Create. Connect. Control.

anyweb

Manual

AnyRover V3

Projekt	
Datum	6. März 2023
Status	Public
Version	2.0.20
Autor(en)	Marco Wirz Manuela Bachmann Armin Weiss
Verteiler	

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

Inhaltsverzeichnis

1 Übersicht 5
1.1 Beschreibung 5
1.2 Hardware 5
1.3 Software 5
1.4 Bibliotheken und Tools 6

2 Kurzübersicht oder Handbuch lesen ist feig 7

3 Betrieb des AnyRover – Hardware 8
3.1 Frontseite 8
3.1.1 Antennen 8
3.1.2 USB 8
3.1.3 Konsole 9
3.1.4 Netzwerk 9
3.1.5 LEDs 9
3.1.6 Mode und Reset Button 9
3.2 Rückseite 10
3.2.1 Power 10
3.2.2 GPIO-Stecker 10
3.2.3 DR Inputs (nur an Dead Reckoning GPS Version) 11
3.2.4 Serielle Schnittstellen (optional) 13
3.2.5 DIP Switch Schalter 14
3.2.6 SIM-Karte 14
3.3 Interne Anschlüsse 14
3.3.1 MicroSD-Card 14
3.3.2 Modem 15
3.3.3 PoE 15
3.3.4 Wireless LAN 15
3.4 Fahrzeugeinbau 16
3.4.1 Montageort des AnyRover 16
3.4.2 Montage der Antenne(n) 17
3.4.3 Einbaulage und Rückfahrsignal einstellen (nur bei der 3D Dead Reckoning GPS Variante) 18
3.4.4 Rücksetzen der 3D Dead Reckoning Lagekalibrierung 18
3.4.5 Optionale GPS-Kalibrierungsfahrt (3D Dead Reckoning GPS Variante) 19
3.5 Masszeichnung 22

4 Konfiguration 23
4.1 Systemkonfiguration 23
4.1.1 Änderung der Konfiguration über die Kommandozeile 24
4.1.2 Änderung der Konfiguration über einen Memory Stick 24
4.1.3 Änderung der Konfiguration über SMS 25
4.1.4 Abfragen der Konfiguration über SMS 26
4.1.5 Zurücksetzen der Konfiguration 27
4.1.6 Speichern von Konfigurations-Vorlagen 27
4.2 Variablen 27

4.2.1 Definitionen....27
4.2.2 Laufzeit-Information....28
4.3 Sektionen....28
4.3.1 [system]....30
4.3.2 [switch]....32
4.3.3 [time]....33
4.3.4 [watchdog]....34
4.3.5 [crontab]....34
4.3.6 [gpio]....34
4.3.7 [gps]....36
4.3.8 [sms]....42
4.3.9 [modem]....43
4.3.10 [usb]....45
4.3.11 [dhcp]....45
4.3.12 [dhcprelay]....47
4.3.13 [ftp]....47
4.3.14 [tftp]....48
4.3.15 [firewall]....48
4.3.16 [dyndns]....51
4.3.17 [ppp]....52
4.3.18 [chat_script]....53
4.3.19 [wan]....53
4.3.20 [ipsec]....53
4.3.21 [certificate]....57
4.3.22 [openvpn]....58
4.3.23 [clientconfigfile]....59
4.3.24 [tunnel]....59
4.3.25 [bridge]....60
4.3.26 [banner]....61
4.3.27 [daemons]....61
4.3.28 [script]....61
4.3.29 [webserver]....62
4.3.30 [wlan]....62
4.3.31 [authentication]....67
4.3.32 [ospf]....68
4.3.33 [snmp]....68
4.3.34 [dns]....71
4.3.35 [serports]....71
4.3.36 [openconnect]....72
4.3.37 [mobileip]....72
4.3.38 [scep]....74
4.3.39 [pelix]....77
4.3.40 [login]....78
4.3.41 [802.1X]....79
4.3.42 [tpm]....81
4.3.43 [power]....81
4.3.44 [bondix]....83
4.3.45 [voltage]....85

5 Unterhalt 87
5.1 Lock-Files 87
5.2 Hilfsprogramme 87
5.2.1 Modem Status 87
5.2.2 SMS senden 87
5.2.3 Zentraler Dienst 88
5.2.4 GPIO 88
5.2.5 AD-Wandler 88
5.2.6 Beschleunigungssensoren 88
5.2.7 Datcom 88
5.2.8 PIC-Tool 88
5.2.9 Zertifikate und TPM 89
5.3 Log-Files 90

6 Beispielkonfigurationen 91
6.1 Permanenter IPsec Server mit mehreren Clients und Pre-Shared Key 91
6.2 Permanenter IPsec Server mit mehreren Clients und Zertifikaten 93
6.3 IPsec Tunnel bei Bedarf 94
6.4 2 lokale Subnetze mit NAT 94
6.5 Wireless Client 94
6.6 Wireless Access Point mit DHCP Server 95
6.7 Mehrere Client-Verbindungen über IPsec mit PSK 97
6.8 Dateien über Email versenden 98
6.9 IPsec Server für Cisco VPN Clients 99
6.10 GPO setzen 100
6.11 SCEP Zertifikats-Management 100
6.11.1 Datei basierte Zertifikate 100
6.11.2 TPM basierte Zertifikate 101
6.12 IEEE 802.1X Port Security 101
6.12.1 Authenticator 101
6.12.2 Supplicant 103

A Kontakt 105
A.1 Projektverantwortliche 105
A.1.1 Kommerziell 105
A.1.2 Technische Projektleitung 105
A.1.3 Support und Wartung 105

B Default Konfigurations-Datei 106

C GNU General Public License 149

1 Übersicht

1.1 Beschreibung

Der AnyRover ist ein High Speed 4G Router. Er enthält einen 4-Port Switch, ein LTE-Modem, eine GPS Empfänger sowie einen USB und ein paar General Purpose IO (GPIO) Anschlüsse. Das Gerät kann mit Power over Ethernet (PoE) und Wireless LAN ausgerüstet werden.

Der AnyRover kann über die LTE-Schnittstelle eine Verbindung ins Internet herstellen und diese Verbindung den angeschlossenen Geräten zur Verfügung stellen.

1.2 Hardware

Element	Spezifikation	Mögliche Erweiterung
Prozessor	ARM9, 800MHz Dual Core	ARM9, 800MHz Quad Core
RAM	1GB	
Flash	NAND Flash, 1 GB	
Serielle Konsole	RS-232, 115200 8N1	
Switch	100Mbit, 5 Port (4 extern, 1 intern)	Unterstützt VLANs
Power over Ethernet 802.3af	PSE: Power Sourcing Equipment PD: Powered Device	Total 2 Ports, beide als PSE oder PD
LTE/HSPA/GPRS Modem	Huawei ME909s-120P	2x Huawei ME909s-120P
SIM-Karte	2 externe Slots	
USB Port	USB2.0 HiSpeed	
GPS Empfänger	u-blox 8 Multi-GNNS Receiver	u-blox 8 Multi-GNNS 3D-Dead Reckoning Receiver
SD-Card Slot	Support für SDHC bis 32 GB	
Mehrzweck-Eingänge	3 digitale oder analoge Eingänge mit 10bit ADC	
Mehrzweck-Ausgang	1 digitaler Ausgang, max. 1.8A	
RTC	Mit Stützbatterie	
Wireless LAN	IEEE 802.11 a/b/g/n	2x IEEE 802.11 a/b/g/n
TPM		

1.3 Software

Viele der verwendeten Programme stehen unter der GPL oder der LGPL. Der genaue Text der Lizenzen ist im Anhang abgedruckt. Der Source-Code der entsprechenden

Programme kann bei AnyWeb bezogen werden. Die folgende Tabelle gibt Auskunft über die Programme.

Funktion	Programm	Lizenz	Webseite
Betriebssystem	Linux Kernel 4.10	GPL	www.kernel.org
Kommandozeile	Busybox	GPL	www.busybox.net
SSH Client und Server	Busybox	GPL	www.busybox.net
Telnet Client und Server	Busybox	GPL	www.busybox.net
Firewall	IPtables	GPL	www.netfilter.org
DHCP Server	Busybox	GPL	www.busybox.net
DynDNS Unterstützung	inadyn		
PPP Verbindung	pppd / chat	GPL, BSD, Public Domain	
IPsec	StrongSwan	GPL	www.strongswan.org
OpenVPN	openvpn	GPL	www.openvpn.net
Webserver	boa		www.boa.org
FTP Server	Busybox	GPL	www.busybox.net
TFTP Server und Client	Busybox	GPL	www.busybox.net
NTP Server und Client	ntpd		ntp.isc.org
Cron Jobs	Busybox	GPL	www.busybox.net
SMS Konsole	gpio_daemon	Eigenentwicklung	
Editoren vi und 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	Diverse BSD	www.net-snmp.org

1.4 Bibliotheken und Tools

Verschiedene Bibliotheken und andere Tools sind ebenfalls im Einsatz. Die folgende Tabelle gibt darüber Auskunft.

Bibliothek / Tool	Lizenz	Webseite
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 Kurzübersicht oder Handbuch lesen ist feig

Für die Inbetriebnahme muss der AnyRover am Netzteil angeschlossen werden, die Eingangsspannung liegt im Bereich von 8 – 52V. Eine LTE sowie eine (aktive) GPS Antenne müssen an die entsprechenden Stecker angeschlossen werden.

Mit der Standard-Konfiguration kann eine SIM-Karte von Swisscom (ohne PIN) eingesetzt werden und kurz darauf über die Ethernet-Schnittstelle (mit DHCP) auf das Internet zugegriffen werden.

Der Zugriff auf das Gerät erfolgt über die Konsole mit einem Standard Cisco Konsolenkabel über die serielle Schnittstelle, die Einstellungen sind 115200 8N1. Alternativ kann über das Netzwerk mit ssh (über Ethernet und 3G/4G) und telnet (nur Ethernet) zugegriffen werden.

Das Login erfolgt entweder als User config, Passwort cabtronix oder als User root, Passwort root. Der Config-User kann im Wesentlichen die Konfiguration bearbeiten und den System-Status anschauen (help zeigt die verfügbaren Befehle an). Der root User hat alle Rechte und sollte mit der nötigen Vorsicht agieren.

Die Konfiguration erfolgt über eine zentrale Datei /etc/cablynx.conf. Die Default-Konfiguration ist unter /etc/conf.d/cablynx.factory nochmals abgespeichert. Auf dem Gerät sind 2 Editoren installiert, vi und nano (im Zweifelsfalle nano verwenden!). Die Konfigurationsdatei ist ausführlich kommentiert. Wenn die Konfiguration als Benutzer config bearbeitet wird, werden alle Änderungen nach dem Beenden des Editors automatisch übernommen, bei der Bearbeitung als Benutzer root muss das manuell ausgelöst werden (reboot oder /etc/init.d/rcS config).

Aus Sicherheitsgründen sollten auf dem Gerät, bevor es in den produktiven Einsatz geht, die Passwörter geändert werden (mit dem Befehl passwd sowohl für den Benutzer root als auch config).

Die ausführliche Dokumentation ist auf dem Gerät als PDF Dokument auf deutsch und englische im Verzeichnis /root abgelegt und auch unter www.anyrover.ch zu finden.

3 Betrieb des AnyRover – Hardware

3.1 Frontseite

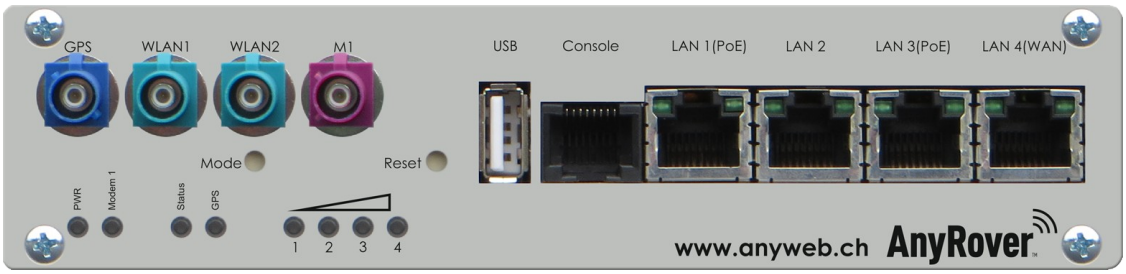


Bild 1: AnyRover V2 Frontseite

3.1.1 Antennen

Der AnyRover besitzt bis zu neun Antennenanschlüsse, welche wahlweise mit SMA-Steckern oder mit Fakra-Steckern ausgerüstet werden können. Der GPS Anschluss benötigt eine aktive GPS-Antenne mit 3V Speisung (für passive Antenne erhältlich auf Anfrage), der GSM-Anschluss eine UMTS und LTE-fähige GSM-Antenne. Für Modem und WLAN können optional je zwei Antennenanschlüsse gewählt und somit MIMO genutzt werden.

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

Eine passende Kombinationsantenne ist zum Beispiel die CT-AT104m von Celphone (www.celphone.ch).

Warnung: Durch Anschliessen einer GPS-Antenne am GSM-Anschluss kann diese zerstört werden.

3.1.2 USB

An der USB-Buchse können beliebige USB 2.0 HiSpeed Geräte angeschlossen werden. Der AnyRover ist standardmässig so konfiguriert, dass Memory Sticks automatisch im System eingebunden werden.

Grundsätzlich ist der Betrieb beliebiger Geräte möglich, sofern ein Treiber für Linux verfügbar ist. Gegebenenfalls muss der Treiber im System selbst installiert und die gewünschte Funktion des Geräts entsprechend implementiert werden.

3.1.3 Konsole

Über den Konsolen-Stecker kann auf die Systemkonsole zugegriffen werden. Die Konsole ist eine RS-232 Schnittstelle, auf die mit einem Cisco-Konsolenkabel zugegriffen werden kann. Die Pin-Belegung der Schnittstelle ist in Tabelle 1: Pin-Belegung des Konsolen-Steckers angegeben. Die Baudrate ist 115200 8N1.

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

Tabelle 1: Pin-Belegung des Konsolen-Steckers

Sicherheitshinweis: Über die Systemkonsole ist ein kompletter Systemzugriff möglich.

3.1.4 Netzwerk

Die vier Netzwerk-Anschlüsse sind gemäss Standardeinstellung gleichwertig. Es ist allerdings möglich, VLANs zu konfigurieren, so dass nicht mehr alle Ports im gleichen logischen Netz sind.

2 Ports (1 und 3) können mit einem PoE Modul ausgerüstet werden, jeweils entweder als PSE (der AnyRover speist ein Peripheriegerät) oder als PD (der AnyRover wird über PoE gespiesen).

3.1.5 LEDs

Die ersten fünf LEDs sind für Power,zwei mal Modem (falls vorhanden), Status und GPS und zeigen so den Status des AnyRovers an. Die vier Level LEDs sind frei konfigurierbar. Standardmässig wird die Signalstärke der Mobilfunknetzes angezeigt.

3.1.6 Mode und Reset Button

Die beiden Buttons auf können über das cablynx.conf beliebig konfiguriert werden. Standardmässig ist der Reset-Button so konfiguriert, dass wenn man kurz drauf drückt (zwischen 2 und 5 Sekunden), die Konfiguration wieder zur Standartkonfiguration zurückgesetzt wird. Wenn man lang drauf drückt (mehr als 5 Sekunden) wird der AnyRover neu gebootet. Der Mode-Button hat in der Standartkonfiguration keine Funktion. Geräte mit MIMO-Antennen-Anschlüssen besitzen keinen Mode-Button.

3.2 Rückseite

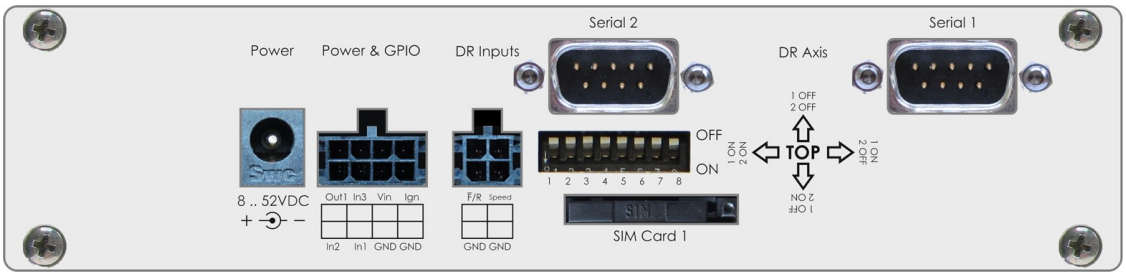


Bild 2: AnyRover Rückseite

3.2.1 Power

Die Speisung des AnyRover erfolgt entweder über den DC Power- oder den GPIO-Anschluss.

Der Power-Stecker benötigt ein handelsübliches Netzgerät mit einem 5.5mm Stecker mit 9.5mm Länge. Der Pluspol ist der Pin in der Mitte (Durchmesser 2.1mm). Die Eingangsspannung kann im Bereich von 8V bis 52V DC liegen. Als Ausgangsleistung wird im Minimum 10W empfohlen für die Version ohne Power over Ethernet. Mit PoE-Verbrauchern sollte die Leistung mindestens 50W betragen und die Spannung mindestens 10V.

Wenn der AnyRover über den Power-Stecker gespiesen wird, verbleibt das Gerät unabhängig vom Zündsignal immer im eingeschalteten Zustand.

3.2.2 GPIO-Stecker

Dieser Anschluss ermöglicht die Stromversorgung des AnyRovers sowie das automatische Ein- und Ausschalten entsprechend des Zündsignals von Fahrzeugen. Die Pin-Belegung ist in Bild 3: GPIO Stecker mit Pin-Belegung angegeben.

Fertige Anschlusskabel sind lieferbar, für spezielle Kabelkonfektionen können folgende Kabelbuchsen verwendet werden:

Hersteller / Serie	Molex / MicroFit 3.0
Kabelbuchsengehäuse 8-polig:	43025-0800
Crimpkontakte:	43030-0009
Crimp-Zange:	63911-2800
Kontakt-Auswerfwerkzeug:	11-03-0043

8	7	6	5
4	3	2	1

Bild 3: GPIO Stecker mit Pin-Belegung

Pin Nummer	Funktion	Klemme nach DIN 72552	Farbe Kabel	Anmerkungen
1	Masse/GND	31	blau	Es können beide oder nur ein Masseanschluss verwendet werden.
2	Masse/GND	31	braun	Es können beide oder nur ein Masseanschluss verwendet werden.
3	Input 1	-	gelb	Digital: Schaltschwellen bei ca. 4.6V (ansteigend) und 2.0V (sinkend). Analog: Messbereich 0 ... 6.6V. Der Eingangswiderstand beträgt 94kΩ.
4	Input 2	-	orange	Digital: Schaltschwellen bei ca. 4.6V (ansteigend) und 2.0V (sinkend). Analog: Messbereich 0 ... 6.6V. Der Eingangswiderstand beträgt 94kΩ.
5	Zündung	15	schwarz	Schaltschwellen: Einschalten bei ca. 2.0V, Detektion Zündung ein/aus bei 4.6V (ansteigend) und 2.0V (sinkend). Der Eingangswiderstand beträgt 20kΩ. Soll AnyRover immer eingeschaltet sein, soll dieser Pin an Vin gelegt werden.
6	Vin 8..52V	30	rot	Standby-Strom: <100µA. Versorgungsstrom 12V / 24V ohne PoE: <460mA / <230mA. Versorgungsstrom 12V / 24V mit 2x PoE: <1.9A / <950mA.
7	Input 3	-	violett	Digital: Schaltschwellen bei ca. 4.6V (ansteigend) und 2.0V (sinkend). Analog: Messbereich 0 ... 6.6V. Der Eingangswiderstand beträgt 94kΩ.
8	Output	-	grün	Schaltet nach Vin. Garantierter Mindeststrom 1.8A.

Tabelle 2: Pin-Belegung des GPIO-Anschlusses

Beim Betrieb in einem Fahrzeug kann sich der AnyRover selbständig ausschalten (Timeout konfigurierbar), wenn die Zündungsleitung (Pin 5) auf Masse liegt (um die Autobatterie zu schonen). Wenn die Spannung an der Zündungsleitung anschliessend wieder >4.6V beträgt, schaltet der AnyRover wieder ein.

3.2.3 DR Inputs (nur an Dead Reckoning GPS Version)

Fertige Anschlusskabel sind erhältlich, für spezielle Kabelkonfektionen können folgende Kabelbuchsen verwendet werden:

Hersteller / Serie:	Molex / MicroFit 3.0
Kabelbuchsengehäuse 4 polig:	43025-0400
Crimpkontakte:	43030-0009
Crimp-Zange:	63811-2800
Kontakt-Auswerfwerkzeug	11-03-0043

Dead Reckoning GPS (auch Koppelnavigation genannt) kombiniert die reine Satellitenortung mit einem Winkelgeschwindigkeits-Sensor (Gyroscope oder kurz Gyro), dem Tacho-Impulssignal und einem Signal, welches Vorwärts- oder Rückwärts-Fahrt

signalisiert. Damit kann die Ortung auch dort fortgesetzt werden, wo kein GPS-Empfang möglich ist, z.B. in Tunnels.

Damit dieses System aber zuverlässig funktioniert, müssen gewisse Voraussetzungen gegeben sein. Die Einbaulage des Gerätes (Siehe dazu in Montageort des AnyRover) sowie die Pegel, Polaritäten und Eigenschaften der angelieferten Signale müssen stimmen. Ist dies nicht gegeben, wird das Resultat unter Umständen schlechter sein als ohne Dead Reckoning.

Am besten funktioniert das System, wenn der Tachoimpuls von der Hinterachse des Fahrzeugs abgenommen wird und die GPS-Empfangsantennen möglichst auch über der Hinterachse montiert wird. Die Quelle des Tachoimpulses kann meistens nicht mit vernünftigem Aufwand verändert werden, bei der Anordnung der Antenne kann aber schon viel erreicht werden. Besonders bei sehr langen Fahrzeugen sollte darauf geachtet werden.

