C=CH, ST=ZH, L=Zurich, O=anyweb, OU=IT, CN=anyca

## Name of the certificate file
cert-file = cert.pem

## Name of the private key file
key-file = key.pem

## Keep CSR file after fetching certificate
keep = no

## Store certificate in PKCS#11 store.
## Only relevant if key_type=pkcs11
store_cert = no

## DN data for the certificate. Allowed parameters:
## Country, State, Location, Organization, OrgUnit, CommonName, Email
Country = CH
State = zh
Location = zurich
Organization = anyweb
OrgUnit = IT
CommonName = AnyRover001
Email = acc@anyweb.ch

## alternative name for certificate.
altname = info@anyweb.ch

## Put quotes (") around altname in CSR?
quote_altname = no

[pelix]
##########################################################################
###  (34) PELIX
##########################################################################
start = no

## Listen for GPRMC messages on this socket
listen = tcp, 127.0.0.1:13181

## IP-Address and Port of PELIX server
## Currently only one target supported.
target = 192.168.1.1:11310

## Source address and port to use when contacting PELIX server.
## If not given, choose according to routing table.
## Syntax:
#source = ipaddr[:port]
#source = 192.168.1.3

## Time in seconds to wait if connecting to server fails until
## the next retry is due.
retry = 5

## Send position message every X seconds
interval = 10

## How to send coordinates to PELIX server:
## CH1903, WGS84 microdegrees, WGS84
coordinates = WGS84 microdegrees
```

```
## ID: usually IMEI
## Has to be entered manually until further notice.
id = 359515050012345

## Login credentials for PELIX server
username = user
password = passwd

## Retransmit un-acked position messages?
retransmit = no

[login]
###############################################################################
###  (36) Login
###############################################################################

## Enable or disable local password login.
## If set to yes, login password is checked against local password file,
## but only after checking against a Radius server (if configured).
## If set to no, make sure to enable another method, otherwise
## logging into the AnyRover is not possible anymore.
## Use passwd_console for console or telnet login,
## passwd_ssh for ssh login, and passwd for both console and ssh.
## passwd[_console|_ssh] = yes|no
#passwd_console = yes
#passwd_ssh = yes
passwd = yes

##=============================================================================
## Radius or TACACS+ server
##=============================================================================

## Blacklist for Radius or TACACS+ Logins.
## All user names listed in the blacklist file are not checked with
## the Radius or TACACS+ server, but only locally.
## This parameter only affects Radius or TACACS+ server configs that
## appear later in the config file.
## Default for blacklist file:
## Radius: /etc/pam.d/rad_blacklist
## TACACS+: /etc/pam.d/tac_blacklist
## Multiple entries are possible, but only the names in the last
## configured blacklist file are checked.
## Note: Files in /etc/pam.d/ are deleted before being written (but only
## once before parsing the first entry).
## Files in other directories are only appended to.
## Files without absolute path are placed in /etc/pam.d/.
## File format: one username per line, lines starting with '#' are
ignored.
## Syntax:
## tacacs_blacklist = [[file]:][user[,user]]
## radius_blacklist = [[file]:][user[,user]]
## Use 'tacacs_blacklist = :' to disable TACACS+ blacklist for further
## TACACS+ server configurations.
## Examples:
#tacacs_blacklist = /etc/pam.d/tac_blacklist: root
#tacacs_blacklist = :
#radius_blacklist = root, config

## Login credentials are sent to Radius or TACACS+ server for checking.
## If server does not respond within timeout interval,
## local passwd file will be queried (if configured to do so, see above).
## Use postfix _console or _ssh to limit to single service.
## Syntax: radius[_console|_ssh] = <SRV>,<KEY>[,<TIMEOUT>]
##         tacacs[_console|_ssh] = <SRV>,<KEY>[,[<TIMEOUT>]
[,SOURCE[:PORT]]]
##                 <SRV> = IP address or hostname of Radius/TACACS+
server
##                 <KEY> = Shared secret for Radius/TACACS+ server
##                 <TIMEOUT> = timeout in seconds, until the next
##                             method is tried.
##                             Default: 3s (Radius), 5s (TACACS+).
##                 <SOURCE> = Source address to use for queries.
##                 <PORT> = Source port to use for queries.
#radius = 192.168.123.1,verysecret,4
#radius_ssh = 192.168.1.1,verysecret,4
#tacacs = 192.168.124.1,anyrover
#tacacs_ssh = 192.168.124.1,anyrover,,192.168.1.3
#tacacs_console = 192.168.124.1,anyrover,3,192.168.1.3:50049

## Allow unknown users
## If this is set to yes, then users that have not yet been registered
## on the system may login when a TACACS+ or RADIUS server authenticate
## the user.
## The user is then created upon login and deleted again when the last
## session terminates.
## If not set, a user must be created on the system before login
## is possible.
## Use postfix _console or _ssh to limit to single service.
#allow_unknown_ssh = yes
#allow_unknown_console = no
#allow_unknown = yes

## Count number of unsuccessful logins and block further tries
## The number of unsuccessful logins on console or ssh are counted
## and the account blocked for a while after reaching limit.
## tally_count enables the feature if set to value >= 0. The account
## is blocked after <n> consecutive login failures. The counter is
## reset on successful login, or with the command pam_tally2.
## tally_root defines whether counting for root user is also active.
## tally_timeout defines the interval to block the account. If not set,
## account is blocked forever (or until reset with pam_tally2).
## Syntax:
## tally_count[_console|_ssh] = <n>
## tally_root[_console|_ssh] = yes|no
## tally_timeout[_console|_ssh] = <t>
#tally_count = 5
```

```
#tally_root = yes
#tally_timeout = 30

[8021x]
###############################################################################
###  (37) IEEE 802.1X Port security
###############################################################################

## This section contains all parameters for IEEE 802.1X port security.
## The section can either define the AnyRover to be an 802.1X supplicant,
## i.e. the AnyRover authenticates with the remote port before starting
## any communications. Or the AnyRover can close its switch ports
## and only accept traffic after a successful 802.1X authentication.

## Name must be the first argument and defines the interface, which
## this section belongs to. E.g. vlan2
## This section has no effect if the parameter "ipaddr" belonging to
## this interface does not contain the keyword 8021x.
name = vlan2

## Mode: supplicant or authenticator.
## Must be the second argument after name.
## Alternatively: client or server.
mode = supplicant

## FIXME: is this actually used?
eapol_version = 1

##============================================================================
## Supplicant
##============================================================================

## Key managment protocol. Possible values: PSK, EAP
key_management = EAP

## List of EAP methods to use (comma separated):
## Possible values: PEAP, TLS, TTLS, MD5, MSCHAPV2
eap = PEAP

## Pre shared key for PSK authentication
pre_shared_key = verysecretkey

## Identity string for EAP phase 1 authentication
#anonymous_identity = anonymous

## Identity string for EAP phase 2 authentication
identity = eapuser

## Password string for EAP phase 2 authentication
password = verysecret

## Phase 1 (outer authentication, i.e. TLS tunnel) parameters.
## This is a string with field-value pairs, e.g.
## peapver=0
phase1 =

## Phase 2 (inner authentication with TLS tunnel) parameters.
## This is a string with field-value pairs, e.g.
## auth=MSCHAPV2
phase2 = auth=MSCHAPV2

##============================================================================
## Certificates (authenticator and supplicant)
##============================================================================