Besondere Vorsicht ist bei Fahrzeugen geboten, welche einen Fahrtenschreiber eingebaut haben. Weil dieser geeicht ist, wird der Tachoimpuls für angeschlossene Systeme oft von diesem abgegriffen oder von einem Fahrtenschreiber-Simulator (zum Beispiel Siemens/VDO "TSU 1391"). Es kann jedoch vorkommen, dass diese Geräte selbst oder daran angeschlossene Systeme auch im Stillstand einzelne Impulse ausgeben, was die Messungen eines Dead Reckoning GPS Systems absolut unbrauchbar macht.

Das Tachosignal muss proportional zur Geschwindigkeit sein – auch bei Stillstand des Fahrzeugs, d.h. es dürfen dann keine Impulse zum AnyRover gelangen. Zudem muss das Tachosignal während des Impulses eine gewisse Spannung erreichen, damit es vom AnyRover erkannt wird. Die Schaltschwellen liegen bei 4.6V (Low-to-High-Übergang) und 2.0V (High-to-Low-Übergang). Am sichersten geht man, wenn man den Impuls direkt vom Impulsgeber abzweigt - wenn dieser die Pegel erreicht, wie sie in Tabelle 3: Pin-Belegung des DR-Anschluss angegeben sind.

Die Pin-Belegung des Steckers ist in Bild 4: DR Anschluss mit Pin-Belegung und Tabelle 3: Pin-Belegung des DR-Anschluss angegeben.

4	**3**
2	**1**

Bild 4: DR Anschluss mit Pin-Belegung

Pin Nummer	Funktion	Klemme nach DIN 72552	Farbe Kabel	Anmerkungen
1	Masse/GND	31	schwarz	Verbunden mit Masse-Pins des GPIO-Anschlusses. Verwendung bei Bedarf.

Pin Nummer	Funktion	Klemme nach DIN 72552	Farbe Kabel	Anmerkungen
2	Masse/GND	31	grün	Verbunden mit Masse-Pins des GPIO-Anschlusses. Verwendung bei Bedarf.
3	Speed Tick / Tachosignal	-	weiss	High-Pegel: +4.75 V bis +30 V Low-Pegel: -30 V bis +1.5V Impulse: 1 Impuls/Meter bis 50 Impulse/Meter Der Eingangswiderstand beträgt mindestens 10kΩ.
4	Vorwärts/ Rückwärts	-	rot	High-Pegel: +4.75 V bis +30 V Low-Pegel: -30 V bis +1.5V

Tabelle 3: Pin-Belegung des DR-Anschluss

Als Rückfahrsignal muss die geschaltete Spannung des Rückfahrscheinwerfers verwendet werden. Dabei entspricht 12V oder 24V der Rückwärts-Fahrt, und 0V der Vorwärts-Fahrt. Liegt das Signal nur invertiert vor, kann dies in der Softwarekonfiguration angepasst werden.

Der AnyRover belastet beide Signale nur mit maximal 2 mA bei 24V- Fahrzeugen und 1mA bei 12V-Fahrzeugen. Es ist uns kein Tachosignalgeber bekannt, der diesen Strom nicht liefern könnte.

Wichtig: Nach dem Einbau muss die GPS-Kalibrierung zurückgesetzt und eine Kalibrierungsfahrt gemacht werden! Siehe dazu in Kapitel Optionale GPS-Kalibrierungsfahrt (3D Dead Reckoning GPS Variante).

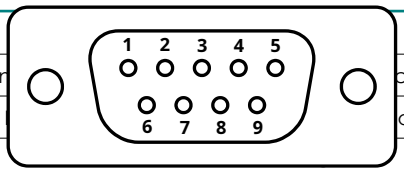
3.2.4 Serielle Schnittstellen (optional)

Serial1 und Serial2 sind serielle Anschlüsse gemäss EIA/RS232. Als normale COM-Ports verwendet unterstützen beide Ports alle gängigen Baudraten bis 115200 Bits/s.

Serial1 ist ohne HW-Handshake ausgeführt und kann direkt die GPS-Daten ausgeben mit der Baudrate 57600 8N1 (Aktivierung über cablynx.conf).

Serial2 weist ein HW-Handshake mittels RTS/CTS auf.

Die folgende Tabelle zeigt die Pin-Belegung der beiden COM-Ports.



Pin	Serial1 Funktion	Serial2 Funktion
1	NC	
2	RxD (Device → Ar	ce → AnyRover)
3	TxD (AnyRover →	over → Device)
4	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

Tabelle 4: Pin-Belegung der COM-Ports

3.2.5 DIP Switch Schalter

Mit Hilfe dieser acht Schalter können verschiedene Konfigurationen rund um die Dead Reckoning vorgenommen werden. Dabei gilt: Schalter oben: OFF, Schalter unten: ON.

Schalter	Funktion	Ergebnis
1 & 2	Konfiguration der Einbaulage des AnyRovers	Siehe Kapitel Fahrzeugeinbau
3	Ausschalten der Dead Reckoning Funktion	OFF: DR ist eingeschaltet ON: DR ist ausgeschaltet
4	Rückwärtssignal invertieren	OFF: high = rückwärts, low = vorwärts ON: high = vorwärts, low = rückwärts
5	DR Puls Filter	OFF: Pulse <100 µm werden heraus gefiltert ON: Pulse <500 µm werden heraus gefiltert
6		
7	Input1 = Speedticks	OFF: Input1 und Speedticks sind getrennt ON: Input1 ist mit Speedticksignal verbunden
8	Input 2 = Vorwärts/Rückwärts-Singal	OFF: Input2 und V/R-Signal sind getrennt ON: Input2 ist mit V/R-Signal verbunden

Tabelle 5: DIP Switch Funktionen

3.2.6 SIM-Karte

Ohne SIM-Karte kann das interne Modem keine Verbindung über das Mobilfunknetz aufnehmen. Es werden auch mit PIN geschützte SIM-Karten unterstützt. Der PIN-Code muss in der Konfigurationsdatei gespeichert werden. Weitere PINs (PIN2, PUK) werden nicht unterstützt. Wenn die Eingabe des PIN-Codes 3 Mal fehlgeschlagen ist, muss die SIM-Karte entnommen und in einem externen Gerät (z.B. Mobiltelefon) wieder entsperrt werden. Eine entsprechende Meldung ist im Logfile zu finden.

3.3 Interne Anschlüsse

Der AnyRover muss in der Regel nicht geöffnet werden, alle benötigten Anschlüsse sind von aussen zugänglich.

Für gewisse Modifikationen am Gerät ist dennoch ein Öffnen des Gehäuses unumgänglich. Vor dem Öffnen muss das Gerät ausgeschaltet und vom Netz getrennt werden. Zum Öffnen wird ein Kreuzschlitz-Schraubenzieher Grösse 1 benötigt.

3.3.1 MicroSD-Card

Der AnyRover kann mit einer MicroSD-Card erweitert werden, damit er mehr

Speicherplatz zur Verfügung hat. Um eine MicroSD-Card einzusetzen oder zu entnehmen, muss die hintere Abdeckung entfernt werden (Power Stecker).

3.3.2 Modem

Der AnyRover kann mit verschiedenen Modem-Typen betrieben werden. Um das Modem auszuwechseln, muss der Deckel entfernt werden. Nach dem Einbau muss das Antennenkabel sauber eingesteckt werden, da sonst das Modem keine Verbindung herstellen kann. Die Steckerverbindung und das Antennenkabel sind sehr empfindlich und müssen mit der nötigen Vorsicht ein und aus gesteckt werden. Das Antennenkabel darf nicht geknickt werden.

Das Modem ist eine Mini PCI Express Karte im normalen Format (30x56mm). Nach einem Wechsel ist keine softwaremässige Änderung nötig, sofern ein unterstütztes Modem eingebaut wird (Huawei ME909s-120P).

Eine allfällige Voice-Fähigkeit wird vom AnyRover nicht unterstützt.

3.3.3 PoE

Um eine Änderung der PoE Konfiguration vorzunehmen, muss der Gerätedeckel entfernt werden. Die maximal zwei PoE Module werden hinter den Ethernet-Buchsen aufgesteckt und mittels zweier Nylon-Distanzhalter befestigt.

Nach einer Änderung an der PoE-Konfiguration ist keine softwaremässige Änderung nötig. Nur um die PoE PSE Module ein- oder auszuschalten kann eine Anpassung der Konfiguration nötig sein.

3.3.4 Wireless LAN

Der AnyRover kann mit maximal zwei Wireless LAN Karten ausgerüstet werden. Die Karten werden über USB angebunden.

Standardmässig kommt eine Karte von DeLock zum Einsatz (www.delock.de). Diese Karte enthält einen Chipsatz von Ralink und unterstützt IEEE 802.11b/g/n und kann sowohl als Client wie auch als Access Point betrieben werden.

Die Karte wird zwischen dem Prozessormodul und der Rückwand eingebaut, zwei Löcher im PCB sind dafür vorgesehen. Die Verbindung erfolgt über ein Kabel zum internen USB Stecker, der gleich hinter dem externen Stecker liegt. Für die WLAN-Antenne ist bereits ein Loch in der Frontplatte vorhanden.

Die Befestigung der WLAN Karte auf der Hauptplatine erfolgt durch passende Distanzbolzen und passende Schrauben. Die Karte muss hoch genug über dem PCB angebracht werden, so dass sich die Bauteile nicht berühren.

Andere WLAN-Karten, welche über USB angeschlossen werden, können ebenfalls zum Einsatz kommen. Allerdings muss dafür eine Lösung zur Befestigung gefunden werden,

und gegebenenfalls der Treiber manuell im System integriert werden (sofern ein anderer Chipsatz als Ralink RT73 oder Ralink RT2800 zum Einsatz kommt).

3.4 Fahrzeugeinbau

Ein sauber geplanter und sorgfältig ausgeführter Fahrzeugeinbau kann viele spätere Probleme verhindern, deshalb sollte diesem Punkt entsprechende Aufmerksamkeit geschenkt werden.

3.4.1 Montageort des AnyRover

Die Wahl des Orts im Fahrzeug, wo der AnyRover eingebaut wird, soll anhand folgender Faktoren bestimmt werden:

Kabellänge der Antennenkabel
Jeder Meter Antennenkabel ist eigentlich einer zu viel. Während bei der aktiven GPS-Antenne nur Verluste auftreten, wenn man mit den Kabeldämpfungen in den Bereich des Gains der Antenne kommt (je nach Kabel und Antennentyp 10 bis 20 Meter), ist bei passiven GPS-Antennen und allen GSM-, UMTS- und WLAN-Antennen jeder zusätzliche Meter ein Verlust von Reichweite. Bei grossen Fahrzeugen wie z.B. einem Reisebus ist die Anbringung des AnyRover im Bereich oberhalb der Fenster und in der Nähe der Antennen deshalb dringend empfohlen.

Vibrationen, Hitze und Schmutz
Alle diese drei Faktoren Verringern die Lebensdauer des AnyRover . Im Fahrzeug können schnell Temperaturen über 70° C auftreten, im Motorraum sogar deutlich mehr.
Bei der Dead Reckoning GPS Variante können starke Vibrationen zudem die Messwerte des Winkelgeschwindigkeitssensors verfälschen. Von einer Montage im Motorraum raten wir aus diesen Gründen ab.

Zugänglichkeit der Signale und Speisungen
Der AnyRover benötigt eine oder zwei Speisungsleitungen (geschaltet und/oder die Batteriespannung), Masse und bei der Dead Reckoning GPS Variante auch noch den Tachoimpuls und das Rückfahrsignal. Die Verlegung der Leitungen kann zeitaufwändig sein. Die Wahl des Montageorts muss auch dies berücksichtigen.

Einbaulage (Standardversion)
Grundsätzlich kann der AnyRover in allen Lagen montiert werden. Es ist aber günstiger, wenn die Anschlussbuchsen des Gerätes nicht nach oben gerichtet sind, weil sich in dieser Lage darin Staub und Schmutz sammeln kann.

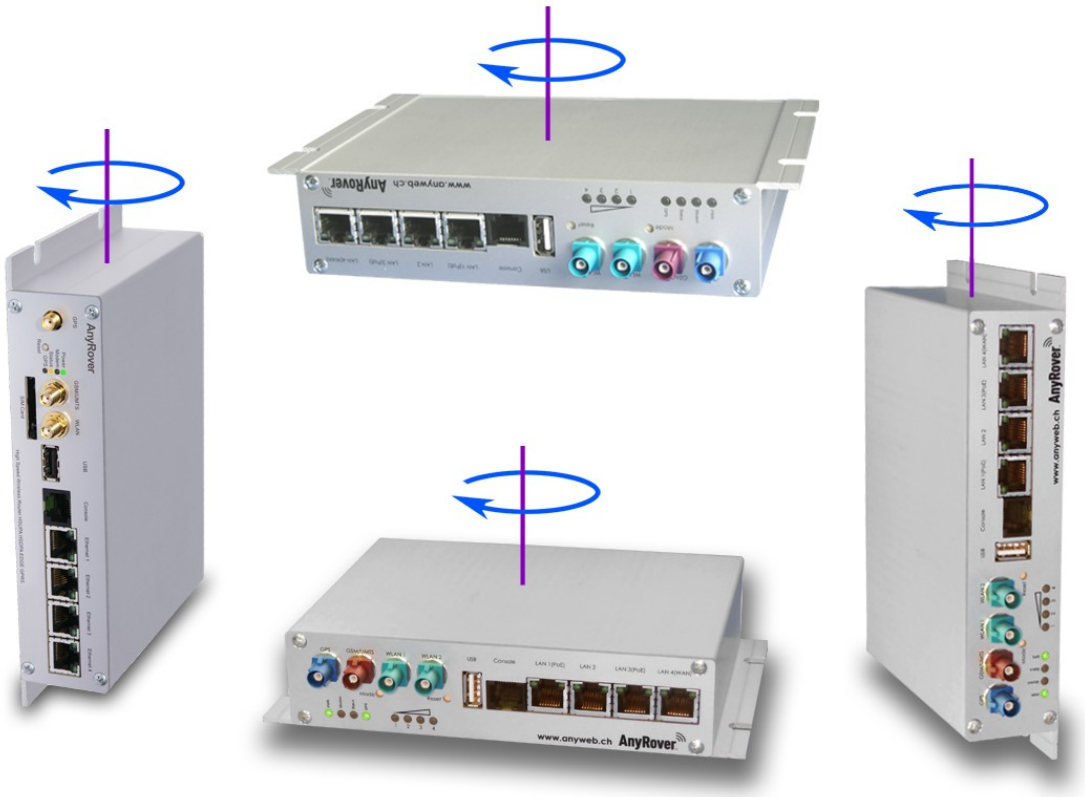


Bild 5: Mögliche Einbaulagen der Dead Reckoning GPS Variante

Einbaulage (3D-Dead Reckoning GPS Variante)
Mit dem neuen 3D-Dead Reckoning GNSS (GPS, GLONASS und Galileo, nachfolgend jedoch nur GPS genannt) gibt es für den AnyRover bezüglich seiner Einbaulage keine Einschränkungen mehr wie bei den Vorgängermodellen. Die höchste Genauigkeit erreicht der AnyRover jedoch dann, wenn eine seiner sechs Seiten parallel zum Erdboden liegt. Der Unterschied zu einer schrägen Einbaulage dürfte jedoch nicht sehr gross sein

3.4.2 Montage der Antenne(n)

Zahlreiche Probleme in Lokalisierungs- und Dispositionssystemen können auf unpassende Einbauorte von GPS-Antennen zurückgeführt werden. Sogenannte "Ausreisser", also Positionsberechnungen, die mehrere hundert Meter bis Kilometer neben der tatsächlichen Position des Fahrzeugs liegen, kommen bei jedem GPS-Empfänger vor und werden vom System weit gehend ausgefiltert. Trotzdem ist guter GPS-Empfang entscheidend für die Genauigkeit und Zuverlässigkeit eines Dispositionssystems, und soll entsprechend gewichtet werden.

Grundsätzlich gilt die Devise, dass Einbauort und Qualität der Antennen einer der entscheidendsten Faktoren überhaupt ist.

Kabel und Anschlüsse
Die Kabel sollen so verlegt werden, dass der kleinste Biegeradius 20mm nicht

unterschreitet. Zudem dürfen die Antennenkabel nirgends scheuern, wenn bewegte Teile im Spiel sind (z.B. Kofferraumdeckel).

Hinweis für die AnyRover-version mit SMA-Antennenbuchsen: Während des Festschraubens den Stecker immer nach schieben, dann läuft die Mutter wesentlich leichter auf dem Gewinde. Das maximale Anzugmoment der Befestigungsmuttern soll 0.2 Nm nicht überschreiten (das entspricht ca. 1.5N oder 150g Druck an einem 12cm-Gabelschlüssel).

Einbauorte

In Bezug auf den GPS-Empfang gilt bei der Wahl des Einbauortes: Je mehr freie "Sicht" zum (gesamten) Himmel, desto besser. Dabei dringt das Signal relativ ungehindert durch (unbeschichtetes) Glas und die meisten Kunststoffe, nicht aber durch Metall. Zudem verhalten sich Wellen im GPS-Frequenzbereich ähnlich wie Licht.

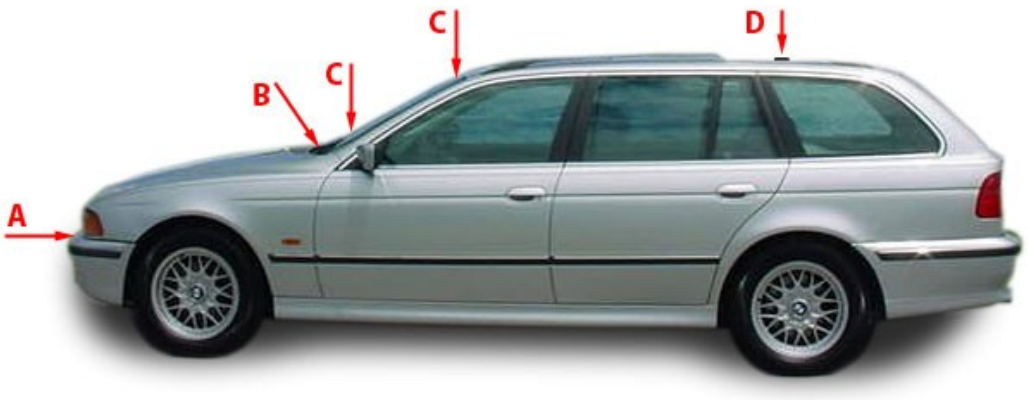


Bild 6: Mögliche Antennen-Montageorte am Beispiel eines Personenwagens

3.4.3 Einbaulage und Rückfahrsignal einstellen (nur bei der 3D Dead Reckoning GPS Variante)

Im Gegensatz zu den Vorgängermodellen muss keine Einbaulage eingestellt oder ermittelt werden – dies passiert beim Fahren automatisch. Auch die Polarität des Rückfahrsignals wird automatisch ermittelt. Abgesehen von der korrekten Zuführung eines Tacho-Signals mit > 5.0 Volt Signalamplitude sowie des Rückfahrsignals entfällt somit jeglicher Aufwand für die Lageneinstellung und eine Kalibrierungsfahrt.

3.4.4 Rücksetzen der 3D Dead Reckoning Lagekalibrierung

Wird ein AnyRover in einem Fahrzeug kalibriert und später in ein anderes Fahrzeug eingebaut, kann dies einige Kilometer oder mehr zu Schwierigkeiten führen – die GPS-Position kann dann um hunderte Meter von der tatsächlichen Position abweichen und dabei seltsame Sprünge machen. Es wird darum dringend empfohlen, das 3D Dead Reckoning GPS des AnyRover vor dem Ausbau zurückzusetzen. Dies geschieht durch Drücken der Reset-Taste für mindestens 10 Sekunden.

3.4.5 Optionale GPS-Kalibrierungsfahrt (3D Dead Reckoning GPS Variante)

Nach dem Einbau muss sich ein 3D Dead Reckoning GPS Empfänger auf seine neue Einbaulage und die Signale des Fahrzeugs einstellen. Dies klappt nur reibungslos, wenn er beim Einbau zurückgesetzt wurde. Eine Kalibrierungsfahrt in Anschluss ist nicht mehr nötig. Hier wird der Vorgang trotzdem beschrieben, um allfällige Fehler besser erkennen und beheben zu können.

Anmerkung: Nach zwölf Minuten Stand im Freien bei eingeschaltetem AnyRover und mit GPS-Empfang (GPS-LED blinkt im Sekundentakt) sind alle Daten zu den aktuellen, genauen Satelliten-Erdumlaufbahnen bekannt. Eine Kalibrierung der Einbaulage, des Tachoimpuls-Signals und des Gyroscopes erfolgt dann schneller und einfacher. Am meisten macht sich dieser Unterschied gleich nach der Erstbestimmung der Position bemerkbar: In den ersten drei Minuten ist eine Kalibrierung kaum möglich.

Anmerkung: In Europa ist die Konstellation der GPS-Satelliten nachmittags nicht optimal. Gerade bei mangelnder Sicht Richtung Süden kann zwischen 14:00 und 17:30 das Finden der GPS-Position mühsam sein. Dies ist nicht wegen mangelnder Signalstärke, sondern weil die wenigen sichtbaren Satelliten oft in einer Reihe stehen und dadurch eine dreidimensionale Positionsbestimmung schwierig ist. Auf einem offenen Platz ist es aber meistens kein Problem.

So sieht der Status im AnyControl aus bei einem nicht kalibrierten 3D Dead Reckoning GPS Empfänger:

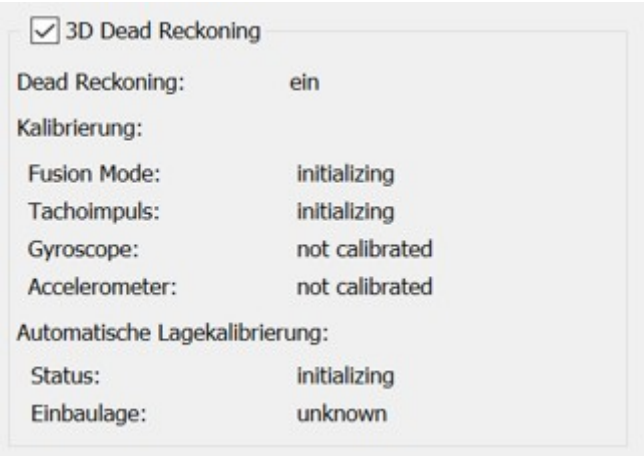


Bild 7: AnyControl 3D DR Status unkalibriert

Zu Beginn versucht der AnyRover, seine Einbaulage zu ermitteln. Am einfachsten gelingt ihm dies, wenn man zuerst geradeaus fährt mit anfahren und abbremsen. Eine anschliessende lang gezogene Kurve mit gleichbleibender Geschwindigkeit in beide

Richtungen hilft ihm, mit der linearen Beschleunigung und der Schwerkraft die Richtungen anhand der Beschleunigungswerte zu bestimmen. Der Hersteller schreibt „Apply some dynamics“, was dem beschriebenen Vorgang entspricht.

Anmerkung: Beachten Sie die zulässigen Höchstgeschwindigkeiten und die herrschenden Bedingungen. Es gilt NICHT „je schneller desto besser“, sondern die Beschleunigen sollen spürbar, aber nicht übertrieben oder gar gefährlich sein!

Nach einigen oben beschriebenen Manövern wechselt die Anzeige unter “Automatische Lagekalibrierung / Status“ auf “coarse“ und die Anzeige bei Tachoimpuls auf „initialized“:

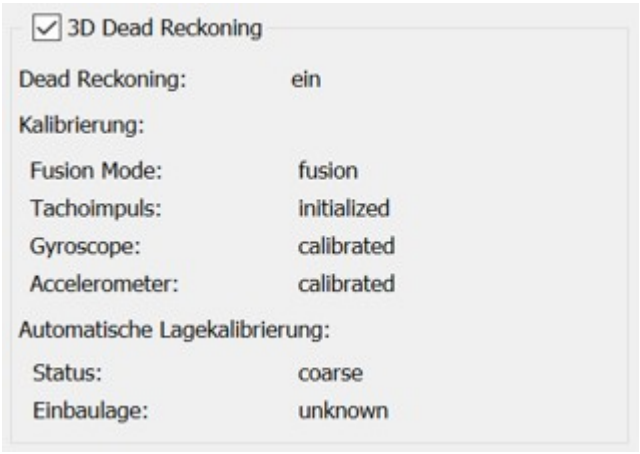


Bild 8: AnyControl 3D DR Status coarse

Die Kalibrierung der Gyroscope-Empfindlichkeit kann erst erfolgen, wenn die grobe (“coarse“) Einbaulage bekannt ist. Ist diese dann abgeschlossen, wechseln “Fusion Mode“ auf “Fusion“ und “Gyroscope“ auf “calibrated“. Dies passiert besonders gut bei einer Fahrt von 500m möglichst geradeaus mit einer anschliessenden Kurve von mindestens 90 Grad, dann wiederum 500m geradeaus. In der Praxis hat sich eine gerade Strasse von ca. 1km Länge bewährt, an deren Ende ein Kreisel eineinhalb-fach umfahren und dann wieder die gleiche Strecke zurückgefahren wurde. Hintergrund: Nur auf einer geraden Strecke kann die gefahrene Himmelsrichtung sehr genau bestimmt werden und entspricht die tatsächlich zurückgelegte Strecke mit der GPS-Strecke überein.

Nach einiger Zeit in Gebrauch (normalerweise ein paar Tage) wechselt die Anzeige “Automatische Lagekalibrierung / Status“ auf “fine“:

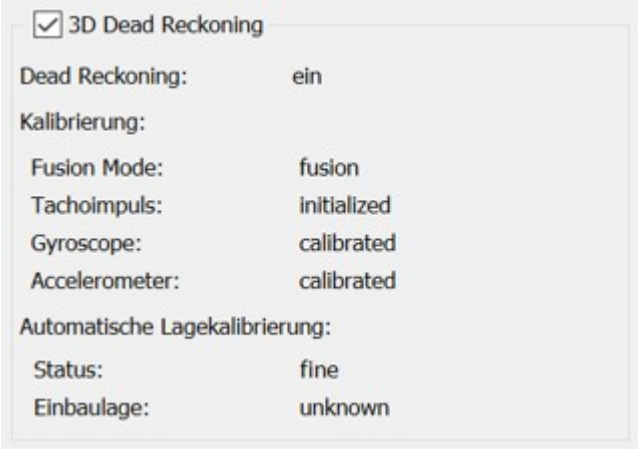
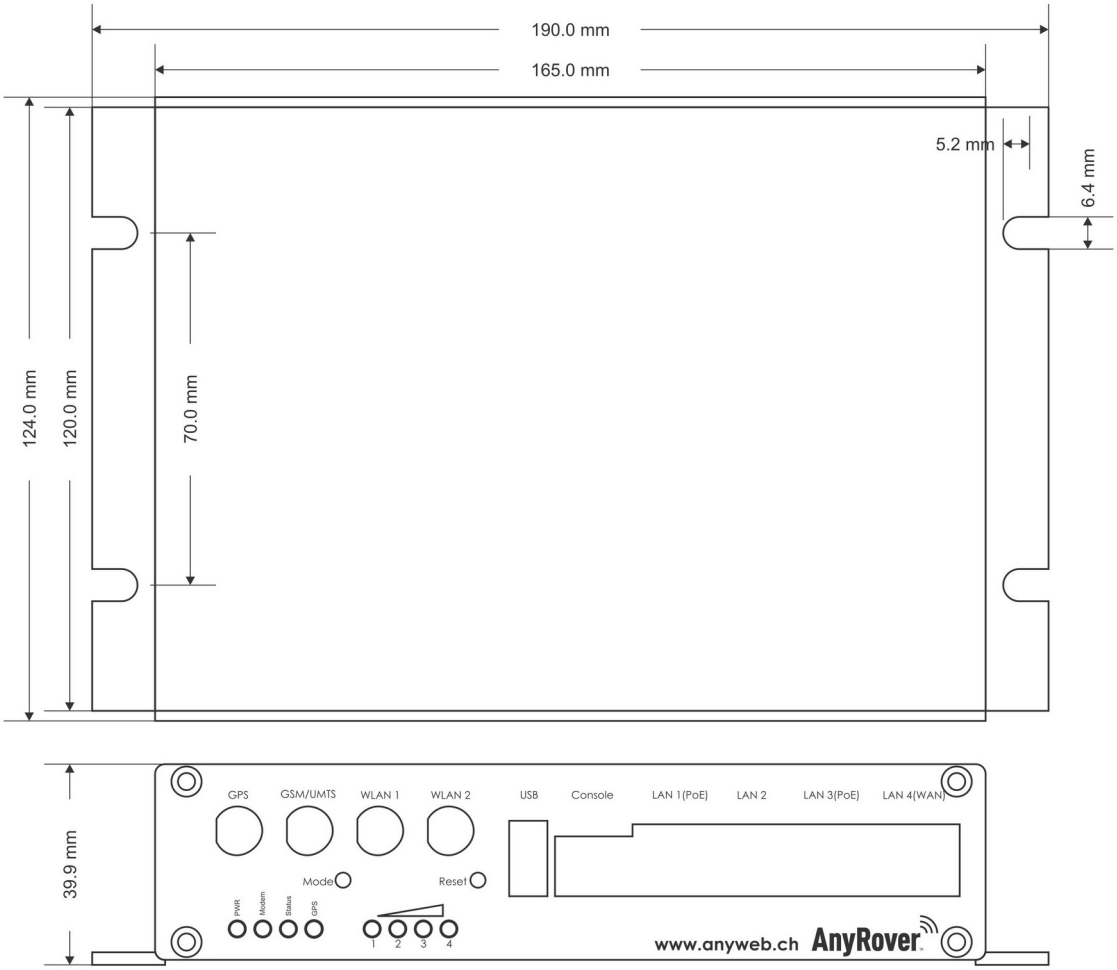


Bild 9: AnyControl 3D DR Status fine

Nun ist das System auf das Fahrzeug geeicht. Die Kalibrierungswerte werden im AnyRover automatisch im Batterie-gestützten Speicher abgelegt, wo sie ohne Stromzufuhr über eine Woche gespeichert werden.

Anmerkung: In Fahrzeugen mit vielen Erschütterungen, oder wenn der AnyRover nicht 100% fix montiert ist (z.B. auf Winkelblech, welches etwas federn kann), ist es möglich, dass ein Wechsel auf “fine“ nie passiert. Trotzdem ist die Funktionalität gewährleistet und die Daten sind ausreichend genau.

3.5 Masszeichnung



4 Konfiguration

Der AnyRover kann über die Kommandozeile konfiguriert werden.

Für die Konfiguration über die Kommandozeile kann über die Konsole oder über das Netzwerk auf den AnyRover zugegriffen werden. Standardmässig ist SSH-Zugriff auf allen Interfaces aktiviert.

Für die Konfiguration existiert ein Benutzer config (Passwort: cabtronix). Dieser Benutzer hat nur wenige Befehle zur Verfügung.

Für weiter gehende Systemanpassungen ist der Benutzer root (Passwort: root) verfügbar. Root-Login über SSH ist möglich. Dateien können mit SCP hin und her kopiert werden.

Sicherheitshinweis: Nach Inbetriebnahme sollten für beide Benutzer die Passwörter geändert werden. Dazu dient der Befehl passwd.

Sicherheitshinweis: Der Benutzer root ist das übliche Linux-Administratorkonto ohne Netz und doppelten Boden. Durch unvorsichtige Bedienung ist es möglich, das System unbrauchbar zu machen. Wer keine Erfahrung mit Linux hat, sollte den Root-Benutzer nicht verwenden (ausser um das Passwort zu ändern).

4.1 Systemkonfiguration

Die gesamte Konfiguration ist in der Datei /etc/cablynx.conf gespeichert. Die Datei ist eine gewöhnliche Textdatei und besteht aus verschiedenen Sektionen. Jede Sektion enthält die Parameter, welche für einen bestimmten Dienst zuständig sind.

Eine Sektion wird durch den Sektionsnamen in eckigen Klammern gestartet und endet, wenn die nächste Sektion beginnt (oder das Dateiende erreicht ist).

Die Konfigurations-Einträge sind einfache Attribut-Wert Paare (AVP – Attribute-value pair) und haben die Form

attribute = value

wobei vor und nach dem Gleichheitszeichen beliebig viele Leerzeichen stehen dürfen. Jedes AVP steht auf einer eigenen Zeile. Leerzeilen zwischen den AVPs sind erlaubt und werden von der Software ignoriert.

In der Konfigurationsdatei können auch Kommentare eingefügt werden. Alle Zeichen

nach einem Doppelkreuz (#) bis zum Zeilenende werden ignoriert. Ausnahme: in der Sektion [chat_script] muss das Doppelkreuz am Zeilenanfang stehen, um als Kommentarzeichen interpretiert zu werden (der Befehl zum Wählen heisst 'atd*99#').

Standardmässig ist die Konfigurationsdatei mit ausführlichen Kommentaren zu allen möglichen Attributen versehen (in Englisch).

4.1.1 Änderung der Konfiguration über die Kommandozeile

Der Benutzer config kann die Konfiguration ändern, indem er den Befehl

```
edit config
```

eingibt. Dadurch wird ein Editor gestartet mit der Konfigurationsdatei. Nach dem Beenden und Speichern wird das System mit der neuen Konfiguration aktualisiert. Wenn der Benutzer über das lokale Netzwerk eingeloggt ist und in der Konfiguration die IP-Adresse des Systems ändert, bricht die Verbindung ab und muss neu aufgebaut werden.

Bei Änderungen an den UMTS Einstellungen wird die aktuelle 3G-Verbindung terminiert und anschliessend neu aufgebaut. Allfällige Verbindungen über die UMTS-Schnittstelle brechen ab.

Auf dem System sind zwei verschiedene Editoren installiert: vi und nano. Der Benutzer config kann mit dem Befehl

```
edit
```

anzeigen, welche Editoren verfügbar sind, und welcher im Moment aktuell benutzt wird.

Hinweis: Wer den Editor vi nicht schon kennt, sollte den nano verwenden. Um vi zu beenden (ohne zu speichern), ist folgender Befehl zu verwenden: :q! (Doppelpunkt – q – Ausrufezeichen – Return).

4.1.2 Änderung der Konfiguration über einen Memory Stick

Es ist möglich, eine Konfiguration mit Hilfe eines Memory Sticks einzuspielen. Dazu muss ein komplettes Konfigurationsfile mit dem Namen cablynx.conf auf dem Stick im Wurzelverzeichnis vorhanden sein. Diese Datei wird mit folgendem Verfahren an die Stelle der bisherigen Konfiguration kopiert.

Wenn beim Booten des AnyRover eine Datei namens /etc/reset existiert, wird ein eingesteckter Memory Stick auf die Datei cablynx.conf untersucht. Ist diese vorhanden, wird sie über die bestehende kopiert und alle Dienste neu gestartet. Der Memory Stick wird sogleich wieder aus dem System entfernt und die Datei /etc/reset gelöscht.

Standardmässig ist in der [gpio] Sektion folgender Befehl definiert:

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

Wenn der Reset-Button mehr als 5 Sekunden lang gehalten wird, bootet der AnyRover neu und sucht nach einer Konfiguration auf einem angeschlossenen Stick.

4.1.3 Änderung der Konfiguration über SMS

Der AnyRover kann über SMS konfiguriert werden. Standardmässig ist der SMS-Zugang nicht freigeschaltet. Drei Einträge in der Konfigurationsdatei definieren den Zugang:

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

Der Zugang kann auf zwei Arten gesperrt werden. Wenn console=no eingetragen ist, kann der Zugang mit einem SMS freigeschaltet werden, wenn der console_key=- ist, kann der Zugang über SMS nicht freigeschaltet werden.

Für die Freischaltung muss ein SMS mit dem Text „eco enable abc123“ mit dem korrekten console_key an den AnyRover geschickt werden. Das Programm ändert darauf den Eintrag auf console=yes, und weitere SMS werden akzeptiert. Mit „eco disable“ wird der Eintrag wieder auf console=no zurückgesetzt. Wird vorher per SMS der console_key auf „-“ gesetzt, kann die SMS-Konsole nach dem nächsten disable nicht mehr freigeschaltet werden.

Die SMS-Konsole versteht folgende Befehle (das SMS muss mit diesen Befehlen in Kleinbuchstaben anfangen):

Befehl	Argumente	Wirkung
eco enable	Passwort	Schaltet die SMS-Konsole ein
eco disable	-	Schlatet die SMS-Konsole aus
eco conf	Section[:name] attr[+|-]=val	Ändert die Konfiguration
eco list	Section1 [section2]	Schickt die gewünschten Sektionen der Konfiguration an den Sender zurück
eco reload		Startet alle Dienste neu und übernimmt die Konfigurations-Änderungen
eco reset		Setzt die Konfiguration auf die Standardconfig zurück
eco templ	Name	Setzt die Konfiguration auf die als Name gespeicherte
eco save	Name	Speichert die aktuelle Konfiguration unter dem angegebenen Namen

Die Syntax des Befehls eco conf ist wie folgt:

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

Ein SMS kann mehrere AVPs (Attribute-Value Pair) enthalten. Vor dem ersten AVP muss eine Sektion angegeben werden, weitere AVPs können mit einem Leerzeichen getrennt angefügt werden. Ebenso können Einträge für weitere Sektionen angegeben werden,

indem der neue Sektionsname eingefügt wird. Der AnyRover interpretiert alle Wörter ohne '=' als Sektionsnamen, und alle Wörter mit '=' als AVP.

Ein AVP darf keine Leerzeichen enthalten. In den Value-Teilen der AVPs werden alle Unterstriche '_' in Leerzeichen umgewandelt (nicht aber in den Attributen).

Über den Zuweisungsoperator zwischen dem Attribut und dem Wert wird die auszuführende Aktion definiert. Der Operator '=' ersetzt das vorhandene AVP mit dem neuen. Falls das AVP nicht existiert, passiert nichts. Neue AVPs können mit dem Operator '+=' hinzugefügt werden, bestehende mit '-=' entfernt.

Wenn ein AVP geändert werden soll und in der Konfiguration mehrere AVPs mit dem gleichen Attribut bestehen (z.B. in der [firewall] Sektion das Attribut accept), wird beim ersten gefundenen AVP mit dem passenden Attribut der Wert ersetzt. Um ein anderes AVP zu ändern, muss dieses zuerst gelöscht und anschliessend mit dem neuen Wert neu eingefügt werden (was allerdings im gleichen SMS geschehen kann).

Neu eingefügte AVPs erscheinen immer am Anfang der Sektion, bei Sektionen mit angegebenem Namen nach dem name Attribut. Wenn also mehrere AVPs eingefügt werden müssen und die Reihenfolge eine Rolle spielt (z.B. bei den Firewall-Regeln), muss das letzte AVP zuerst eingefügt werden.

Wenn ein AVP gelöscht werden soll, müssen sowohl Attribut als auch der Wert genau mit dem Eintrag in der Konfiguration übereinstimmen.

Beispiel: dieses SMS ändert die IP-Adresse auf 192.168.1.3, fügt eine neue Firewall-Regel hinzu (accept = eth0,tcp,80) und löscht eine allfällig bestehende Regel (accept = ,udp,67):

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

Nach einer Änderung der Konfiguration per SMS werden die betroffenen Dienste nicht automatisch neu gestartet (und damit die Änderungen übernommen). Dazu dient der Befehl eco reload.

Sicherheitshinweis: mit diesem Verfahren kann der AnyRover komplett umkonfiguriert werden, allenfalls mit mehreren SMS. Das Feature sollte deshalb mit der nötigen Vorsicht verwendet werden.

4.1.4 Abfragen der Konfiguration über SMS

Mit dem Befehl eco list und der Liste der gewünschten Sektionen kann die Konfiguration ausgelesen werden. Zu beachten ist, dass ein SMS nur bis 160 Bytes lang sein kann; falls das Kommando mehr Ausgabe produziert, werden die ersten 160 Bytes zurückgeschickt und der Rest verworfen.

Wird nun ein SMS in der Form

```
eco list firewall
```

an den AnyRover geschickt, so kommt kurz darauf die entsprechende Sektion ohne Kommentare per SMS zurück. Wenn mehrere Sektionen durch Leerzeichen getrennt spezifiziert werden, liefert der AnyRover alle diese zurück. Wenn keine Sektion angegeben wird, sendet der AnyRover die komplette Konfiguration.

4.1.5 Zurücksetzen der Konfiguration

Mit dem Befehl eco reset wird die Konfiguration auf die Ursprungskonfiguration zurückgesetzt.

Im Verzeichnis /etc/conf.d/ können verschiedene Konfigurations-Vorlagen abgelegt werden, welche dann per SMS geladen werden können. Das entsprechende Kommando lautet

```
eco templ NAME
```

4.1.6 Speichern von Konfigurations-Vorlagen

Konfigurations-Vorlagen können im Verzeichnis /etc/conf.d/ gespeichert werden. Um eine gespeicherte Vorlage zu übernehmen, steht dieser Befehl zur Verfügung:

```
eco save NAME
```

Zu beachten ist, dass die aktuelle Konfiguration überschrieben und nicht gespeichert wird.

4.2 Variablen

Im der Konfigurations-Datei können Variablen verwendet werden, damit Werte, welche mehrfach vorkommen (z.B. IP-Adressen) nur einmal angeben zu müssen.

4.2.1 Definitionen

Variablen werden in einer eigenen Sektion [variables], welche am Anfang der Datei steht, definiert. Da die Konfigurations-Datei zeilenweise verarbeitet wird, können Variablen in allen Zeilen, welche auf die Definition folgen, verwendet werden, aber nicht in Zeilen, welche vor der Definition stehen. Variablen-Definitionen können auch auf bereits definierte Variablen zugreifen.

Variablen werden durch einfache name=value Einträge definiert. Wenn eine Variable verwendet werden soll, muss der Name in doppelten geschweiften Klammern angegeben werden: {{variable}}.

Beispiel:

```
[variables]
ipbase = 192.168.1
gateway = 1
```

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

Variablen können auch verschachtelt werden. Ein Beispiel:

```
[variables]
num = 1
config1 = file_a.conf
config2 = file_b.conf
file = {{config{{num}}}}
# file hat den Wert file_a.conf
```

4.2.2 Laufzeit-Information

Um Laufzeit-Informationen in Variablen einzufügen, kann in der [variables] Sektion mit dem Befehl {{run:VAR}} die Ausgabe eines Konsolen-Befehls eingefügt werden. Der Inhalt der Variable VAR wird dabei in einer Shell ausgeführt, und die erste Zeile der Ausgabe als Wert eingesetzt. Solche {{run:VAR}} Elemente dürfen nicht verschachtelt werden. Ein {{run:VAR}} Element kann allerdings verschachtelte {{variables}} enthalten.

Hinweis: es sollten keine lang laufende Befehle ausgeführt werden, da die Konfigurations-Datei doch recht häufig gelesen wird, und dann jedes Mal eine Zeitverzögerung eingeführt wird.

Beispiel:

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

4.3 Sektionen

Die Sektionen in der Konfigurationsdatei sind in dieser Tabelle ausführlich beschrieben.

Die Reihenfolge der Sektionen ist irrelevant, mit folgenden Ausnahmen:

- [chat_script] muss nach [ppp] stehen
- [certificate] muss nach [ipsec] oder [openvpn] stehen (je nachdem, in welcher Sektion die Zertifikate verwendet werden).