## Root certificate to use for cert based authentication.
## References a [certificate] section.
#root = 8021x-root

## Certificate to use for cert based authentication.
## References a [certificate] section.
#cert = 8021x-cert

## Private key for certificate.
## References a [certificate] section.
#key = 8021x-key

## Key for certificate in PKCS#11 storage (TPM). Use instead of key=
## Only available for supplicant.
## pkcs11 = file:<filename>, <PKCS#11 user pwd>
## pkcs11 = cert:<sectname>, <PKCS#11 user pwd>
## pkcs11 = id:<key_id>, <PKCS#11 user pwd>
##   <filename>: file containing hex ID of PKCS#11 key
##   <sectname>: name of [certificate] section referencing file
##   <key_id>:   hex ID of PKCS#11 key
#pkcs11 = cert:wlan-key, abc123

##============================================================================
## Authenticator
##============================================================================

## Route or bridge EAPOL messages (for authenticator use only)
## If the interface is part of a bridge, the EAPOL packets can be
## received either on the bridge interface or on the physical interface.
## Setting route=yes will route the packets, i.e. they are received on
## the physical interface. If unset or set to no, the packets are bridged
## and thus received on the bridge interface.
## Example:
##   Protecting vlan1 (switch port 1) and vlan2 (port 2) with 802.1x.
##   They are in the same IP subnet, so they are bridged to br0.
##   But the authenticator is run on the vlanX interfaces.
##   In this case, set route=yes, then EAPOL packets will appear on
vlanX,
##   and all other traffic (especially IP traffic) on br0.
#route = yes
```

```
## Define how the port should behave when clients authenticate:
## single-host: Only one host can be authenticated at a time.
##              When a second host authenticates, the first will be
##              deauthenticated (default mode).
## multi-host:  When one client authenticates successfully, the port
##              is open for all possible clients.
##              Needed e.g. if another 802.1x enabled switch is
##              connected to this port.
## multi-auth:  Several hosts can authenticate on this port. Used when
##              a non-802.1x switch is connected.
##              Note: When the cable is unplugged, all hosts are
##              deauthenticated.
port_mode = multi-host

##-------------------------------------------------------------------
## MAC Authentication Bypass
##-------------------------------------------------------------------

## Add MAC address that does not have to authenticate.
## The Radius server is not contacted for this address.
#mab = 00:11:22:33:44:55

##-------------------------------------------------------------------
## internal Radius server
##-------------------------------------------------------------------

## Set to yes to enable internal Radius server, and configure
## users below.
standalone = no

## Username/password pair for EAP phase 1 authentication.
## Syntax: eap_phase1_id = TYPE [user[:password]]
## type can be one of: PEAP TTLS TLS
## If username and/or password are omitted, no checking occurs
## in phase 1 negotiation.
eap_phase1_id = PEAP

## Username/password pair for EAP phase 2 authentication.
## Syntax: eap_phase2_id = TYPE [user[:password]]
## type can be one of: MSCHAPV2 (for PEAP), TTLS-MSCHAPV2 (for TTLS)
eap_phase2_id = MSCHAPV2 fancyuser:verysecretpassword

##-------------------------------------------------------------------
## external Radius server
##-------------------------------------------------------------------

## Define IP address to use as source for communication with
## radius server.
## Default: use address according to routing table
#source_addr = 192.168.1.3

## The IP address of the access point (used as NAS-IP-Address)
## If not given, the system uses the IP address of the WLAN card.
radius_ipaddr = 192.168.1.3

## IP address and port of the Radius server. If port is not given,
## the default port 1812 is used.
## Multiple Radius and Accounting servers can be configured by repeating
## these two statements. They are used if the first one does not reply.
radius_server = 192.168.1.1:1812

## The shared secret used for accessing the Radius server.
radius_secret = thisisverysecret

## IP address and port of the Accounting server. If port is not given,
## the default port 1813 is used.
radius_accounting = 192.168.1.1:1813

## The shared secret used for accessing the Accounting server.
radius_acct_secret = thisisevenmoresecret

## The interval (in seconds) to try and return to first radius server.
## If set, the system will try to return to the first server even if the
## current server still works.
## If not set, the system will try the given Radius servers consecutively
## and stays with a working server until it fails.
#radius_retry = 600

## Additional RADIUS attributes
## Specifiy additional RADIUS attributes that are sent to the RADIUS
server.
## Format: attribute = <attr_id>[:<syntax:value>]
## attr_id: RADIUS attribute type
## syntax: s = string, d = integer, x = octet string
## value: attribute value in format specified by syntax
## If syntax and value are omitted, a null value (0x00) is used.
## Examples (cf. RFC2865):
## attribute = 6:d:2    -> Service-Type = 2 (Framed)
## attribute = 61:d:15  -> NAS-Port-Type = 15 (Ethernet)

###====================================================================
### (37a) 802.1X Certificates
###====================================================================
[certificate]
name = 8021x-root
type = file
file = /etc/certs/8021x/ca.pem
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----

[certificate]
name = 8021x-cert
type = file
file = /etc/certs/8021x/cert.pem
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
```

```
[certificate]
name = 8021x-key
type = file
file = /etc/certs/8021x/key.pem
-----BEGIN RSA PRIVATE KEY-----
-----END RSA PRIVATE KEY-----

[tpm]
################################################################################
###  (38) TPM - Trusted Platform Module
################################################################################

## The TPM can be used to securely store certificates, such that the
## private key never leaves the TPM.
## The TPM is initialized by default, with both owner password and
## SRK password set to abc123.
## Passwords can be changed with tpm_changeownerauth.
## To re-initialize the TPM, see /etc/tpm/HOWTO
## The owner password is needed to take ownership of the TPM, and
## is not stored in the config file.
## The SRK password is needed to access some data when accessing
## stored private keys, so it must be given here.
srk = abc123

## Private Keys are stored in PKCS#11 containers, which is an addon
## to the TPM itself. The PKCS#11 container itself has two additional
## passwords:
##  - SO pin: security officer pin, the Master password
##            This pin is not stored in the config file.
##  - user pin: this is needed to access the stored certificates.
pkcs11 = abc123

[power]
################################################################################
###  (39) Power management
################################################################################

## Set the power governor for CPU frequency scaling.
## The CPU knows 3 different clock speeds: 1GHz, 800MHz, 400MHz
## The governor is responsible to set the current clock speed.
## Syntax:
## governor = <type> [when]
## Possible values for type:
## performance (default): The CPU always runs on 1GHz
## powersave: The CPU always runs on 400MHz. Power reduction: ~10%
## ondemand: CPU frequency is set dynamically according to load.
## userspace: CPU frequency can be set from a userspace application.
##   To set CPU frequency, write frequency value to
##   /sys/devices/system/cpu/cpufreq/policy0/scaling_setspeed
##   Possible values: 996000, 792000, 396000
## Possible values for when:
## preboot (default): set governor as soon as possible during boot
## postboot: set governor only after booting has finished.
##   When setting powersave governor, booting will take longer, so by
##   setting it only after boot, the boot process will stay fast.
## Note: See also "bbu_governor" for governor in BBU mode.
governor = performance

##====================================================================
## Battery Backup Unit (BBU)
##====================================================================

## Start the BBU daemon (default: no)
## Syntax:
## bbu_start = no|yes
bbu_start = no

## Set the power governor for CPU frequency scaling in BBU mode.
## For syntax, see "governor" (default: powersave)
bbu_governor = powersave

## Output voltage of the BBU in V (default: 12 V).
## This voltage must match the voltage of the power supply, that is
## powering the device if the BBU is inactiv.
## Possible values are: [12..24]
## Syntax:
## bbu_voltage = <voltage>
bbu_voltage = 12

## Regulation voltage of the BBU battery in mV (default: 4000 mV)
## Important: Chaning this voltage to a higher value will decrease
##            the batteries lifetime significantly.
## Possible value: [3600..4200]
## Syntax:
## bbu_regulation_voltage = <voltage>
#bbu_regulation_voltage = 4000

## Disable PoE module in BBU mode
## If set to yes, the PoE module will be disabled if the device is
## powered by the BBU and restored if the power resumes.
## Syntax:
## bbu_disable_poeX = no|yes (default: yes)
bbu_disable_poe1 = yes
bbu_disable_poe2 = yes

[bondix]
################################################################################
###  (40) Bondix - Channel bonding
################################################################################

## Enable the channel bonding software from Bondix to use various
## uplinks simultaniously. This function uses the saneclient
## application from bondix and needs a server with the saneserver
## application running to connect to. The saneclient creates a tunnel
## to the saneserver and uses all defined interfaces for the
```

```
## connection. The connection is not encrypted. If encryption is
## wanted, use ipsec over the bondix-connection.