Sektion	Beschreibung

variables	Hier können Variablen definiert werden, welche nachher im Rest des Config-Files verwendet werden können.
system	Definiert die IP-Adresse und Netzmaske des lokalen Netwzerkinterfaces den Hostnamen des Systems sowie statische Routen. Proxy ARP wird ebenfalls hier konfiguriert.
switch	Definiert die Konfiguration des Switches (einschalten, VLANs)
time	Informationen über die Systemzeit und NTP Server
watchdog	Konfiguration des Watchdogs
crontab	Konfiguration der Crontab. Damit können beliebige Programme zu bestimmten Zeiten gestartet werden.
gpio	Definiert die Aktionen, welche ausgeführt werden, wenn an den GPIOs ein Ereignis auftritt. Dazu gehören auch der Reset- und der Mode-Button vorne auf dem AnyRover.
gps	Konfiguration des GPS Daemons. Der Daemon kann die GPS-Daten über UDP oder TCP an andere Hosts schicken (aktiv oder passiv).
sms	Konfiguration des SMS-Zugangs. Hier kann definiert werden, was der AnyRover beim Empfang eines SMS macht.
modem	Hier wird ein allfälliger PIN-Code der SIM-Karte eingegeben.
usb	Schaltet die Versorgungsspannung am USB-Port ein oder aus und definiert, ob Memory Sticks automatisch eingebunden werden.
dhcp	Konfiguration für den DHCP-Server.
ftp	Konfiguration für den FTP Server.
tftp	Konfiguration für den TFTP Server.
firewall	Hier kann die Firewall des AnyRover konfiguriert werden.
dyndns	Ein DynDNS Hostname wird in dieser Sektion konfiguriert.
ppp	Diese Sektion ist für die UMTS-Verbindung zuständig. Login und Passwort für den Zugang werden hier eingetragen.
chat_script	Das chat_script bereitet das Modem für den Verbindungsaufbau vor.
ipsec	Eine IPsec Verbindung zu einem beliebigen Host kann hier eingestellt werden.
certificate	Allfällige Zertifikate für eine IPsec-Verbindung werden in dieser Sektion gespeichert.
openvpn	Hier werden OpenVPN Server und Client konfiguriert.
certificate	Allfällige Zertifikate für OpenVPN werden hier gespeichert.
tunnel	Hier können IP-in-IP und GRE Tunnels konfiguriert werden.
bridge	Hier werden Ethernet Bridges konfiguriert.
banner	Message of the Day. Diese Meldung wird beim Login angezeigt.
daemons	Definiert Benutzer-Programme, welche automatisch gestartet werden.
script	Über diese Sektion können beliebige Skripte in der Konfigurationsdatei abgelegt werden.
webserver	In dieser Sektion wird der Webserver und das WebGUI konfiguriert.
wlan	Konfiguration für eine Wireless LAN Karte, falls vorhanden.
authentication	Definiert EAP und RADIUS Server, z.B. für WLAN.
certificate	Allfällige Zertifikate für den authentication Dienst (EAP-Server).
ospf	Konfiguration für das OSPF Routing Protokoll.
snmp	Konfiguration für SNMP.
serports	Konfiguration für die seriellen Schnittstellen
openconnect	Konfiguration für Cisco AnyConnect VPN.
mobileip	Konfiguration für MobileIP VPN.
scep	Konfiguration für SCEP (Simple Certificate Enrollment Protocol) Zertifikats-Management.

pelix	Konfiguration für die Übermittlung von Positionsdaten an einen Pelix Server.
login	Konfiguration für Login über Radius- oder Tacacs-Server
8021x	Konfiguration für IEEE 802.1X Port Security
tpm	Konfiguration für das Trusted Platform Module
Power	Konfiguration für das Power Managment des AnyRovers
bondix	Konfiguration der Bondix-Software

4.3.1 [system]

Attribut	Wert (Default)	Beschreibung
ipaddr	192.168.1.3/24	IP Adresse und Netzmaske oder Präfix des lokalen Interfaces. Die Adresse kann als 192.168.1.3/24 oder 192.168.1.3 255.255.255.0 angegeben werden. Mit dem Parameter mtu:1492 wird die MTU des Interfaces festgelegt. Für dynamische Konfiguration kann dhcp angegeben werden. Wird "dhcp default" eingetragen, so setzt der DHCP Client auch die Default-Route. Weitere mögliche Parameter nach dhcp sind: metric:M setzt die Metrik der Route auf M (default:0) timeout:T setzt den Timeout auf T Sekunden (default:30) dns: Fragt den DHCP Server nach DNS Server Adressen und ersetzt die bisherigen hostname: Fragt den DHCP Server nach einem Hostnamen, und setzt diesen, falls der bisherige Hostname "localhost" ist. Nolinklocal: Verwende keine Link Local Adressen (169.254.X.Y). noarp: Sende keine ARP Abfrage der Adresse um zu prüfen, ob sie noch frei ist. vendor:V setzt den Vendor Class Identifier String auf V. clientid:I setzt den Client ID String auf I. require:A,B Die Attribute A,B[,…] müssen in der Offer präsent sein, ansonsten wird die Offer ignoriert. Beispiele: bootfile_name, tftp_server_name, ntp_servers, … 8021x: IEEE 802.1X Port Security ist auf diesem Interface aktiviert, die Details werden in einer separaten [8021x] Sektion definiert.
loopback		Diese IP-Adresse wird auf dem Loopback Interface konfiguriert. Dieses Attribut kann mehrfach vorhanden sein.
gateway		IP Adresse des Standard-Gateways. Interfaces, welche als dhcp konfiguriert werden, können selber die Default-Route setzen. Diese Option wird nur benötigt, wenn die Default-Route über ein statisch konfigurierten Interface läuft.
policy		Regeln für Policy-based Routing. policy = SELECTOR ACTION SELECTOR ist einer von from PREFIX, to PREFIX, tos TOS, dev DEVICE und ACTION ist table NUM, prohibit, reject, unreachable Beispiel: policy = from 1.2.3.4 table 200
static_route		Statische Routen Format: [target][/prefix] [netmask] gateway [metric:M] [table:T] [src:S] Wenn prefix und netmask angegeben werden, wird der prefix

		ignoriert; wenn beide fehlen, werden die Standard-Klassen A/B/C verwendet. Wenn das target fehlt, wird die Default-Route gesetzt. Gateway kann entweder eine IP-Adresse (die mit der bisherigen Routing-Table erreichbar sein muss) oder ein Interface (das existieren muss) sein. Der Parameter T stimmt mit der table der Policy-Routen überein. Mit src:S kann die Source Adresse für diese Route gesetzt werden.
proxy_arp		Liste von Interfaces (mit Leerzeichen getrennt), auf welchen Proxy ARP eingeschaltet wird.
hostname	cablynx	Der Hostname des Systems. Wenn der Hostname durch einen DHCP Server gesetzt werden soll, muss hier localhost eingetragen werden.
nameserver		Dieser Parameter definiert einen Nameserver, der vom System verwendet wird. Der Parameter kann mehrfach verwendet werden, um bis zu 3 Nameserver zu konfigurieren. Überzählige Einträge werden ignoriert. Die ersten zwei Einträge werden auch von IPsec verwendet und an Clients ausgegeben, welche einen Mode Config Request machen (z.B. Cisco VPN Clients).
domain		Wird bei DNS Lookups ein Hostname nicht gefunden, wird dieser Domainname angehängt und ein erneuter Lookup ausgeführt.
winsserver		Bis zu zwei WINS Server können hier definiert werden, welche von IPsec an Mode Config Clients ausgegeben werden. Diese Server können nur global für alle Verbindungen gesetzt werden, nicht für jede IPsec-Verbindung einzeln.
log_server	192.168.3.2	Wenn ein Server angegeben ist, werden alle Log-Meldungen auch an diesen Server geschickt. Zum Ausschalten Zeile auskommentieren (default).
loc_level		Alle Log-Meldungen mit einem Level kleiner als dem angegebenen Wert werden ins Log-File /var/log/messages geschrieben. Standartmässig wird alles geloggt.
log_file	/var/log/messages	Name des Log-Files. Das Default-File liegt auf einer RAM-Disk und geht bei jedem Reboot verloren. Mit diesem Parameter kann eine andere Datei angegeben werden. Bei Angabe nur eines Dateinamens wird die Datei im Verzeichnis /var/log angelegt. Wenn der angegebene Pfad nicht existiert, wird er angelegt.
log_rotate_size	200	Das Log-File wird automatisch rotiert, sobald es eine gewisse Grösse erreicht hat. Mit diesem Parameter wird diese Grösse in KB definiert.
log_rotate_files	1	Wenn das Log-File rotiert wird, werden ältere Dateien gelöscht. Dieser Parameter gibt an, wieviele ältere rotierte Dateien erhalten werden.
tftp_server	192.168.3.2	Der Befehl *update system* benützt diesen Wert als Vorschlag, bzw. u*pdate system quiet* sucht auf diesem tftp server nach dem Update.
partition		Definiert weitere Partitionen, welche im System eingebunden werden können. Syntax: partition = device, filesystem, mountpoint [, option [,option]] Der mountpoint wird erstellt, falls er noch nicht existiert. Die Optionen sind die bekannten Optionen aus /etc/fstab. Die Option noatime ist stets gesetzt. Mögliche Optionen: rw (default), ro, [no]exec, ... Beispiel:

	partition = /dev/mtdblock4, jffs2, /media/log, noexec

4.3.2 [switch]

Attribut	Wert (Default)	Beschreibung
start	yes	Der Switch wird eingeschaltet, wenn dieser Wert "yes" ist. Ansonsten bleibt der Switch ausgeschaltet, und der Zugriff übers Netzwerk ist nicht möglich. Im Bootvorgang wird der Switch erst eingeschaltet, wenn die Firewall aktiv ist.
poe1	no	Definiert, ob das PoE PSE Modul auf dem Ethernet Port 1 eingeschaltet wird. Hat keine Wirkung für ein PD Modul.
poe2	no	Definiert, ob das PoE PSE Modul auf dem Ethernet Port 3 eingeschaltet wird. Hat keine Wirkung für ein PD Modul.
ports	4,4,4,4	Definiert die Switch-Ports, jeder Port wird mit einem Wert zwischen 0 und 4 konfiguriert; die Werte sind mit Komma getrennt. Fehlende Werte werden als 4 angenommen. Die Werte bedeuten: 0: 10 Mbit, half duplex 1: 10 Mbit, full duplex 2: 100 Mbit, half duplex 3: 100 Mbit, full duplex 4: auto negotiation
port_disable		Liste von Switch-Ports (mit Komma getrennt, Port 1 bis 4), welche ausgeschaltet werden sollen. Dies funktioniert nur, wenn start_vlan auf yes gesetzt ist. Beispiel: port_disable = 3, 4
start_vlan	no	VLANs werden nur aktiviert, wenn sie hier eingeschaltet werden. Dies bewirkt, dass die IP-Adresse aus der [system] Sektion dem VLAN 0 zugewiesen wird. Im VLAN 0 sind alle Ports, die nicht explizit in einem anderen VLAN zugewiesen werden.
vlanX	-	In diesem Parameter werden die Ethernet Ports aufgezählt, welche zum VLAN gehören, getrennt durch Kommas. X kann einen Wert von 1 bis 4 annehmen. vlan1 = 1,2
ipaddrX	-	Dieser Parameter definiert die IP-Adresse und die Netzmaske des Netzwerk-Interfaces, welches auf dem AnyRover angelegt wird im VLAN X. ipaddr1 = 192.168.1.1/24 ipaddr2 = 192.168.2.1 255.255.255.248 Die weiteren möglichen Werte sind gleich wie beim Parameter ipaddr in der [system] Sektion. Die Interfaces mit X=1..4 können für die lokalen vlans verwendet werden, Interfaces mit X>4 können nur über einen externen Trunk mit der Aussenwelt kommunizieren. Es werden maximal 16 Interfaces unterstützt.
trunk		Mit diesem Parameter können VLAN Trunks auf den externen Switch-Ports konfiguriert werden. Der Parameter erwartet die Nummer des Switch-Ports, der als Trunk eingerichtet werden soll. Der Parameter kann mehrfach vorkommen. Beispiel: trunk = 3 Wenn in einem vlanX-Parametern der Trunk-Port eingeschlossen ist, kann das entsprechende vlan direkt über den Trunk

		kommunizieren, sonst müssen die Pakete vom AnyRover geroutet werden, damit eine Kommunikation zustande kommt.
dhcp_rebind	no	Mit diesem Parameter kann eingeschaltet werden, dass ein DHCP-Client einen rebind macht, wenn am Switchport ein Kabel eingesteckt wird.
reset	50	Internes Signal. Eine Änderung bewirkt, dass der Switch nicht mehr eingeschaltet werden kann.
ps1	123	Internes Signal. Eine Änderung bewirkt, dass der Switch nicht mehr konfiguriert werden kann.
port_poe1	53	Internes Signal. Eine Änderung bewirkt, dass das POE-Modul 1 nicht mehr eingeschaltet werden kann.
port_poe2	54	Internes Signal. Eine Änderung bewirkt, dass das POE-Modul 2 nicht mehr eingeschaltet werden kann.

4.3.3 [time]

Attribut	Wert (Default)	Beschreibung
timezome	UTC	Zeitzonen-Information. Der Wert ist einer der Einträge aus der ersten Spalte der Datei /etc/timezones. Mögliche Werte sind z.B.: CET, GMT (mit Sommerzeit), UTC (ohne Sommerzeit), EET, EST, ... Alternative: Genaue Zeitzone angeben, z.B. Europe/Zurich, America/Vancouver, Pacific/Auckland, ...
start	yes	Falls dieser Wert "yes" ist, wird ein NTP Daemon gestartet (als Server und als Client). Für Server-Betrieb muss noch der Port 123 in der Firewall freigeschaltet werden.
time_source	gps	gps: Die Systemuhr wird nach dem GPS-Empfänger gerichtet ntp: Die Systemuhr wird nach einem NTP-Server gerichtet none: Die Systemuhr wird nicht gerichtet
ntp_server	pool.ntp.org	NTP Server. Wird nur verwendet, wenn time_source=ntp
ntp_flags		Mit diesen Flags kann die Funktionalität des ntp-Servers eingeschränkt werden. Die Flags werden in der ntp.conf Datei in einer Zeile „restrict default <flags>" eingefügt. Erlaubt sind die Flags kod, limited, lowpriotrap, nomodify, noquery, nopeer, noserve, notrap, notrust, ntpport, version Die Liste kann mit Komma oder Leerzeichen getrennt werden. Sie sind in der man-Page des ntp.conf-Files dokumentiert (http://www.google.com/?#man+ntp.conf) Empfohlen wird die Zeile ntp_flags = kod nomodify notrap nopeer noquery Ohne Flags kann der NTP-Server für Denial-of-Service Angriffe ausgenutzt werden: eine NTP-Query generiert eine Antwort, welche viel grösser als die Query ist, und mit einer gefälschten Absenderadresse in der Query geht die Antwort zum Ziel-Host.
localaccess		Wenn dieser Wert auf yes gesetzt wird, können lokale Prozesse auf Status-Informationen zugreifen. Wird benötigt, damit show ntp korrekt funktioniert.
ntp_option		Mit diesem Attribut können zusätzliche ntpd Konfigurations-Optionen aktiviert werden.

jump_clock		Der NTP Daemon stellt nach dem Start die Systemzeit einmal auf die aktuelle Zeit, sofern die Differenz kleiner als 1000s ist. Ansonsten bricht er mit einer Fehlermeldung ab. Wenn dieser Parameter auf yes gesetzt ist, stellt der NTP Daemon die Systemuhr auf jeden Fall richtig und hält sie dann aktuell.
Modemclock	yes	Mit diesem Befehl wird die Systemzeit mit der Modemzeit abgeglichen, sobald eine 3G Verbindung besteht.
syncled		Wenn dieser Wert auf yes gesetzt wird, werden Time Syncs an den externen LEDs angezeigt.

4.3.4 [watchdog]

Attribut	Wert (Default)	Beschreibung
start	yes	Der Watchdog startet das System neu, wenn die feed-Leitung ca 1 Minute nicht ändert. Mit diesem Attribut kann der Watchdog gestartet werden. Ein Programm im System füttert den Watchdog regelmässig.
interval	15	Das Intervall (in Sekunden) nachdem die Watchdog-Leitung geändert wird. Der Watchdog schlägt nach ca. 45-75 Sekunden zu, der Wert sollte nicht über 30 Sekunden liegen.
gpio_on	122	Internes Signal.
gpio_feed	124	Internes Signal.
cmd_on	1	Internes Signal.

4.3.5 [crontab]

Attribut	Wert (Default)	Beschreibung
start	no	Wenn der Wert auf yes gesetzt ist, wird der Cron Daemon gestartet und die folgenden Einträge in die Crontab eingefügt. Der Cron Daemon kann auch von anderen Diensten gestartet werden, wenn dieser Wert auf no gesetzt ist. Dann werden allerdings die folgenden Einträge nicht in die Crontab eingefügt.
loglevel		Definiert, wieviel der Cron Daemon loggen soll. Mögliche Werte (tiefere Werte enthalten alle Logmeldungen der höheren Werte): 8: Schreibt für jedes ausgeführte Kommando eine Logmeldung 9: Nur Warnungen und Fehler Default-Wert: 8
entry		Ein Eintrag in der Crontab. Diese Zeile wird direkt in die Crontab-Datei kopiert, der Eintrag entspricht deshalb der Crontab-Syntax.

4.3.6 [gpio]

In dieser Sektion werden verschiedene Aktionen definiert, die bei Ereignissen an den GPIOs ausgeführt werden. Eine Aktion hat diese Form:

`time, action`
Wenn der Wert time (in Sekunden) vorhanden ist, wird die Aktion um time Sekunden verzögert ausgeführt. Die Präzision des Timeout liegt etwa im Bereich einer Sekunde.

Die Aktion kann ein beliebiges Programm mit Argumenten sein, dass dann gestartet wird (die ganze action wird als Parameter an „sh -c“ übergeben).

Es existiert eine Reihe von vordefinierten Aktionen, die auch in der [sms] Sektion verwendet werden können.

Aktion	Beschreibung
OFF	Das System wird ausgeschaltet. Das funktioniert nur, wenn das System über den GPIO-Stecker gespiesen wird und die Ignition-Leitung auf Low ist. Das System schaltet wieder ein, wenn die Ignition-Leitung auf High geht.
CANCEL	Bricht eine laufende OFF Aktion ab
RESET	Setzt die Konfiguration auf die Standardkonfiguration zurück. Die aktuelle Konfiguration wird nicht gespeichert.
REBOOT	Startet das System neu.
HALT	Fährt das System herunter. Es wird nicht ausgeschaltet.
MOUNT	Bindet alle vorhandenen USB Massenspeicher im System ein.
UMOUNT	Entfernt alle eingebundenen USB Massenspeicher aus dem System.
OUT_ON	Schaltet den GPO Pin ein (auf High)
OUT_OFF	Schaltet den GPO Pin aus (auf Low)
SMS_STATUS	Sendet Informationen über die GPI Pins als SMS an den Sender zurück. Funktioniert nur in der [sms] Sektion.

Die Attribute der [gpio] Sektion sind

Attribut	Wert (Default)	Beschreibung
button	2, RESET 5, UMOUNT 10, REBOOT	Aktion, die ausgeführt wird, wenn der Reset-Button länger als die angegebene Zeit (in Sekunden) gedrückt wird. Es können beliebig viele Aktionen definiert werden, nur müssen alle eine andere Zeit haben.
mode		Aktion für den Mode Button
ignition	(CANCEL), (60, OFF)	Aktion, die ausgeführt wird, wenn die Ignition Leitung ändert. Die erste Aktion wird bei der positiven, die zweite bei der negativen Flanke ausgeführt.
ign_boot		Definiert, welchen Wert für die Ignition-Leitung beim Booten angenommen werden soll. Abhängig von diesem Wert und dem aktuellen Stand der Ignition-Leitung wird augenblicklich eine Flanke detektiert und die entsprechende Aktion ausgeführt. Mögliche Werte: 0 (oder nicht gesetzt): Liest den aktuellen Wert der Ignition-Leitung aus 1: nimmt an, dass Ignition aus war während dem Booten 2: nimmt an, dass Ignition ein war während dem Booten

inputX	(),()	Aktionen, die ausgeführt werden, wenn das InputX Signal ändert. Die erste Aktion wird bei der positiven Flange, die zweite bei der negativen Flanke ausgeführt. Der AnyRover hat 3 Inputs.
hysteresis hysteresisX		Definiert eine Hysterese für Input-Signale. Wenn nicht definiert, wird der Wert 0 angenommen. Dieser Parameter muss nach dem entsprechenden inputX Attribut stehen. Syntax: hysteresis[X] = positive[,negative] X = Nummer des Inputs (1-3), kein X für Ignition-Leitung positive und negative bezeichnen die Anzahl an vorangegangenen Messwerten, welche den gleichen Wert haben müssen, bis eine Änderung am Eingang erkannt und die entsprechende Aktion ausgeführt wird. Messwerte werden alle 250ms gemessen. Wenn negative nicht definiert wird, gilt der gleiche Wert wie für positive.
gpio_ign	46	Internes Signal
gpio_in1	30	Internes Signal
gpio_in2	29	Internes Signal
gpio_in3	28	Internes Signal
gpio_off	126	Internes Signal
gpio_but	39	Internes Signal
gpio_mode	41	Internes Signal
gpio_status	87	Internes Signal

4.3.7 [gps]

In dieser Sektion werden TCP und UDP Verbindungen konfiguriert, über die GPS-Daten (NMEA-Strings) übermittelt werden. Es können beliebig viele Verbindungen konfiguriert werden, die Daten werden an alle Ziele übermittelt.

4.3.7.1 Targets

tcp_target und udp_target definieren aktive Verbindungen, d.h. der AnyRover versucht, die Verbindung aufzubauen (was bei einer UDP-Verbindung immer klappt, sofern eine Route zum Ziel vorhanden ist). Syntax:

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

Feld	Beschreibung
ret	Definiert den Zustand der Verbindung in Gegenrichtung. ret = 0: Daten in Gegenrichtung werden stillschweigend verworfen. ret = 1: In Gegenrichtung können Daten direkt an den GPS-Empfänger geschickt werden (z.B. mit dem u-center von u-blox: www.u-blox.ch) ret = 2: In Gegenrichtung können Kommandos an das System abgesetzt werden, in der Form CBCTL:{COMMAND}, wobei {COMMAND} ein Kommando ist, welches auch im cablynxctrl abgesetzt werden kann (z.B. CBCTL:ekfreset). Allfällige Ausgaben des Kommandos werden wiederum als GPTXT-Strings mit dem Tag CBCTL über die Verbindung geschickt. ret = 3: ret=1 und ret=2

target	IP-Adresse oder Hostname des Ziels. Bei der Verwendung von Hostnamen muss sichergestellt werden, dass der AnyRover den Hostnamen auflösen kann (z.B. über DNS).
port	Port-Nummer des Ziels. Falls kein Port angegeben wird, verwendet der AnyRover den Port 13179.
source	IP-Adresse, von der aus die Pakete verschickt werden. Der AnyRover muss diese IP-Adresse auf einem Interface konfiguriert haben. Wenn keine Adresse angegeben wird, verwendet das System die Adresse des Interfaces, über das die Pakete verschickt werden. Das wird z.B verwendet, wenn die Pakete über einen IPsec Tunnel verschickt werden.
port	Port Nummer, von der aus die Pakete verschickt werden. Kann interessant sein, wenn Firewalls zwischen dem AnyRover und dem Ziel sind.
interface	Netzwerk-Interface, das als Source für den Versand der Pakete verwendet wird. Mögliche Interfaces sind eth0 (Ethernet) und ppp (Modem). Das wird z,B. benötigt, wenn die Pakete durch einen IPsec Tunnel verschickt werden.
port	Port Nummer, von der aus die Pakete verschickt werden. Kann interessant sein, wenn Firewalls zwischen dem AnyRover und dem Ziel sind.
Id	Referenziert eine Filterregel mit dem Namen X (siehe unten). Wenn keine Filterregel definiert ist, dann werden alle Meldungen über diese Verbindung verschickt.

tcp_server und udp_server definieren passive Verbindungen, d.h. Der AnyRover wartet auf einen Verbindungsaufbau von aussen, und schickt danach die Daten über diese Verbindung. Syntax (Felder wie oben):

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

serial_target definiert eine serielle Schnittstellen, an welche die Daten übermittelt werden. Die Syntax ist:

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

Feld	**Beschreibung**
ret	Falls ret = 1 ist, können auf der gleichen Verbindung in der Gegenrichtung Befehle an den GPS-Empfänger geschickt werden (z.B. Mit dem u-center von u-blox: www.u-blox.ch). Falls ret = 0 ist, wird jeglicher Verkehr in Gegenrichtung stillschweigend verworfen.
serport	Name des seriellen Ports, z.B. /dev/ttyS1 für die erste serielle Schnittstelle, /dev/usbser0 für einen USB-Serial Konverter.
baudrate	Baudrate für die Übertragung. Mögliche Werte sind 2400, 4800, 9600, 19200, 38400, 57600, 115200
id	Referenziert eine Filterregel mit dem Namen X (siehe unten). Wenn keine Filterregel definiert ist, dann werden alle Meldungen über diese Verbindung verschickt.

Mittels file_target können die Daten auch in eine Datei geschrieben werden. Die Datei wird automatisch rotiert und gezipped, wenn sie eine definierte Grösse überschreitet. Alte Zip-Files werden gelöscht. Die Syntax ist:

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

Feld	Beschreibung
ret	Bei file_targets wird der Parameter ret nicht verwendet.
file	Name der Datei, in welche die Daten geschrieben werden sollen. Beim Rotieren wird dem Namen noch ".XX" angehängt und das File gezipped, d.h. das rotierte File heisst nachher z.B. [file].01.gz. Wenn die Dateien auf die Root-Partition geschrieben werden, ist die Grösse limitiert auf 10MB und rotate = 1. Damit wird verhindert, dass die Partition voll läuft.
maxsize	Wenn die Grösse der Datei diesen Wert überschreitet, wird die Datei rotiert. Die Grösse der Datei wird dabei periodisch geprüft, die exakte Grösse, bei welcher die Rotation stattfindet, kann daher nicht vorhergesagt werden. Mögliche Werte: 1 – 2'147'483'647 (=2GB) Default-Wert: 4MB
rotate	Definiert, wie viele alte Dateien aufbewahrt werden. Die rotierten Dateien werden als [file].01.gz, [file].02.gz, [file].03.gz, usw. angelegt, wobei [file].01.gz die jüngste Datei ist. Jeweils die älteste wird beim Rotieren gelöscht, wenn die maximale Anzahl erreicht ist. Mögliche Werte: 0 – 99 Default-Wert: 5
hook	Hier kann ein Programm definiert werden, welches ausgeführt wird, wenn die Datei rotiert wird. Wenn ein Hook definiert ist, wird die Datei nicht gezipped. Das Programm erhält den Namen der rotierten Datei als Parameter übergeben.
id	Referenziert eine Filterregel mit dem Namen X (siehe unten). Wenn keine Filterregel definiert ist, dann werden alle Meldungen über diese Verbindung verschickt.

4.3.7.2 Filter

Grundsätzlich werden alle Meldungen, welche vom GPS-Empfänger kommen, und über gptxt Parameter definiert wurden, an alle definierten Ziele weitergeleitet. Um die Flut an Meldungen einzuschränken, können Filter definiert werden, die in den Targets mit dem Attribut id:X angezogen werden, wobei X den Namen des Filters definiert und eine beliebige Kombination aus Buchstaben und Ziffern sein kann. Filter mit dem Namen anytracker und at_script werden intern verwendet. Kommentare sind am Ende der target-Definitionen nicht zulässig.

Wenn für ein Target kein Filter definiert ist, werden alle Meldungen gesendet. Die Filter-Regeln werden nacheinander abgearbeitet, bis eine passende Regel gefunden wird. Danach wird die Verarbeitung abgebrochen.

Jedes Filter kann zwischen verschiedenen Profilen hin- und herschalten, so dass basierend auf externen Einflüssen andere Filter-Regeln gelten können. Das Umschalten der Profile geschieht über cablynxctrl mit dem Befehl profile. Standardmässig ist immer Profil 0 aktiv.

Die Syntax für Filter-Regeln ist:

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

Achtung: hier wird ausnahmsweise das Semikolon als Feld-Trenner verwendet, nicht das Komma.

Feld	Beschreibung
ID	Name der Filter-Regel. Wird bei den Target-Definitionen referenziert.
profile	Profil-Nummer. Beliebige Zahl >= 0.
pattern	Text, der mit dem Anfang der Meldung verglichen wird, wobei ein Fragezeichen (?) für ein beliebiges Zeichen steht.
time	Zeit-Intervall in Sekunden, nach dem die nächste Meldung geschickt werden soll. Der Wert 0 heisst, dass nie eine Meldung geschickt wird.
dist	Distanz-Intervall in Metern. Das Gerät muss sich mindestens so weit bewegt haben, bis die nächste Meldung verschickt wird.

Regeln für eine ID können sowohl auf einer Filter-Zeile aufgeführt als auch auf verschiedene Zeile aufgeteilt werden, das Verhalten ist gleich.

Eine Filter-Regel generiert nie neue Meldungen. Jede Meldung wird, sobald sie erstellt wurde, von den Filter-Regeln verarbeitet und dann entweder verworfen oder versendet. Wenn keine Meldungen eintreffen, werden die Filter auch nicht aktiviert. Das heisst, wenn ein Filter definiert, dass alle 5 Sekunden eine $GPGGA Meldung verschickt werden soll, aber der GPS-Empfänger nur alle 10 Sekunden eine $GPGGA Meldung sendet, dann geht auch nur alle 10 Sekunden eine Meldung an die Empfänger.

UBX-Meldungen (zum UBX Protokoll siehe entsprechende Dokumentation auf der ublox Homepage) können ebenfalls gefiltert werden, wobei das Pattern so aussieht:

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

Die Liste der UBX-Klassen und -IDs ist auf dem AnyRover in der Datei /etc/ubx.txt zu finden.

Beispiele:

Sende nur GPGGA und GPRMC Meldungen:

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

Versende GPGGA Meldungen jede Sekunde, aber wenn die Zündung ausgeschaltet ist nur alle 10 Sekunden:

```
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 Attribute

Die Attribute für die [gps] Sektion sind

Attribut	Wert (Default)	Beschreibung

start	yes	Soll der GPS-Empfänger eingeschaltet werden?
run_gpsd	no	Definiert, ob der GPSD Dienst gestartet werden soll
gpsd_port	2947	TCP Port, auf dem der GPSD Verbindungen entgegennimmt. Muss in der Firewall freigeschaltet werden.
gpsd_debug	0	Definiert, wie gesprächig der GPSD ist. Je höher die Zahl, desto mehr Meldungen erscheinen im Syslog. Bei 2 werden u.a. alle NMEA-Strings geloggt.
assist_now		AssistNow ist ein Dienst, welcher aktuelle Satellitendaten online zur Verfügung stellt, mit deren Hilfe der GPS Empfänger nach dem Einschalten sofort eine gültige Position hat. Die Daten haben nur eine kurze Gültigkeit (einige Tage), daher muss dafür gesorgt werden, dass immer aktuelle Daten verfügbar sind. Wenn hier ein Dateiname angegeben wird, lädt das System die Daten beim Einschalten in den GPS-Empfänger. Die Daten können auch zu einem anderen Zeitpunkt über das Programm cablynxctrl geladen werden.
udp_target		Ziel für eine UDP-Verbindung
tcp_target		Ziel für eine TCP-Verbindung
tcp_server		Konfiguration für einen TCP Server
udp_server		Konfiguration für einen UDP-Server
serial_target		Konfiguration für eine serielle Schnittstelle
file_target		Schreibt die Daten in eine Datei.
filter		Filter Definitionen
tcp_init_str		Der hier angegebene String wird unverändert bei jeder TCP-Verbindung als erste Meldung übertragen.
atmsg_en		Der AnyRover kann Meldungen im AnyTracker Format erzeugen. {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 Aktuell können nur POSITION Meldungen erzeugt werden, und das Battery-Level (BATT) Feld enthält stets den Wert 99. Mit diesem Parameter wird definiert, ob solche Meldungen erzeugt werden. Mittels Filtern können diese Meldungen limitiert werden: filter = at; { = 0
atmsg_id		Definiert den Wert, der in das ID-Feld der AnyTracker Meldung eingefügt wird. Mit dem Wert <IMSI> oder <IMEI> können die entsprechende Werte des ersten Modems, mit <IMSI1> oder <IMEI1> jene des zweiten Modems verwendet werden.
atmsg_buggytime		Frühe Versionen des AnyTrackers hatten einen Bug in der Software, so dass die Zeit als hh.mm:ss anstatt hh:mm:ss übertragen wurde. Wenn dieser Parameter auf yes gesetzt wird, versendet der AnyRover dieses falsche Zeitformat.
atmsg_interval		Definiert die Intervalle (Zeit in s und Distanz in m), um AT-Meldungen zu erzeugen. Syntax: atmsg_interval = time, dist [, profile]
atmsg_use_utc_time		Definiert, ob in den AT-Meldungen UTC (yes) oder Lokalzeit (no) verwendet werden soll.
atmsg_script		Mit diesem Parameter wird ein Skript definiert, welches regelmässig aufgerufen wird, und dessen stdout dann als Meldung verschickt wird.

		Im Skript können diverse Umgebumgs-Variablen benutzt werden, um auf die aktuellen Positionsdaten zugreifen zu können: 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		Definiert Zeit- und Distanzintervall, gemäss dem dann atmsg_script aufgerufen wird. Zusätzlich kann ein Profil angegeben werden, sowie ein Pattern, das den generierten Meldungen entspricht, so dass diese von den anderen Meldungen unterschieden werden können.
gptxt		Konfiguration von GPTXT Meldungen. Syntax: gptxt = interval, action Alle intervall Sekunden wird die Aktion action ausgeführt, und die Ausgabe davon über GPTXT übermittelt. Intern sind folgende Aktionen definiert: GPI: Sendet den Zustand der GPI Pins in dieser Form: $GPTXT,GPI,A,B,#1,V1,#2,V2,#3,V3*XX A: Ignition, B: Reset Button #X: Nummer des GPI Pins, VX: Zustand des GPI Pins XX: Checksum (XOR von allen Bytes zwischen $ und *). WLAN: Liste von aktuell sichtbaren WLAN Access Points. Das funktioniert nur, wenn die WLAN-Karte als Client konfiguriert und gestartet ist. Format: $GPTXT,WLAN,<ssid>,<mac>,<channel_freq>,<signal>* DIP: Position der DIP-Schalter 1-6. Format: $GPTXT,DIP,0,0,0,1,0,0*3f Alternativ kann der Name eines beliebigen Kommandos (inkl. Pfad) angegeben werden, das dann ausgeführt und dessen Standardausgabe übermittelt wird. Wenn die Ausgabe Zeilenumbrüche enthält oder länger als 70 Zeichen ist, wird der Text auf mehrere GPTXT-Strings verteilt.
gptxt_file		Die aktuellen GPTXT-Meldungen werden regelmässig in diese Datei geschrieben, wo sie von anderen Programmen ausgewertet werden können. Das Verzeichnis muss existieren. Empfohlener Wert: /var/gps/gptxt.txt Die Meldungen werden nach dem ersten Feld nach GPTXT indiziert, und für jeden Index wird nur eine Meldung gespeichert. Beispiel: $GPTXT,INFO,Hello World* Der Index hier ist INFO, und die nächste Meldung mit GPTXT,INFO, überschreibt diese Meldung.
gptxt_writeout		Die GPTXT werden im hier definierten Intervall (in Sekunden) geschrieben. Ein Intervall 0 deaktiviert die Datei.
gptxt_clean		GPTXT-Meldungen, welche älter als die hier definierte Anzahl Sekunden sind, werden aus der Datei entfernt. Sobald sie wieder neu verschickt werden, kommen sie wieder in die Datei.
directory	/var/gps	Verzeichnis für die FIFOs, wo die NMEA-Strings zur Verfügung gestellt werden. Die FIFOs sollte nicht mit cat gelesen werden, da dieser Befehl nie terminiert. Besser ist “head -n 1 FIFO”.
gps_bypass	no	Schaltet den gps-bypass ein und aus. Mit dem gps-bypass können Daten direkt vom gps-Modul an die serielle Schnittstelle gesendet werden. Die gleiche Funktion kann auch mit einem serial_target ausgeführt werden, der Bypass ist jedoch schneller. Wenn der gps-bypass aktiviert wird, müssen die seriellen Schnittstellen in der Sektion serports ebenfalls aktiviert sein.
device	/dev/ttyS2	Internes Signal. Serielle Schnittstelle, an welcher der GPS-

		Empfänger angeschlossen ist.
baudrate	9600	Baudrate, mit welcher der GPS-Empfänger die Daten schickt. Eine Änderung dieses Parameters bewirkt KEINE Umprogrammierung des GPS-Empfängers.
gps_reset	38	Internes Signal
gps_on	125	Internes Signal

4.3.8 [sms]

Attribut	Wert (Default)	Beschreibung
start	yes	Soll der AnyRover auf eingehende SMS checken?
phone_number		Liste von Telefonnummern, die Befehle absetzen dürfen. Wenn keine Nummern angegeben sind, werden alle SMS zugelassen. Beispiel: phone_number = +41790123456, +41760987654
interval	15	Interval in Sekunden, nach dem der AnyRover das Modem nach neuen Meldungen fragt.
console	no	Definiert, ob der Befehl eco_% aktiviert ist oder nicht.
console_key	-	Wenn die SMS-Konsole deaktiviert ist, kann sie per SMS aktiviert werden ("eco enable KEY"). Wenn dieser Wert auf "-" gesetzt ist, kann die Konsole per SMS nicht eingeschaltet werden.
send_answer_back		Wenn dieser Wert auf yes gesetzt ist, dann schickt das System die ersten 160 Bytes der Ausgabe des Kommandos per SMS an den Sender zurück.
send_answer_to		Liste von Telefonnummern, an welche die Antworten der SMS-Kommandos (160 Bytes) geschickt werden (unabhöngig vom Wert von send_answer_back).
catch_all		Definiert einen Befehl, welcher ausgeführt wird, wenn für ein SMS kein passendes Kommando gefunden wird. Das Programm erhält die Absender-Nummer und den Text des SMS in zwei Umgebungsvariablen: $PHONE_NUMBER und $SMS_TEXT. Wenn kein catch_all definiert ist, werden die SMS verworfen.
sender_as_text	no	Hier kann angegeben werden, ob ein definierter SMS-Absender, wie zum Beispiel SWISSCOM als Text oder Nummer angezeigt und weiterverwendet wird. Wenn nichts gesetzt ist, wird die Nummer verwendet
anygator_tpm		Falls wget Downloads im AnyGator einen TPM key benützen müssen, kann hier die ID des Keys angegeben werden.

Der Rest der Einträge sind Kommandos, welche per SMS übermittelt werden können. Dabei entspricht das Attribut dem Kommando (Leerzeichen im SMS werden durch Unterstriche (_) ersetzt, die Attribut-Werte dürfen keine Leerzeichen enthalten). Der Wert besteht aus einem Flag und dem Befehl, der bei Erhalt ausgeführt wird (mittels „sh -c CMD").

Das Flag definiert, ob der Befehl mit einem Hash geschützt werden muss. Falls das Flag den Wert 0 hat, braucht das SMS keinen Hash, beim Wert 1 muss ein gültiger Hash vorhanden sein, und beim Wert 2 muss ein korrekte 3-Wege Handshake stattfinden (noch

nicht implementiert).

Attribute können auch Parameter enthalten. Wenn dem Attribut ein %-Zeichen angehängt wird, können die Parameter im Wert verwendet werden. Das eigentliche Kommando entspricht allen Zeichen bis zum ersten %. Im Wert definiert ein String %1 den ersten Parameter, %2 den zweiten, usw. Gültig sind die Parameter %1 bis %9, sowie %@, der alle Parameter repräsentiert. Wird im Kommando ein % selber benötigt, muss es verdoppelt als %% eingegeben werden.

Beispiele:

ping_router	0, ping -q -c 4 `route | awk '/^def/{print $2}'` | awk 'BEGIN{a=0}/^---/{a=1;next}a'	Schickt 4 Pings an den Standard-Gateway
ping_client	0, ping -q -c 4 `awk '/ List /{a=1;next}a{print $1;a=0}' < /etc/hosts` | awk '/^---/{a=1}a'	Schickt 4 Pings an den ersten DHCP-Client
position	1, head -n 1 /var/gpgga.fifo	Liefert die aktuelle GPS-Position zurück
config_%	1, cp /etc/cablynx_templates/%1 /etc/cablynx.conf	Kopiert die Datei /etc/cablynx_config/??? nach /etc/cablynx.conf. ??? wird durch das erste Wort im SMS nach config ersetzt.
eco_%	0, /etc/scripts.d/eco.sh %@	Einige vordefinierte Aktionen.

Die vordefinierten Befehle aus der [gpio]-Sektion können hier auch verwendet werden.

4.3.9 [modem]

Attribut	Wert (Default)	Beschreibung
name	modem1	Identifiziert das Modem, falls mehrere vorhanden sind
band	3	Definiert, auf welchem Band das Modem kommuniziert: Für 3G-Modems gelten diese Werte 0 = Automatisch 1 = nur UMTS 3G 2 = nur GSM 2G 3 = wenn möglich UMTS 3G 4 = wenn möglich GSM 2G Im 2G-Modus kann das Modem unter Last keine SMS empfangen. Für LTE-Modems gelten erweiterte Werte: 0-2 wie 3G-Modems 3, 4 = Automatic 5 = nur GSM und UMTS 6 = nur LTE 7 = GSM, UMTS, LTE 11 = nur UMTS und LTE 12 = nur GSM und LTE
disable_roaming	No	Deaktiviert das Roaming auf fremde Mobilnetze. Die Funktion ist aktiv, wenn der Parameter auf yes gesetzt wird.

sim_pin		Falls die SIM-Karte mit einem PIN geschützt ist, muss der PIN hier angegeben werden, ansonsten kann keine Verbindung aufgebaut werden.
imsi		IMSI Checker: Mittels Regeln kann definiert werden, ob die PPP Verbindung abhängig von der SIM-Karte gestartet werden soll, und auf welchem Interface. Die IMSI der aktuell eingelegten SIM-Karte kann mit diesem Befehl ermittelt werden: show id Die Regeln werden in der Reihenfolge, wie sie in der Konfigurations-Datei stehe ausgewertet. Die Form der Regeln ist: <IMSI>, X (-1 <= X < 2147483648). Wenn die IMSI übereinstimmt, wird ppp auf dem Interface pppX gestartet. Wenn X negativ ist, wird ppp nicht gestartet. Die IMSI "-" trifft auf alle SIM-Karten zu, darum macht es keinen Sinn, nach einer Regel mit "-" noch weitere einzufügen, die werden nie getestet. Beispiele (mit IMSI 228013520284438): Um die PPP-Verbindung nur für eine spezielle IMSI zu starten: imsi = 228013520284438, 0 imsi = -, -1 Um die PPP-Verbindung nur für eine IMSI auf ppp0 zu starten, und für alle anderen auf ppp100: imsi = 228013520284438, 0 imsi = -, 100 Um die PPP-Verbindung für eine IMSI nicht zu starten, aber für alle anderen schon: imsi = 228013520284438, -1 imsi = -, 0
wait_for_sim		Definiert, ob die SIM-Karte gefragt werden soll, ob sie bereit ist. Manche SIM-Karten brauchen nach Eingabe des PINs etwas Zeit, bis sie bereit sind, und manchmal wird die Verbindung zu schnell gestartet, so dass sie nicht zustande kommt. Dieser Parameter sollte immer auf yes gesetzt sein.
gpio	47	Internes Signal
disable	48	Internes Signal
cmd_on	0	Internes Signal
slot	0	Slot, in dem das Modem steckt. Für AnyRover immer 0.
get_modem_status	yes	Modeminformation werden zur Weiterverwendung in ein internes File geschrieben.
status_interval = 60	60	Definiert, wie oft die Modeminformationen abgefragt werden. Der Wert kann nicht kleiner als 20 Sekunden gewählt werden.
show_rx_led	yes	Zeigt den aktuellen RX-Level auf den Level-LEDs an
log_value		Loggt schlechte Modem-Verbindungen, wenn der Wert unter diesem Grenzwert liegt.
ringtime	30	Zeitdauer in Sekunden, die der AnyRover beim Auslösen eines Telefon-Anrufs (via cablynxctrl) klingeln lässt, bevor er den Anruf wieder beendet.
calltime	10	Zeitdauer in Sekunden, die der AnyRover den Anruf nach dem Verbinden offen lässt.
max_call_retry	10	Anzahl der Versuche, den Anruf aufzubauen, wenn der Versuch nicht erfolgreich war.
modem	/dev/clmodem	Internes Signal
hip	/dev/clhip	Internes Signal

ctrl	/dev/clctrl	Internes Signal
gps	/dev/clgps	Internes Signal

4.3.10 [usb]

Attribut	Wert (Default)	Beschreibung
poweron	yes	Definiert, ob an der externen USB-Schnittstelle die Power-Leitung eingeschaltet wird. Die interne WLAN Karte hängt ebenfalls an dieser Leitung, um sie zu verwenden muss dieser Parameter auf yes stehen.
usb1	yes	Schaltet die externe USB-Schnittstelle ein (Anschluss für WLAN-Modul). Um die Schnittstelle einschalten zu können, muss poweron auf yes stehen.
usb2	yes	Schaltet die externe USB-Schnittstelle ein (Anschluss für WLAN-Modul). Um die Schnittstelle einschalten zu können, muss poweron auf yes stehen.
usb4	yes	Schaltet die externe USB-Schnittstelle ein (USB Anschluss aussen am Gerät). Um die Schnittstelle einschalten zu können, muss poweron auf yes stehen.
switch_wlan	no	Mit diesem Parameter können die beiden WLAN-Karten wlan0 und wlan1 vertauscht werden.
restart_wlan		Ist dieser Wert auf yes gesetzt, wird die WLAN Karte bei Vendor Request Failed Meldungen automatisch neu gestartet.
start_sdcard		Schaltet die Stromversorgung der SD-Karte ein.
automount	yes	Definiert, ob USB Memory Sticks und SD-Karten automatisch ins System eingebunden werden sollen.
ignore_errors		Wenn der Parameter auf yes gesetzt ist, werden VFAT Filesystem-Fehler auf der SD-Karte ignoriert. Wenn der Parameter auf no ist, wird bei einem Fehler die SD-Karte auf read-only umgestellt.
sdpart		Mount Points für die Partitionen auf der SD-Karte. Dieser Parameter kann mehrfach vorkommen, für jede Partition einmal. Die Partitions-Nummern starten bei 1. Dieser Parameter ist nur wirksam, wenn automount = yes gesetzt ist. Syntax: sdpart = PartNum, MountPoint Beispiel: sdpart = 1, /media/sdcard1

4.3.11 [dhcp]

Diese Sektion konfiguriert einen DHCP-Server für ein Interface. Wenn mehrere Server auf verschiedenen Interfaces benötigt werden, kann diese Sektion mehrfach vorkommen.

Die Parameter sind in 4 Kategorien aufgeteilt. Zuerst stehen die generellen Informationen, dann folgen die bootp-spezifischen Parameter (next_server, hostname und boot_file). Diese Werte stehen im DHCP/Bootp Paket selbst. Der Rest besteht aus sehr vielen DHCP-Einstellungen, welche dem eigentlichen Paket als Optionen angehängt werden. Ganz am Schluss sind schliesslich die Einträge für statische Leases zu finden.

Attribut	Wert (Default)	Beschreibung
name	-	Name des Interfaces, auf dem der DHCP-Server läuft. Möglich sind: eth0, vlan1 .. vlan4, wlan0
start	yes	Starte den DHCP-Server
lease_file		Speicherort des Lease-Files für den Server. Wenn nicht definiert, wird das File unter /var/lib/misc/udhcp.leases.<IFACE> abgelegt, was in einer RAM-Disk liegt und beim nächsten Reboot verloren geht.
log		Wenn der Parameter auf syslog gesetzt wird, dann loggt der DHCP Server seine Aktionen im Syslog.
port	67	UDP Port, auf dem der Server DHCP Requests erwartet.
dhcpd_start	192.168.1.11	Niedrigste Adresse, welche der DHCP-Server ausgibt
dhcpd_end	192.168.1.254	Höchste Adresse, welche der DHCP-Server ausgibt
next_server		Diese IP-Adresse wird im Feld next_server im bootp Header platziert.
server_hostname		Dieser Name wird als Hostname des DHCP-Servers publiziert.
boot_file		Name des Files, welches der bootp-Client verwenden soll.
prefix		Präfix für den dynamischen Bereich.
netmask		Netzmaske für den dynamischen Bereich. Wenn prefix und netmask angegeben sind, wird der Wert von netmask verwendet. Wenn weder prefix noch netmask angegeben sind, verwendet er die klassenbasierte Maske der Adresse von dhcpd_start.
router	192.168.1.3	Default Router für die Clients. Dieser Parameter kann mehrfach vorkommen, dann werden den Clients mehrere Router gemeldet. Wenn der Wert auf default gesetzt ist, wird die IP Adresse des Interfaces, auf dem der Dienst läuft, verwendet.
dns	192.168.1.3	Nameserver, welcher der DHCP-Server an die Clients meldet. Dieser Parameter kann mehrfach vorkommen.
lease	864000 (10 Tage)	Zeit in Sekunden, für welche der Lease abgegeben wird. Der Wert kann alternativ gefolgt von einer Einheit (min, hour, hours, day, days) angegeben werden. Die Einheit muss mit einem Leerzeichen von der vorangehenden Zahl getrennt werden.
timezone		Zeit-Offset in Sekunden gegenüber UTC. Damit kann die aktuelle Zeitzone definiert werden.
*logsrv *lprsrv *nissrv *ntpsrv *wins swapsrv tftp		IP-Adresse für die angegebenen Server. Die mit * markierten Parameter können mehrfach vorkommen, um mehrere Server anzubieten.
hostname		Hostname, welcher den Clients übermittelt wird.
bootsize		Grösse der Boot-Datei in 512 Byte Blöcken.
domain		Domainname, welche der Client für Anfragen im DNS verwenden soll.
rootpath		Pfad zur Root-Disk des Clients.
ipttl		TTL-Wert, den der Client verwenden soll.
mtu		MTU für das lokale Netzwerk.
broadcast		Die Broadcast-Adresse im lokalen Netzwerk.
nisdomain		NIS-Domainname.

requestip	IP-Adresse
dhcptype	Nummer
serverid	IP-Adresse, welche als Server-ID gesendet wird.
message	Text
vendorclass	Text
clientid	Text
bootfile	Name des Files, welches der DHCP-Client als Boot-Image verwenden soll.
wpad	Einstellungen für MSIE Web Proxy Autodiscovery Protocol
vendorspec	Hier kann ein beliebiger Hex-String angegeben werden. Das Format ist vendorspec = 41:65:d:a:0
sipsrv	Adresse eines SIP Servers. Wenn mehrere Werte angegeben werden, müssen alle den gleichen Type haben. Format: type:URL Type = 0: URL ist eine echte URL; 1: URL ist eine IP-Adresse
captiveport114	Captive Portal gemäss RFC 8910 (Option 114)
captiveport160	Captive Portal gemäss RFC 7710 (Option 160)
static_lease	Definiert einen statischen Lease für eine bestimmte MAC-Adresse. Die MAC-Adresse und die IP-Adresse sind durch ein Leerzeichen getrennt. Dieser Parameter kann mehrfach vorkommen. Beispiel: static_lease = 00:11:22:33:44:55 192.168.1.11

4.3.12 [dhcprelay]

In dieser Sektion kann ein DHCP-Relay konfiguriert werden.

Attribut	Wert (Default)	Beschreibung
start	no	Definiert, ob der Service gestartet werden soll.
client		Liste mit Interfaces (mit Komma getrennt), auf denen DHCP-Request empfangen werden. Wenn nichts eingetragen wird, empfängt der Daemon auf allen Interfaces. Wenn ein '!' vor einem Interface steht, wird dieses von der Liste explizit ausgeschlossen. Der Eintrag “client = !vlan1” empfängt auf allen Interfaces ausser vlan1.
server		Liste mit Servern (mit Komma getrennt), an welche die DHCP-Requests weitergeleitet werden sollen. Das können IP-Adressen oder Interfaces sein. Wenn eine IP-Adresse angegeben wird, schickt der Daemon das Paket als Unicast-Paket an diese Adresse, bein einem Interface wird es als Broadcast auf dem Interface verschickt. Das gw-addr Feld im DHCP-Header wird mit der Adresse, auf der das Paket empfangen wurde, gefüllt.

4.3.13 [ftp]

Attribut	Wert (Default)	Beschreibung
start	no	Starte den FTP Server
basic	yes	Grundlegende Konfigurations-Optionen.
anonymous	yes	Erlaube anonymous-Logins?
anonymous_dir	/media/sda1	Verzeichnis für den anonymous-Users.
anonymous_write	no	Darf der anonymous-User Dateien hochladen?
anonymous_delete	no	Darf der anonymous-User Dateien löschen?
option		Diese Optionen werden direkt ins vsftpd.conf-File geschrieben. Hinweis: vsftpd erlaubt keine Leerzeichen in den Optionen.

4.3.14 [tftp]

Attribut	Wert (Default)	Beschreibung
start	no	Starte den TFTP Server?
upload	no	Sind Datei-Uploads erlaubt?
rootdir	/tftp	Verzeichnis für den TFTP Server. Nur Dateien in diesem Verzeichnis können über TFTP geladen werden, und Uploads werden hier gespeichert.
port	69	UDP Port, auf dem der TFTP Server Verbindungen annimmt.

4.3.15 [firewall]

In diesem Abschnitt werden Firewall-Regeln definiert. Die Regeln werden in der Reihenfolge, wie sie definiert wurden, angewendet, daher ist auf korrekte Anordnung der Regeln in der Konfigurationsdatei zu achten.

Die Regeln sind in zwei Kategorien eingeteilt: Als Firewall-Regeln werden diejenigen Regeln bezeichnet, welche die Pakete nur analysieren und dann eine Entscheidung treffen. Als Mangle-Regeln gelten alle Regeln, welche Pakete auch verändern. Dazu gehört z.B. NAT oder Port Forwarding.

Attribut	Wert (Default)	Beschreibung
filter_bridged	yes	Wenn dieser Wert gesetzt ist, werden die Pakete, welche über die Bridge gehen, von der Firewall gesehen. Dieser Wert kann nur global und nicht pro Bridge gesetzt werden.
filter_vlan	yes	Wenn dieser Wert gesetzt ist, werden Pakete mit VLAN Tags, welche über die Bridge gehen, von der Firewall gesehen. Dieser Wert kann nur global und nicht pro Bridge gesetzt werden.
forward	yes	Der Kernel agiert als Router und leitet Pakete weiter. Falls dieser Wert auf no gesetzt wird, können z.B. Clients im lokalen Netz nicht aufs Internet zugreifen.
nflog_start		NFLOG ist ein Logging-Verfahren, wo die geloggten Pakete von Userspace Programmen empfangen und ausgewertet werden können. Wenn dieser Parameter auf yes gesetzt ist, dann wertet das System NFLOG Pakete aus.

nflog_script	/etc/scripts.d/ nflog.sh	Wenn ein Paket über NFLOG geloggt wird, dann führt das System dieses Skript aus und übergibt alle relevanten Informationen über Umgebungsvariablen (NFLOG_*). Das Default-Skript führt alle Skripte in /etc/scripts.d/nflog/ in alphabetischer Reihenfolge aus.
nflog_group	7	Das NFLOG Target kennt verschiedene Gruppen. Mit diesem Parameter wird definiert, in welcher Gruppe das System auf Meldungen wartet. Mögliche Werte: 1-32
nflog_payload_length	64	Definiert die Anzahl Bytes, welche bei UDP Paketen in der Variablen NFLOG_PAYLOAD dem Skript übergeben werden. Wenn beim Kopieren nicht druckbare Zeichen gefunden werden, bricht der Vorgang schon früher ab.
start_firewall		Die Firewall-Regeln werden nur eingeschaltet, wenn dieser Wert auf yes gesetzt ist.
basic	yes	Schaltet einige grundsätzliche Regeln ein: – Keine Verbindungen zum AnyRover oder durch den AnyRover (Wurzel-Chain Policies) – Erlaubt sind ICMP Echo Requests (Ping) – Erlaubt sind bestehende Verbindungen – Erlaubt sind verwandte Verbindungen (z.B. FTP data, ICMP errors)
new_chain		Syntax: new_chain = NAME[,CHAIN[:POS]] Dieser Parameter erlaubt die Erzeugung einer neuen Filter-Chain, das Argument ist der Name der neuen Chain. Der Name darf keine Leerzeichen und Unterstriche (_)enthalten. Um mehrere neue Chains zu erzeugen, muss das Attribut mehrfach verwendet werden. Bei Angabe von CHAIN wird eine Jump-Regel zur neuen chain NAME in CHAIN an Position POS eingefügt.
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 von Firewall-Regeln. Syntax: TARGET = [SRC][,[!]proto[,DST]][,R:RATE][,L:prefix][,I:ICMP][,MAC:[!]ADDR] wobei: 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) Ein Ausrufezeichen vor einem Wert invertiert die Auswahl, die Regel trifft dann auf alles ausser dem angegebenen Wert zu. if: Input oder Output Interface. Für Bridge-Interfaces kann das physikalische Interface angegeben werden: br0>vlan1. ipsec: Die Regel trifft nur dann zu, wenn das Paket über einen IPsec Tunnel eintrifft (für SRC) oder verschickt wird (für DST). Der Parameter ipsec kann nur entweder in SRC oder in DST angegeben werden (z.B. für *_fw Regeln). net: Quell- oder Zielnetzwerk port: Quell- oder Zielport; nur aktiv wenn das Protokoll tcp oder udp ist. Mehrere Ports können mit einem Doppelpunkt getrennt angegeben werden.

		proto: Protokoll (tcp, udp, esp, icmp) rate: Damit kann das Zutreffen der Regel limitiert werden. Mögliche Werte sind z.B. 3/sec, 6/min, 13/hour, 2/day. Diese Funktion wird hauptsächlich für Log-Regeln verwendet, um ein Überlaufen des Logfiles zu verhindern. Dieser Parameter ist nicht geeignet, um Bandbreiten-Begrenzungen zu machen. burst: Wenn eine Regel eine Limite hat, werden zuerst soviele Treffer gelandet, bevor die Limitierung startet (Default: 5). ADDR: Eine MAC-Adresse (Source MAC des Pakets), nach der gefiltert werden soll. ICMP: ICMP-Meldung, welche bei Reject Regeln zum Sender zurückgeschickt werden soll. (Default: icmp-port-unreachable) prefix: Ein Text, der im Logfile vor den Paket-Informationen ausgegeben wird. Der Text darf keine Komma und Hochkomma (') enthalten. Er kann in Anführungszeichen (") gesetzt sein, was nur gebraucht wird, wenn der Text mit Leerzeichen endet. Der Unterschied zwischen reject und drop ist, dass drop das Paket stillschweigend verwirft, während reject den Sender per ICMP informiert. Reject sollte nur verwendet werden, wenn es wirklich nötig ist. Return beendet die Verarbeitung in der aktuellen chain und kehrt zur Eltern-Chain zurück, oder appliziert die Policy für die Wurzel-Chains. Wenn als RULE der Name einer eigenen Chain angegeben wird, führt ein Zutreffen der Regel dazu, dass die Verarbeitung in der angegebenen Chain fortgesetzt wird. Die _in Chain gilt für Pakete, welche an den AnyRover selber addressiert sind, die _fw Chain für Pakete, welche durch den AnyRover geroutet werden, und die _out Chain für Pakete, welche auf dem AnyRover generiert wurden und nun den AnyRover verlassen. _NAME kann der Name einer eigenen Filter-Chain sein (siehe oben), die Regel wird dann in diese Chain eingefügt. Wenn keine Chain angegeben wird (z.B. accept = ...), werden die Regeln in die Input-Chain eingefügt. Wenn eine Regel zutrifft, wird die Verarbeitung beendet und das Paket entsprechend der Regel behandelt. Die Ausnahme sind die (nf)log-Regeln, welche die Verarbeitung nicht beenden
rule		Hier können direkt IPTABLES Regeln angegeben werden. Der Wert wird direkt an iptables übergeben. Für Dokumentation siehe www.netfilter.org
start_mangle		Die folgenden Parameter zum Paket Mangling werden nur aktiviert, wenn dieser Wert auf yes gesetzt wird.
nat	ppp0	Liste von Interfaces, auf denen NAT eingeschaltet wird.
new_natchain		Wie new_chain, aber für die NAT-Tabellen. Die Chains PPPD_NAT und IPSEC_NAT werden vom System angelegt.
portfw		Definiert Port-Forwarding Regeln. Syntax: [proto],target[:tport],dest[:dport][,source] proto: Protokoll. Wenn kein Protokoll angegeben wird, gilt die Regel für alle Pakete. target: Interface Name oder IP Adresse, die im Paket als Ziel angegeben sind. Es kann auch eine Netzwerkadresse angegeben werden (z.B. 192.168.2.0/24) tport: Port, an den das Paket geschickt wird dest: Neue Zieladresse, die eingetragen wird. dport: Neuer Port, an den das Paket weitergeleitet wird. Wenn dieser Wert weggelassen wird, bleibt der Port unverändert. source: Wenn angegeben, die Quell-Adresse, von der das Paket kommt.

snat[_chain] dnat[_chain]	Source- und Destination-NAT Regeln. Die Syntax ist gleich wie bei den accept-Regeln: snat = [SRC],[proto],[DST],T:target dnat = [SRC],[proto],[DST],T:target Das target definiert die zu setzende Source- oder Destination-Adresse. Für snat ist die Angabe einer Input-Schnittstelle nicht möglich, für dnat die einer Output-Schnittstelle (Adressen dagegen schon). Sowohl für die IP-Adresse wie für den Port im NAT-Target können Bereiche angegeben werden, z.B.: 10.0.0.1-10.0.0.10:1000-1200. Destination-Nat Regeln werden angewendet beim Eingang eines Paketes, bevor eine Routing-Entscheidung getroffen wird, Source-Nat beim Verlassen des Systems, nachdem die Routing-Entscheidung getroffen wurde, aber noch bevor die Entscheidung über IPsec-Verschlüsselung fällt. Destination-Nat ist ist das selbe wie portfw weiter oben, allerdings mit anderer Syntax der Regeln. Beispielanwendung: Für Syslog oder NTP kann keine Source-Adresse konfiguriert werden, es wird die Source-Adresse des Interfaces auf dem direkten Weg zum Target verwendet. Für die Verwendung mit IPsec zum Beispiel kann es nötig sein, eine andere Source-Adresse zu verwenden. Ist der Syslog-Server 10.11.12.13 und 192.168.1.3 die interne Adresse, so lautet die Regel wie folgt: snat = ,udp,10.11.12.13:514,T:192.168.1.3 Weitere Beispiele: 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*	Damit kann die TCP MSS (Maximum Segment Size) geändert werden. Die MSS wird nur im ersten Paket einer TCP-Session übertragen (d.h. SYN Bit gesetzt). Syntax: tcpmss_*chain* = [SRC],tcp,[DST],M:*{mss}* Als chain können in, fwd, out, PREROUTING und POSTROUTING verwendet werden. SRC und DST sind definiert wie in den accept_* Regeln weiter oben. In PREROUTING darf kein Output-Interface, in POSTROUTING kein Input-Interface definiert sein. Quell- und Zielnetze hingegen schon.