## Start saneclient
start = no

## Use software as client.
## Currently only client mode is supported.
mode = client

## Define logfile directory. All entries of the saneclient.log are
## logged to the anyrover messages logfile as well
logfile = /var/log/saneclient.log

##--------------------------------------------------------------------
## Tunnel options
##--------------------------------------------------------------------

## Define how multiple channels are used
## Possible values are: bonding, failover, loadbalancer
## For bonding, the full license on the server is needed
## Default value is bonding
tunnelmode = bonding

## Define server ip address
#server = 11.12.13.14
server =

## Define port to connect to on server
## Default port is 443
#port =

## Define backup server ip address for redundancy
#backupserver = 11.12.13.15

## Define port to connect to backup server
## Default port is 443
#backupport =

##--------------------------------------------------------------------
## Encryption
##--------------------------------------------------------------------
## The bondix client can encrypt data between client and server.
## If set to yes, ChaCha20 encryption is applied.
use_encryption = no

##--------------------------------------------------------------------
## Authentication
##--------------------------------------------------------------------

## Authentication method to authenticate with the bondix server
## Possible values: password, cert
auth_method = password

##.....................................................................
## Password based auth
##.....................................................................

## Define tunnel name
name = bdx-test

## Set connection password. This password must match to the defined
## password on the server.
password =

##.....................................................................
## Certificates
##.....................................................................
## Note: the tunnel name is set to the CN of the client certificate

## the entries reference a [certificate] section
## these entries are only evaluated if auth_method=cert
## * the root certificate is given in root
## * the client certificate is given in cert
## * the private key is given in key
## Bondix certificates have to be in .pem format.
## The Bondix client does not support TPM based certificates yet.
#root = bondix-root
#cert = bondix-cert
#key  = bondix-key

##--------------------------------------------------------------------
## Channels
##--------------------------------------------------------------------
## Add interfaces to bondix tunnel as channels
## format:
## interfaces = interface1:channelname1, interface2:channelname2, ..
## the number of interfaces is not limited. The interfaces must
## correspond to the existing interfaces on the anyrover, the
## channel name can be defined freely but must not contain white space.
interfaces = wlan0:WLAN, ppp0:SWISSCOM, ppp1:SUNRISE

## Set name of the local bondix tunnel interface
local_iface = bondix0

##--------------------------------------------------------------------
## Bonding Proxy
##--------------------------------------------------------------------

## enable bonding proxy
bonding_proxy = no

## Set bonding interface: proxy listens on this interface for
## TCP traffic to pass through tunnel. This value must be set.
proxy_iface = eth0
```

```
## Set bondig proxy host. The proxy listens on this address for
## incoming traffic.
## This is set additionally to proxy_iface and is optional.
## Probably only needed in special situations, e.g. when the proxy
## interface has multiple IP addresses.
## Default: 0.0.0.0
#proxy_host = 0.0.0.0

## Set bonding proxy port. Do not forget to open the port in the
## firewall section.
## Default: 18080
#proxy_port = 18080

##--------------------------------------------------------------------
## WebGUI
##--------------------------------------------------------------------

## Choose wether to start bondix WebGUI.

## Define interface / IP address and port the WebGUI runs on
## Do not forget to open the port in the firewall.
webgui_start = eth0:80

## If set to yes, the tunnel / channel config can be adjusted
## in the WebGUI.
## Note: after a reboot, all changes made in the WebGUI are lost.
webgui_config = yes

## If set to yes, the WebGUI shows the channel monitors
webgui_monitor = yes

## The key needed to log into the WebGUI.
## If not set, the key "123456" is used.
webgui_key = 123456

[certificate]
###====================================================================
### (32a) Bondix Certificates
###====================================================================

## There is no difference between bondix, ipsec, openvpn, wlan,
## or authenticator certificates.
## The different placement in the config file is solely for the
## convenience of the user.
## It is perfectly fine to use the same certificates and thus link to
## the same [certificate] sections in both the [ipsec] and [bondix]
## sections, or any other section using certificates.
name = bondix-root
type = pem
#file = /etc/certs/bondix/cacert.pem
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----

[certificate]
name = bondix-cert
type = pem
#file = /etc/certs/bondix/cert.pem
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----

[certificate]
name = bondix-key
type = pem
#file = /etc/certs/bondix/key.pem
-----BEGIN RSA PRIVATE KEY-----
-----END RSA PRIVATE KEY-----

[voltage]
###########################################################################
## (41) Voltage monitoring
###########################################################################

## Perform action if input voltage falls below a certain threshold
## Useful if running on battery.

## set to yes to enable voltage monitoring
start = no

## limit: Voltage limit in mV, below which the action is triggered.
## delta: The alarm is reset if the voltage rises above (limit + delta)mV
## Note: the ADC used to measure input voltage has a step size
## of approx. 72mV, and the value is rounded to 10mV.
## delta must be set to a value >=150.
## The AnyRover stops working if the voltage drops below ~8.3V.
## Default values: limit = 11200, delta = 300
limit = 11200
delta = 300

## Number of measurements that must be below limit to trigger
## Since battery voltage is not linear, and the AnyRover is not
## a precise voltage meter, it is not recommended to set this to 0.
## Default value: 5
count = 5

## Interval for checking the input voltage in seconds.
## Default value: 30
interval = 30

## Define whether to check ignition signal before performing action.
## They are only executed if ignition signal has the correct value.
## Possible values:
##    ignore: do not check ignition signal
##    on: action only if ignition is on
##    off: action only if ignition is off
## In case the ignition has the wrong value, only a log message
## is written, but no actions performed.
```

```
## If the AnyRover has to shut down on low battery, this only
## makes sense if ignition is off, because otherwise it will
## just reboot.
## Default value: off
ignition = off

## Actions to perform on low voltage.
## action1 is executed first, with action2 following 2s after
## action1 terminates. Both may be left empty.
## The action may be any of these:
##   LOG: write a log message
##   POWEROFF: shut down AnyRover.
##   <script>: Any script in /etc/scripts.d/
## The command may contain arguments, but these are stripped for the
## predefined actions.
## Example:
#action1 = myscript.sh voltage
action1 = LOG
action2 = POWEROFF

## Action to perform when voltage recovers, i.e. rises above
## (limit + delta)mV
## These actions are not executed if the AnyRover shuts down
## on low voltage.
## Usage similar to actionX above.
recover1 =
recover2 =

##############################################################################
### Mark the end of the file.
### Do not remove this mark.
[EOF]
##############################################################################
### END OF FILE
```

C GNU General Public License

For further information about the GNU licenses see www.gnu.org/licenses

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.,
 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

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

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

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

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

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

  Also, for each author's protection and ours, we want to make certain
that everyone understands that there is no warranty for this free
```

software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

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

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

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

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

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

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

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

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

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

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

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

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

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

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

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

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

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

c) Accompany it with the information you received as to the offer

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

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

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

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

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

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

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

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

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

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

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

of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY
FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED
OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,
INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING
OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER
PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

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

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

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year> <name of author>

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

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

    You should have received a copy of the GNU General Public License along
    with this program; if not, write to the Free Software Foundation, Inc.,
    51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

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

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

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

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

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

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

  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice

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