4.3.16 [dyndns]

Attribut	Wert (Default)	Beschreibung
start	yes	Starte den DynDNS-Daemon?
username	user	Benutzername beim DynDNS Provider
password	pass	Passwort beim DynDNS Provider
hostname	myhost.dyndns.org	Hostname des Servers beim DynDNS Provider, bei dem das Update durchgeführt werden muss. Dieses Attribut kann mehrfach vorkommen, wenn der Provider mehrere mögliche Server hat.
option	syslog	Optionen, welche direkt in die Konfiguration des INADYN Programms geschrieben werden.

4.3.17 [ppp]

Diese Sektion definiert eine PPP-Verbindung über ein 3G-Modem. Die Sektion kann mehrfach vorkommen für mehrere Verbindungen (benötigt aber mehrere Modems).

Um bei Zustands-Änderungen (Verbindung aufgebaut oder beendet) Aktionen auszuführen, können Skripte in /etc/scripts.d/ppp-up-hooks oder /etc/scripts.d/ppp-down-hooks gespeichert werden. Die Skripte erhalten informationen über Parameter.

Die Skripte selbst können über Skript-Sektionen definiert werden.

Attribut	Wert (Default)	Beschreibung
modem		Dieser Parameter, der an erster Stelle in der Sektion stehen muss, referenziert eine Modem Sektion und definiert, über welches Modem die PPP-Verbindung aufgebaut wird.
start	yes	Starte den PPP Daemon? Ohne PPP Daemon kann keine UMTS Verbindung hergestellt werden.
user		Username für die Anmeldung beim Provider. Falls keiner benötigt wird, muss die Zeile auskommentiert werden.
password		Passwort für die Anmeldung beim Provider. Falls keins benötigt wird, muss die Zeile auskommentiert werden.
defaultroute	yes	Wenn dieser Wert auf yes gesetzt ist, wird die PPP-Verbindugn als Default-Route eingetragen.
defaultmetric		Mit diesem Parameter kann die Metrik der Default-Route gesetzt werden. PPP setzt keine Default-Route, wenn bereits eine mit der gleiche Metrik existiert (Default: 0)
usedns		Definiert, ob die Nameserver, wie sie der PPP-Peer meldet, verwendet werden sollen.
debug	no	Falls der Wert auf yes gesetzt wird, schreibt der ppp daemon zusätzliche Informationen ins Log-File.
basic	yes	Nimmt die Standard-Optionen für den ppp Daemon.
chat_verbose	yes	Wenn der Wert auf yes gesetzt wird, wird vom chat sowohl der Ausführungsstatus als auch alle gesendeten und empfangenen Nachrichten während dem Einwählen gelogt. Wird nur ausgeführt, wenn basic=yes ist.
chat_script	basic	Auswahl der chat_script Sektion. Dieser Wert referenziert direkt das name-Attribut in einer chat_script Sektion.
restart	yes	Definiert, ob das Modem neu gestartet wird, wenn die PPP-Verbindung abbricht.
timeout	2	Zeit in Sekunden, die das Modem ausgeschaltet bleibt.
hold_nocarrier		Wenn der Verbindungsaufbau mit „NO CARRIER“ fehlschlägt, wird das Modem nicht neu gestartet, wenn dieser Parameter auf yes gesetzt ist. Dadurch verbindet sich das Modem nach Funklöchern in den meisten Fällen schneller wieder.
filter		Filter-Ausdruck für Dial-on-demand. Ein Paket startet die Dial-on-demand Verbindung nur, wenn es diesem Filter entspricht. Wenn nichts angegeben wird, passen alle Pakete. Ebenso wird der Idle-Counter nur zurückgesetzt, wenn ein Paket diesem Filter entspricht. Die Filtersyntax basiert auf den Ausdrücken von tcpdump. Ausdrücke, welche für PPP-Links nicht passend sind, wie ether oder arp, sind nicht zulässig. Für weitere Details wird auf die

		Man-Page von tcpdump verwiesen. Syntax: [(] expr [)] [and | or] [[(] expr [)]] Wenn src und dst weggelassen werden, gelten beide Richtungen [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 ist eins von <, >, <=, >=, =, != expr kann enthalten: Zahlen, +, -, *, /, <<, >>, &, | PROTO [expr:size] Beispiel: filter = outbound and not ((tcp[13] & 4 != 0) of (icmp[0] = 3))
option	demand persist idle 300 holdoff 15	Optionen, welche direkt in die ppp Konfigurationsdatei geschrieben werden. Wenn die Option demand ohne persist verwendet wird, muss explizit nopersist angegeben werden.

4.3.18 [chat_script]

Attribut	Wert (Default)	Beschreibung
name	basic	Identifiziert die Sektion. Der Name wird in der [ppp] Sektion referenziert.
apn	gprs.swisscom.ch	APN für den UMTS-Zugang. Dieser Parameter ist nur in der Sektion mit dem Namen basic wirksam.
script		Alle script-Zeilen werden ins Chat Skript kopiert. Diese Zeilen sind nur in chat_script Sektionen mit einem anderen Namen als basic wirksam.

4.3.19 [wan]

Diese Sektion wird aktuell nicht verwendet.

4.3.20 [ipsec]

Mit dieser Sektion wird ein IPsec Tunnel konfiguriert. Wenn mehrere Tunnels zu verschiedenen Peers benötigt werden, kann die Sektion mehrfach verwendet werden.

Um bei IPsec Zustands-Änderungen Aktionen auszuführen, können passende Skripte im Verzeichnis /etc/scripts.d/ipsec-hooks/ abgelegt werden. Es muss entweder eine Datei oder ein Verzeichnis mit einem dieser Namen existieren: up-host, down-host, up-client, down-client. Im Falle eines Verzeichnisses werden alle Skripte mit Namen *.sh, welche sich in diesem Verzeichnis befinden, ausgeführt. Alle Skripte müssen ausführbar sein.

Die Skripte werden dann bei der jeweiligen Zustands-Änderung ausgeführt, wobei die benötigten Informationen in Umgebungsvariablen übergeben werden.

Die Skripte selber können über script-Sektionen definiert werden.

Attribut	Wert (Default)	Beschreibung
start	yes	Starte den Ipsec Daemon?
name		Name für die Verbindung; dieser Name wird im Config-File /etc/ipsec.conf verwendet.
setup		Definiert die Aktion für diese Verbindung, wenn IPsec gestartet wird. Mögliche Werte start: Versucht, die Verbindung aufzubauen route: Startet die Verbindung erst, wenn Verkehr herrscht add: Startet die Verbindung nicht, wartet auf Gegenstelle
ike	2	Version des IKE-Protokoll, das verwendet werden soll. Mögliche Werte: 1 oder 2. Cisco Geräte (und Cisco VPN Client) verwenden Version 1.
mode	tunnel	Mit diesem Parameter kann der IPsec Mode definiert werden. Mögliche Werte sind tunnel oder transport. Wenn nichts angegeben wird, ist der Tunnel Mode aktiv. Im Transport Mode müssen die beiden Parameter remote und local gesetzt sein, und local_net sowie remote_net werden ignoriert.
fragmentation		Wenn dieser Wert auf yes gesetzt ist, werden grosse IKEv2 Meldungen als Fragmente verschickt. Bei no muss allenfalls IP im Tunnel fragmentieren. Nur für IKEv2 relevant.
mobike		Mit dieser Option kann MobIKE (RFC 4555) aktiviert werden. Damit kann ein Tunnel neu ausgehandelt werden, wenn die IP-Adresse eines Endpunktes ändert. Nur für IKEv2 relevant.
remote	192.168.17.42	IP-Adresse oder Hostname des Peers. Ein Hostname muss in Anführungszeichen (“) stehen und auflösbar sein. Wenn die IP Adresse des Gegenübers nicht bekannt ist (Road Warrior), kann hier “any” angegeben werden.
local		Definiert die Interfaces, auf denen der IPsec-Dämon auf eingehende Verbindungen wartet.
local_within		Wenn die Verbindung zum remote Server über ein Interface läuft, welches hier gelistet ist, dann wird die IPsec Verbindung aufgebaut. Wenn das Interface nicht in der Liste ist, wird die Verbindung nicht aufgebaut. Wenn der Parameter leer oder nicht definiert ist, wird die Verbindung aufgebaut.
local_net		Liste von lokalen Netzwerken oder Interfaces, welche den IPsec-Tunnel benützen können. Ist local_net nicht definiert, wird das erste konfigurierte Interface aus dieser Liste verwendet: eth0, vlan1-4
protocol		Limitiert den IPsec Tunnel auf ein einziges Protokoll und/oder einen Port. Sowohl Protokoll als auch Port können als numerische Werte oder Namen angegeben werden. Syntax: [proto][,[sport][,dport]] Damit wird nur Traffic mit dem entsprechenden Protokoll, Source oder Destination Port durch den Tunnel geschickt. In Gegenrichtung sind die Ports vertauscht. Beispiele: protocol = tcp, http protocol = udp protocol = , 443 protocol = udp, 67, 68
remote_net	192.168.1.0/24	Netzwerk am anderen Ende des Tunnels. Es können mehrere Netzwerke, getrennt durch Leerzeichen, angegeben werden.

		Im Falle von Road Warriors, wo das Netz hinter dem Client nicht bekannt ist, kann any angegeben werden. Dann übernimmt der Server das Netz, wie es vom Client angepriesen wird.
remote_range		Wenn hier eine Netzwerk-Adresse angegeben wird, können Road Warriors nur verbinden, wenn ihre IP Adresse aus diesem Bereich stammt.
remote_address		Die interne Source IP Adresse des Peers zur Verwendung in einem Tunnel. Diese wird zum Beispiel in Verbindung mit Cisco VPN Clients benötigt. Wenn der Wert auf %config gesetzt wird, sendet das System die vom Peer vorgeschlagene Adresse zurück.
tunnel		Definiert, welche local_nets mit welchen remote_nets kommunizieren können. Wenn nichts angegeben wird, sind alle Möglichkeiten erlaubt. Alle local_nets werden mit einer Ziffer (1,2,...), alle remote_nets mit einem Kleinbuchstaben (a,b,...) bezeichnet. Der Parameter enthält eine Liste mit allen erlaubten Verbindungen. Wenn der Wert mit einem Schrägstrich (/) beginnt, werden die verbotenen Verbindungen aufgelistet. Dieser Parameter wird auch verwendet, um Source Policy Routing über den IPsec Tunnel zu implementieren. Wenn der Parameter von einem Doppelpunkt und optional einem Interface oder einer IP-Adresse gefolgt wird, setzt das System, sobald der Tunnel steht, eine Route zum Remote Netz mit der angegebenen Source Adresse. Wenn keine Adresse angegeben wird, verwendet das System die lokale Adresse des lokalen Subnetzes. Um die Source Policy Route für alle Verbindungen zu setzen, muss der Doppelpunkt als erstes Zeichen im String stehen, noch vor einem allfälligen "/". Beispiel: local_net = loc1 loc2 remote_net = rem1 rem2 tunnel = 1a 1b 2b tunnel = /2a Beide Einträge bewirken, dass loc1 mit rem1 und rem2 sowie loc2 mit rem2 verbinden kann. tunnel = 1a:eth0 1b:192.168.1.1 2a: 2b tunnel = :1a 1b 2a tunnel = :/2a tunnel = :/
dpd	5,5,5	Definiert die Parameter für Dead Peer Detection (DPD). Die Werte sind: dpd_delay, dpd_retry, dpd_max dpd_delay: IPsec sendet ein DPD-Paket alle N Sekunden. 0 schaltet DPD ab. dpd_retry: Zeit in Sekunden, bis das Paket als gescheitert betrachtet wird. dpd_max: Anzahl von aufeinanderfolgenden gescheiterten Paketen, bis die Verbindung als tot betrachtet wird.
dpdaction		Definiert die Aktion für den Fall, dass eine tote Verbindung entdeckt wird. Mögliche Werte: restart: Versucht, die Verbindung neu aufzubauen. clear: Löscht die Verbindung und alle Routen. Dadurch kann die Verbindung nicht mehr aufgebaut werden. hold: hält die Verbindung aufrecht
tries	0	Wie viele Versuche sollen unternommen werden, um eine Verbindung herzustellen oder zu erneuern. Der Default-Wert von 0 bedeutet unendlich.

closeaction		Definiert, was passiert, wenn der Peer den Tunnel sauber schliesst: restart: Versuch, den Tunnel wieder aufzubauen clear: Tunnel schliessen und abwarten hold: Traps installieren, so dass der Tunnel bei Verkehr wieder aufgebaut wird.
my_identifier	asn1dn,	Methode, wie der AnyRover sich beim Gegenüber identifiziert (das Komma nach asn1dn ist nötig). Mit Hostname: fqdn, NAME Mit Adresse: address, ADDRESS Mit Zertifikaten: asn1dn, Falls mit Zertifikaten gearbeitet wird, und als ID eine IP-Adresse oder ein Hostname angegeben wird, muss diese ID als subjectAltName im Zertifikat drin stehen, damit der AnyRover das Zertifikat korrekt zuordnen kann.
peers_identifier	asn1dn,	Methode, wie der Gegenüber identifizert werden soll. Siehe my_identifier
auth_method	cert	Definiert die Authentifizierungs-Methode für IKEv1. Mögliche Werte sind: psk: Pre-Shared Key cert: Zertifikate xauth-psk: XAUTH mit Pre-Shared Key xauth-cert: XAUTH mit Zertifikaten
local_auth remote_auth		Authentifizierungs-Methode für IKEv2. Mögliche Werte: any: (Default für remote_auth wenn nichts gesetzt) psk: Pre-Shared Key pubkey: Zertifikate
psk	mysupersecretkey	Pre-Shared Key
xauth		Definiert die Rolle in XAUTH Authentifizierung. Mögliche Werte sind server und client. Dieser Parameter ist nur relevant wenn auth_method auf xauth-psk oder xauth-cert gesetzt ist.
xauth_id		Benutzername und Passwort für XAUTH Authentifizierung. Dieser Parameter kann mehrfach vorkommen. Syntax: xauth_id = username:password
cert	ipsec-cert	Zertifikat für die Identifikation. Referenziert eine [certificate] Sektion.
root	ipsec-root	Root-Zertifikat für die Identifikation. Referenziert eine [certificate] Sektion.
key	ipsec-key	Privater Schlüssel für die Identifikation. Referenziert eine [certificate] Sektion.
pkcs11		Privater Schlüssel in PKCS#11 Store (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>: Datei, in der die Hex-ID des PKCS#11 Keys steht. <sectname>: Name der [certificate] Sektion, die auf das File mit der Hex-ID verweist <key_id>: Hex-ID des PKCS#11 Keys.
crl	ipsec-crl	Certificate Revocation List. Referenziert eine [certificate] Sektion.
ph1_encryption ph2_encryption	aes 256 aes 256	Verschlüsselungsverfahren für Phase 1 (ISAKMP-SA) und Phase 2 (IPsec-SA). Unterstützt sind: aes, twofish, blowfish, 3des

		Die Schlüssellänge kann nach dem Parameter, getrennt durch ein Leerzeichen, angegeben werden. Es dürfen nur Schlüssellängen verwendet werden, die vom Algorithmus auch unterstützt werden: aes, twofish: 128 (default), 192, 256 blowfish: 40-448 (default: 128) 3des: 168 (fix) Der 3des Algorithmus ist veraltet und sollte nicht mehr verwendet werden.
ph1_hash_alg ph2_hash_alg	sha256 sha256	Hash-Algorithmus für die Authentifizierung für Phase 1 (ISAKMP-SA) und Phase 2 (IPsec-SA). Unterstützt sind md5, sha1, sha256, sha384, sha512 md5 und sha1 werden als nicht mehr sicher erachtet.
ph1_prf		PRF Algorithmus für die Phase 1. Wenn nicht definiert wird der gleiche Algorithmus wie für das Hashing verwendet. Mögliche Werte: md5, sha1, sha256, sha384, sha512, aesxcbc, aescmac
ph1_strict ph2_strict		Wenn dieser Wert auf yes gesetzt wird, sind nur die oben definierten Algorithmen zulässig für den Tunnelaufbau. Ansonsten wird jeder vorgeschlagene Algorithmus akzeptiert. Dies sollte nur für Debugging-Zwecke gesetzt werden, um herauszufinden, wie der Peer konfiguriert ist.
ph1_lifetime ph2_lifetime	time 86400 sec time 3600 sec	Lebensdauer für Phase 1 (ISAKMP-SA) und Phase 2 (IPsec-SA). Die Lifetime definiert die Zeit, nach der neue Schlüssel generiert werden. Als Einheit können sec, min und hour verwendet werden.
dh_group	5	Diffie-Hellmann Gruppe für die Verschlüsselung, Phase 1. Je höher die Zahl, desto sicherer, desto langsamer. Möglich sind: 1, 2, 5, 14, 15, 16, 17, 18 Muss an beiden Enden der Verbindung auf den gleichen Wert gesetzt werden. Die Gruppe 1 wird als nicht mehr sicher erachtet.
pfs_group	5	Diffie-Hellmann Gruppe für die Verschlüsselung, Phase 2. Siehe dh_group. Wenn kein Wert angegeben wird, wird kein PFS gemacht (nicht zu empfehlen). PFS (perfect forward secrecy) dient dazu, dass nach einem Wechsel des Schlüssels der neue Schlüssel nicht aus dem alten abgeleitet werden kann.

4.3.21 [certificate]

Die Zertifikate, welche in der [certificate]-Sektion definiert werden, sind für IPsec und OpenVPN verwendbar. Wenn in einer [authentication] Sektion ein EAP-Server definiert wird, werden die Zertifikate ebenfalls in einer [certificate] Sektion definiert.

Attribut	Wert (Default)	Beschreibung
name		Über den Namen wird das Zertifikat identifiziert und in der Sektion [ipsec] oder [openvpn] referenziert.
type	pem	Typ des Zertifikats. Möglich sind pem, p12 und file. Verschlüsselte p12 Dateien werden nicht unterstützt.
file		Datei, in der das Zertifikat in vorliegt. Nur wirksam, wenn type=p12 oder type=file. P12 funktioniert nur mit OpenVPN.
-----BEGIN		Der Rest der Sektion zwischen den Zeilen

MIICWwIBAAKB... -----END	-----BEGIN und -----END wird als Zertifikat interpretiert wenn type=pem.

4.3.22 [openvpn]

Attribut	Wert (Default)	Beschreibung
start_server	yes	Starte den OpenVPN-Server?
start_client	no	Starte den OpenVPN-Client?
basic_server	yes	Grundlegende Optionen für einen Server.
basic_client	yes	Grundlegende Optionen für einen Client. Diese Optionen genügen für eine Verbindung zu einer IPCop Maschine.
server_net	192.168.0.0 255.255.255.0	virtuelles Netzwerk
server_remote_net	192.168.2.0/24	Netzwerke des Peers (Client). Routen in diese Netze werden auf dem Host definiert. Die Liste besteht aus Netzwerk/Präfix-Paaren, getrennt durch Leerzeichen.
push_local_net	192.168.2.0/24	Auf dem Client werden Routen in diese Netze gesetzt. Die Liste besteht aus Netzwerk/Präfix-Paaren, getrennt durch Leerzeichen.
push_default	yes	Setzt auf dem Client die Default-Route durch den Tunnel
remote	vpnsrv.example.org	IP-Adresse oder Hostname des OpenVPN Servers. Falls der Server nicht auf dem Standardport 1194 horcht, kann der Port, mit Doppelpunkt abgetrennt, angehängt werden: vpnserver.example.org:1234
client_remote_net		Netzwerke des Peers (Server). Routen in diese Netze werden auf dem Host definiert. Die Liste besteht aus Netzwerk/Präfix-Paaren, getrennt durch Leerzeichen. Hinweis: diese Routen können auch vom Server übermittelt werden, siehe push_local_net.
server_auth_method		Authentifizierungsmethode, wenn OpenVPN als Server betrieben wird. Mögliche Werte sind cert für Zertifikate und psk für Pre-Shared Key. Hinweis: ein Pre-Shared Key muss mit openvpn selber erzeugt und anschliessend in einer Zertifikats-Sektion eingetragen werden: openvpn --genkey --secret key.file
client_auth_method		Wie server_auth_method, für den Betrieb als Client.
server_cipher		Verschlüsselungs-Algorithmus. In früheren Versionen wurde BF-CBC verwendet. Dieser Algorithmus wird nicht mehr empfohlen, besser ist AES-128-CBC. Alle verfügbaren Algorithmen werden vom Befehl openvpn --show-ciphers angezeigt-
client_cipher		Wie server_cipher, für den Betrieb als Client.
server_psk		Wenn server_auth_method auf psk eingestellt ist, muss hier die Referenz auf die Certificate Sektion mit dem Pre-Shared Key eingetragen werden.
client_psk		Wie server_psk, für den Betrieb als Client.

server_cert	server-cert	Zertifikat für den Betrieb als Server. Referenziert eine [certificate] Sektion, welche nach der [openvpn] Sektion stehen muss.
server_root	server-root	Root-Zertifikat für den Betrieb als Server. Referenziert eine [certificate] Sektion, welche nach der [openvpn] Sektion stehen muss.
server_key	server-key	Schlüssel für den Betrieb als Server. Referenziert eine [certificate] Sektion, welche nach der [openvpn] Sektion stehen muss.
client_cert	client-cert	Zertifikat für den Betrieb als Client. Referenziert eine [certificate] Sektion, welche nach der [openvpn] Sektion stehen muss.
client_root	client-root	Root-Zertifikat für den Betrieb als Client. Referenziert eine [certificate] Sektion, welche nach der [openvpn] Sektion stehen muss.
client_key	client-key	Schlüssel für den Betrieb als Client. Referenziert eine [certificate] Sektion, welche nach der [openvpn] Sektion stehen muss.
server_option client_option		Zusätzliche Option, wird bei Server und Client der OpenVPN-Konfiguration angehängt. Bei Verwendung von BF-CBC wird empfohlen, diese Parameter zu verwenden, um SWEET32 Angriffe abzuwehren: client_option = reneg-bytes 64000000

4.3.23 [clientconfigfile]

Über [clientconfigfile] Sektionen können OpenVPN Client Config Files in der Konfigurationsdatei abgelegt werden. Das Attribute muss zu Beginn der Sektion stehen, anschliessend werden alle Zeilen bis zum Beginn der nächsten Sektion in die Skript-Datei kopiert. Zeilen im Skript dürfen nicht mit einer öffnenden eckigen Klammer '[' beginnen.

Attribut	Wert (Default)	Beschreibung
file		Name des Config File. Der Filename muss ohne Pfad angegeben werden. Das File wird im Verzeichnis /etc/openvpn/ccd abgelegt.

4.3.24 [tunnel]

In dieser Sektion wird ein IP-in-IP oder ein GRE-Tunnel definiert. Falls mehrere solcher Tunnels benötigt werden, muss jeder Tunnel in einer eigenen Sektion [tunnel] definiert werden.

Attribut	Wert (Default)	Beschreibung
name		Name des Tunnels. Das virtuelle Interface wird so genannt, daher darf der Name nicht tunl0, gre0, eth0 oder ppp0 heissen.
start	no	Definiert, ob der Tunnel erzeugt wird.
type	gre	Definiert den Typ des Tunnels. Mögliche Werte: ipip (für IP-in-IP Tunnel), gre (für GRE Tunnel), sit (für IPv6 in IPv4 Tunnel).
local		IP Adresse oder Interface Name des lokalen Tunnel-Endpunktes. Der entfernte Endpunkt wird ausschliesslich über diese IP-Adresse kontaktiert. Wenn die Default-Route zum Gegenüber über eine andere IP-Adresse läuft, kommt die Verbindung nicht zustande.
remote		IP Adresse Rechners, auf dem der entfernte Tunnel-Endpunkt liegt. Hinweis: diese Art von Tunnel funktioniert nicht, wenn ein

	Endpunkt hinter einem NAT-Gateway liegt. Dann muss IPsec oder OpenVPN verwendet werden.
remote_net	Netzwerkadressen der Netzwerke, welche über den Tunnel angesprochen werden. Mehrere Netzwerke werden mit einem Leerzeichen getrennt angegeben. Die Adressen werden mit Präfix angegeben, also z.B. 192.168.17.42/24
vlocal	Adresse des Tunnel-Interfaces auf dem virtuellen Netzwerk. Diese Adresse wird mit Präfix angegeben.
vremote	Adresse des entfernten Tunnel-Interfaces auf dem virtuellen Netz. Dieser Wert ist eine IP Adresse ohne Präfix.

4.3.25 [bridge]

In dieser Sektion werden Ethernet Bridges konfiguriert. Für jede Bridge wird eine eigene Sektion verwendet. Diese Regeln sind beim Einsatz von Bridges zu beachten:
- Ein Interface kann nur in höchstens einer Bridge sein
- Wenn ein Interface in einer Bridge ist, kann es nicht mehr direkt verwendet werden
Das STP-Protokoll benötigt in der Default-Einstellung ca. 50 Sekunden, bis eine Topologie-Änderung umgesetzt wird. Wenn die Werte hello=1, age=4 und fw_delay=4 auf allen an STP beteiligten Geräten gesetzt werden, kann diese Zeit auf 12 Sekunden verkürzt werden.

Attribut	Wert (Default)	Beschreibung
name		Name der Bridge. Der Name darf nicht mit einem anderen Interface kollidieren. Am besten sind die Namen br0, br1, … zu verwenden.
start	no	Die Bridge wird nur eingerichtet, wenn dieser Wert auf yes gesetzt wird.
ipaddr		IP Adresse und Netzmaske oder Präfix des Bridge Interfaces. Die Adresse kann als 192.168.1.3/24 oder 192.168.1.3 255.255.255.0 angegeben werden. Mit dem Parameter mtu:1492 wird die MTU des Interfaces festgelegt. Für dynamische Konfiguration kann dhcp angegeben werden. Wird “dhcp default” eingetragen, so setzt der DHCP Client auch die Default-Route. Weitere mögliche Parameter nach dhcp sind: metric:M setzt die Metrik der Route auf M (default:0) timeout:T setzt den Timeout auf T Sekunden (default:30) dns: Fragt den DHCP Server nach DNS Server Adressen und ersetzt die bisherigen hostname: Fragt den DHCP Server nach einem Hostnamen, und setzt diesen, falls der bisherige Hostname “localhost” ist.
iface		Liste (mit Leerzeichen getrennt) aller Interfaces, welche Teil dieser Bridge sind.
stp	no	Mit diesem Parameter kann das Spanning Tree Protokoll (STP) auf dieser Bridge gestartet werden. Alle weiteren Parameter dieser Sektion beziehen sich auf STP und sind wirkungslos, wenn hier “no” steht.
prio	32768	Die Priorität der Bridge für die Wahl des Root-Switches.
portprio		Eine Liste von Interface:Priorität Werten. Damit kann die Priorität jedes Interfaces definiert werden.

		Bsp: portprio = vlan2:34 vlan3:77
hello	2	Timer für die Hello-Pakete des STP Protokolls.
age	20	Timer für das Ageing des STP-Protokolls.
fw_delay	15	Timer für den Forward Delay des STP-Protokolls.
cost		Eine Liste von Interface:cost Werten. Damit können die Kosten der einzelnen Pfade definiert werden. Bsp: cost = vlan2:46 vlan3:84

4.3.26 [banner]

Attribut	Wert (Default)	Beschreibung
start	yes	Wenn dieser Wert gesetzt ist, wird der anschliessende Text als Message of the Day angezeigt.

Der gesamte Text dieser Sektion (ohne das start-Attribut) wird beim Login (an der Konsole und übers Netzwerk) angezeigt. Die erste Zeile, welche mit '--- END MOTD ---' beginnt oder der Beginn einer neuen Sektion beendet den Text (und wird selber nicht mehr angezeigt).

4.3.27 [daemons]

Attribut	Wert (Default)	Beschreibung
start		Definiert ein Programm, welches am Schluss des Boot-Prozesses gestartet wird. Dieses Programm kann ein Skript sein, welches über eine [script] Sektion definiert ist. Das Attribut start kann mehrfach vorkommen, die Skripte werden in der Reihenfolge gestartet, wie sie in der Konfigurations-Datei stehen.
stop		Definiert ein Programm, welches beim Herunterfahren ausgeführt wird. Hier sollten keine Programme mit langer Laufzeit angegeben werden, da der Shutdown Prozess nicht wartet, bis das Skript beendet ist.

4.3.28 [script]

Über [script] Sektionen können beliebige Skripte oder andere Textdateien in der Konfigurationsdatei abgelegt werden. Die drei möglichen Attribute müssen zu Beginn der Sektion stehen, anschliessend werden alle Zeilen bis zum Beginn der nächsten Sektion in die Skript-Datei kopiert. Zeilen im Skript dürfen nicht mit einer öffnenden eckigen Klammer '[' beginnen.

Achtung: Dateien unter /etc/scripts.d/ werden bei jedem System-Start gelöscht und neu angelegt. Dateien ausserhalb dieses Verzeichnisses werden **nicht** automatisch gelöscht, insbesondere nicht wenn die entsprechende Sektion aus der Konfiguration entfernt wird. Es ist daher nicht zu empfehlen, Dateien ausserhalb /etc/scripts.d/ anzulegen, da das schwer zu findende Nebeneffekte haben kann, wenn später die Konfiguration geändert wird.

Attribut	Wert (Default)	Beschreibung
name		Name der Sektion. Wird im Moment noch nicht verwendet.
file		Name der Skriptdatei. Wenn der Name mit einem Slash '/' beginnt, wird der Pfad als absolut betrachtet, ansonsten als relativ unter /etc/scripts.d/. Verzeichnisse, welche nicht existieren, werden automatisch angelegt.
mode		Mode für die Datei, repräsentiert als normale Unix-Dateirechte. Wenn sie ausgeführt werden soll, ist mindestens 755 nötig, sonst reicht 644. Wenn der Wert die Form Link:FILENAME hat, dann wird der Name des file Parameters als Symlink auf FILENAME angelegt, und der Rest der Sektion ignoriert. In diesem Fall muss der mode Parameter nach dem file Parameter in der Konfigurationsdatei stehen.

4.3.29 [webserver]

Diese Sektion beschreibt die Konfiguration des integrierten Webservers und des WebGUI.

Attribut	Wert (Default)	Beschreibung
start	yes	Wenn dieser Wert gesetzt ist, wird der Webserver gestartet.
port	80	TCP Port, auf dem der Webserver Verbindungen annimmt.
interface	all	Netzwerk-Schnittstellen, auf denen der Webserver verbindungen annimmt. Dieser Wert kann eine Kombination von eth0,ppp0,vlanX,brX sein, oder eine IP-Adresse, oder all für alle Schnittstellen.
document_root	/usr/share/www	Wurzelverzeichnis für den Webserver.
user		Der Webserver läuft als dieser Benutzer. Wenn nicht definiert, wird der Benutzer nobody verwendet.
group		Der Webserver läuft in dieser Gruppe. Wenn nicht definiert, wird die Gruppe nogroup verwendet.
access_log	/var/log/boa/access_log	Log-Datei des Webservers für alle Zugriffe.
error_log	/var/log/boa/error_log	Log-Datei des Webservers für alle Fehler.
default_mime		Der MIME Typ von Dateien wird aufgrund der Datei-Erweiterung festgelegt. Für fehlende oder unbekannte Extensions kann hier der MIME Typ spezifiziert werden. Für unbekannte Dateien wird text/plain verwendet. Beispiel: text/html, application/x-httpd-cgi
option		Hier angegebene Optionen werden direkt in die boa.conf geschrieben.
anygator		Wenn dieser Parameter auf yes gesetzt ist, werden die AnyGator-Skripte aktiviert.

4.3.30 [wlan]

Diese Sektion definiert die WLAN Verbindung. Sie ist wirkunslos, wenn keine WLAN-Karte im AnyRover eingebaut ist.

Die WLAN Karte kann als Client, als Access Point oder im Mesh-Modus (pre-IEEE 802.11s) betrieben werden. Gewisse Optionen in dieser Sektion sind nicht für alle Modi wirksam. In

der Tabelle sind die spezifischen Befehle markiert mit (AP) für Access Point, (CL) für Client und (M) für Mesh. Nicht markierte Optionen gelten für alle Modi.

Diese Sektion kann mehrfach vorkommen.

Es ist möglich, auf einem Access Point mehrere SSID gleichzeitig zu betreiben. Dazu muss für jede SSID eine eigene Sektion verwendet werden, wobei der Device-Name (Parameter device) für jede weitere Sektion den Namen der ersten Sektion als Präfix enthalten muss (z.B. erste Sektion: device = wlan0; zweite Sektion: device = wlan0_1). Zudem können Parameter, welche direkt mit dem Radio-Teil zu tun haben, nicht neu definiert werden (z.B. channel).
Die zusätzlichen Devices erscheinen dann als Netzwerk-Schnittstellen im System und können für Routing, Firewall-Regeln, DHCP Server usw. verwendet werden.

Im Access Point und im Client Modus werden bei Verbindungsaufbau und Verbindungsverlust jeweils alle Skripte aufgerufen, welche sich in den Verzeichnissen /etc/scripts.d/wlan-ap-hooks (für Access Point Modus) respektive /etc/scripts.d/wlan-client-hooks (für Client Modus) befinden. Diese Skripte erhalten folgende Parameter:

```
inteface cmd [clientMAC]
```

interface definiert den Namen des Interfaces, auf welchem die Aktion stattgefunden hat, cmd hat die Werte AP-STA-CONNECTED und AP-STA-DISCONNECTED im Access Point Modus, oder CONNECTED und DISCONNECTED im Client Modus. Im Access Point Modus folgt an dritter Stelle noch die MAC Adresse des Clients.

Attribut	Wert (Default)	Beschreibung
start	yes	Wenn dieser Wert gesetzt ist, wird die WLAN Verbindung gestartet. Damit die WLAN Karte betriebsbereit ist, muss die Stromversorgung auf dem USB-Bus eingeschaltet sein in der usb Sektion.
mode		Auswahl des Betriebsmodus. Mögliche Werte sind ap, client und mesh.
device		Definiert das Wireless Netzwerk Device. Für die optionale interne WLAN-Karte ist das wlan0. Wenn das Gerät als Standalone Radius Server laufen soll, muss dieser Parameter auf none gesetzt werden (und mode auf ap).
country	ch	Land,in dem das Gerät betrieben wird. Der Parameter dient dazu, die in dem Land zulässigen Kanäle auszuwählen. Mögliche Werte: ch, de, fr, us, ...
channel		Kanal, auf dem die Karte betrieben wird. Für den Client Mode kann eine Liste von Kanälen angegeben werden, dann scannt die Karte nur auf diesen Kanälen nach APs.
ipaddr	dhcp	IP Adresse und Netzmaske oder Präfix des WLAN Interfaces. Die Adresse kann als 192.168.1.3/24 oder 192.168.1.3 255.255.255.0 angegeben werden. Mit dem Parameter mtu:1492 wird die MTU des Interfaces festgelegt. (CL, M) Für dynamische Konfiguration kann dhcp angegeben werden. Wird “dhcp default” eingetragen, so setzt der DHCP

Attribut	Wert (Default)	Beschreibung
		Client auch die Default-Route. Weitere mögliche Parameter nach dhcp sind: metric:M setzt die Metrik der Route auf M (default:0) timeout:T setzt den Timeout auf T Sekunden (default:30) dns: Fragt den DHCP Server nach DNS Server Adressen und ersetzt die bisherigen hostname: Fragt den DHCP Server nach einem Hostnamen, und setzt diesen, falls der bisherige Hostname "localhost" ist.
ssid		(AP, CL) SSID des Netzwerkes, mit dem verbunden werden soll. Kann entweder ein ASCII-String oder ein Hex-Wert sein. Wenn ein Hex-Wert angegeben wird, muss er mit 0x starten.
key_management	WPA-EAP	(AP, CL) Schlüssel-Management Protokoll. Mögliche Werte sind: WPA-PSK, WPA-EAP, FT-PSK, FT-EAP, IEEE8021X (CL), NONE Die FT-* Werte stehen für Fast BSS Transition (IEEE802.11r). Mehrere Werte werden durch Leerzeichen getrennt.
pairwise	TKIP	(AP, CL) Liste von akzeptierten paarweisen Chiffren. Mögliche Werte: CCMP, TKIP, WEP104 (CL), WEP40 (CL) Wenn nichts angegeben wird, sind alle erlaubt.
wep_key		(AP, CL) WEP Schlüssel. Bis zu vier WEP Schlüssel können hier (je auf einer eigenen Zeile) eingetragen werden. Die Schlüssel können als ASCII Text oder als Hex-Werte eingegeben werden. Hex-Werte beginnen mit 0x.
wep_default_key		(AP, CL) Index des WEP-Schlüssels, der als Default-Schlüssel verwendet werden soll. Mögliche Werte sind 0-3.
eapol_version	1	(AP, CL) Viele Access Points unterstützen nur EAPOL Version 1.
scan_ssid	no	(CL) Wenn gesetzt wird das Netzwerk mit SSID-spezifischen Probe Frames gescannt. Dies wird benötigt, wenn der Access Point seine SSID nicht publiziert. Dieser Wert sollte nur auf yes gesetzt werden, wenn er benötigt wird, da die Latenz beim Scannen steigt.
scan_interval		(CL) Intervall in Sekunden, nach dem nach neuen Access Points gesucht werden muss, wenn der Client nicht mit einem Access Point verbunden ist.
scan_short scan_long scan_threshold		(CL) Intervall Parameter, nach denen nach neuen Access Points gesucht werden soll, wenn der Client mit einem Access Point verbunden ist. scan_threshold definiert den Signal-Pegel in dB (negativer Wert), der zwischen dem kurzen und dem langen Intervall unterscheidet. scan_short ist das Intervall in Sekunden, das benutzt wird, wenn der Pegel schlechter ist als der threshold. scan_long bezeichnet das Intervall in Sekunden, wenn der Pegel besser ist als threshold. Wenn nicht definiert, wird der gleich Wert wie für scan_short verwendet. Hinweis: ein Scan dauert ca. 15 Sekunden, daher ist es wenig sinnvoll, diese Intervalle auf Werte unter ca. 20s zu setzen.
pre_shared_key		(CL) Wert für den Pre-Shared Key. Kann als ASCII String oder Hex eingegeben werden. Ein Hex-Wert muss mit 0x beginnen.
eap	PEAP	(CL) Liste von EAP-Methode, durch Leerzeichen getrennt. Mögliche Werte: MD5, MSCHAPV2, OTP, GTC, TLS, PEAP, TTLS
group	TKIP	(CL) Liste von akzeptierten Gruppen-Chiffren. Mögliche Werte: CCMP, TKIP, WEP104, WEP40 Wenn nichts angegeben wird, sind alle erlaubt.
anonymous_identity		(CL) Identiät for EAP Phase1 Authentisierung.

Attribut	Wert (Default)	Beschreibung
identity		(CL) Identität für EAP Phase 2 Authentisierung.
password		(CL) Passwort für EAP Phase 2 Authentisierung.
root		(CL) Root Zertifikat für zertifikats-basierte Authentisierung. Referenziert eine [certificate] Sektion.
cert		(CL) Zertifikat für zertifikats-basierte Authentisierung. Referenziert eine [certificate] Sektion.
key		(CL) Private Key für zertifikats-basierte Authentisierung. Referenziert eine [certificate] Sektion.
pkcs11		(CL) Der Private Key im TPM (PKCS#11 Storage). Wird anstelle von key= verwendet. Syntax: pkcs11 = file:<filename>, <PKCS#11 Password> pkcs11 = cert:<sectname>, <PKCS#11 Password> pkcs11 = id:<key_id>, <PKCS#11 Password> <filename>: Das File enthält die Hex-ID des PKCS#11 Keys <sectname>: Die [certificate] Sektion sectname verweist auf das File mit der PKCS#11 Hex ID. <key_id>: die Hex-ID des PKCS#11 Keys.
phase1		(CL) Parameter für Phase 1 (äussere Authentisierung)
phase2	auth=MSCHAPV2	(CL) Parameter für Phase 2 (innere Authentisierung)
mesh_id		(M) Die Mesh-ID. Alle Stationen, welche am Mesh teilnehmen, müssen dieselbe ID haben.
wpa		(AP) Auswahl des WPA Standards. Mögliche Werte: wpa, wpa2. Wenn der Wert nicht gesetzt ist, wird kein WPA verwendet.
broadcast_ssid	yes	(AP) Wenn der Wert auf no gesetzt ist, wird die SSID nicht ausgesendet, sondern unterdrückt. Clients können dann nur verbinden, wenn sie die SSID kennen. Eignet sich nicht als Sicherheits-Element, da es einen Angreifer kaum abhält.
ieee80211d	no	(AP) Der AP übermittelt die aktive Regulatory Domain nach dem Standard IEEE 802.11d.
Ieee802.11n	no	(AP) Verwendung IEEE 802.11n. Für einen 2.4GHz access Point muss der Parameter hw_mode = g gesetzt werden, bei einem 5GHz access point auf hw_mode = a.
hw_mode		(AP) Auswahl zwischen 802.11a,b,g. Angegeben werden muss nur der Buchstabe a,b,g
txpower		(AP) Die Sendeleistung kann eingestellt werden zwischen 0dBm und 20dBm. Standardmässig ist 20dBm eingestellt.
macacl	no	(AP) Der AP kann Clients basierend auf der MAC-Adresse zulassen oder sperren. Mit diesem Parameter wird diese Funktion ein- oder ausgeschaltet. Mögliche Werte: no: Funktion ist deaktiviert; alle Clients können verbinden. accept: Alle Clients können verbinden, ausser ihre MAC-Adresse ist in der Deny-Liste eingetragen. deny: Kein Client kann verbinden, ausser seine MAC-Adresse ist in der Accept-Liste eingetragen.
acl_accept acl_deny		(AP) Liste mit zugelassenen oder gesperrten MAC-Adressen. Diese Parameter sind nur relevant, wenn macacl entsprechend gesetzt ist: macacl=accept: acl_deny ist aktiv. macacl=deny: acl_accept ist aktiv.

Attribut	Wert (Default)	Beschreibung
		Als Wert wird einzig die MAC-Adresse angegeben; bei mehreren Adressen ist für jede ein eigener Eintrag nötig. Beispiel: acl_accept = 00:11:22:33:44:55
cap_htgf		(AP) Für 802.11n. Schaltet den High Throughput Modus (Greenfield Modus) ein. Dieser Modus sollte nur verwendet werden, wenn keine 802.11b/g Clients mit dem AP verbinden, ansonsten wird das Netzwerk nicht zuverlässig funktionieren.
cap_40mhz		(AP) Für 802.11n. Schaltet die Unterstützung für 40MHz Kanäle ein. Kann auf 40- oder 40+ gesetzt werden. Wenn dieser Parameter auf 40- gesetzt ist, können bei 2.4GHz nur die Kanäle 5-13 verwendet werden und bei 5GHz die Kanäle 40,48,56,64. Wenn 40+ gesetzt wird können für 2.4GHz die Kanäle 1-7 und bei 5GHz die Kanäle 36,44,52,60 benutzt werden. Wenn kein Wert gesetzt ist, werden 20 MHz verwendet.
cap_short_gi		(AP) Für 802.11n. Schaltet die Unterstützung für Short Guard Interval ein. Das kann die Datenrate um bis zu 11% erhöhen, auf Kosten eines weniger stabilen Netzwerkes und mehr Paket-Kollisionen.
cap_rx_stbc		(AP) Für 802.11n. Definiert die Nummer der verwendeten Empfangs-Antennen. Möglich Werte: 1, 2
cap_amsdu		(AP) Für 802.11n. Schaltet Frame Aggregation ein. Das führt zu einer höheren Nutz-Datenrate.
wpa_psk		(AP) Pre-shared Key für WPA-PSK Access Point Modus. Der Schlüssel kann ein ASCII-String (8-63 Zeichen) oder ein Hex-Wert (64 Hex-Ziffern) sein. Wenn ein wpa_psk angegeben ist, werden alle wpa_psk_entry Einträge ignoriert.
wpa_psk_entry		(AP) Ein WPA-PSK Schlüssel für eine bestimmte MAC-Adresse. Dieser Eintrag kann mehrfach vorkommen. Wenn ein wpa_psk Eintrag vorhanden ist, werden diese Einträge ingnoriert. Der Eintrag enthält eine MAC-Adresse, gefolgt von einem PSK Eintrag. Als Trennzeichen dient ein Leerzeichen.
ieee8021x	no	(AP) Schaltet 802.1x ein oder aus (yes oder no)
authentication		(AP) Referenziert eine [authentication] Sektion für einen integrierten EAP-Server.
use_radius_server		(AP) Definiert, ob ein externen Radius-Server für die Authentisierung verwendet werden soll (yes oder no).
source_addr		(AP) Definiert die Source IP-Adresse, welche für die Kommunikation mit dem Radius-Server verwendet wird.
radius_ipaddr		(AP) Die IP-Adresse des Access Points, wird als NAS-IP Adresse verwendet. Wenn nichts angegeben wird, verwendet das System die IP-Adresse des unter device angegebenen Interfaces.
radius_server		(AP) IP-Adresse und Port des Radius-Servers. Wenn kein Port angegeben ist, wird 1812 verwendet. Mehrere Radius-Server können konfiguriert werden, indem dieser und der nächste Parameter mehrfach verwendet werden. Beispiel: radius_client_server = 192.168.99.4:1812
radius_secret		(AP) Das Passwort für den Zugriff auf den Radius-Server. Das Passwort gilt immer für den davorstehenden Server.

Attribut	Wert (Default)	Beschreibung
radius_accounting		(AP) IP-Adresse und Port des Accounting Servers. Wenn kein Port angegeben ist, wird 1813 verwendet. Mehrere Accounting-Server können konfiguriert werden, indem dieser und der nächste Parameter mehrfach verwendet werden. Beispiel: radius_client_accounting = 192.168.100.3:1813
radius_acct_secret		(AP) Das Passwort für den Zugriff auf den Accounting-Server. Das Passwort gilt immer für den davorstehenden Server.
radius_retry		(AP) Das Intervall in Sekunden, nachdem wieder versucht wird, den ersten Radius-Server zu erreichen. Wenn der Wert nicht gesetzt ist, werden die Server der Reihe nach verwendet, bis der jeweils aktive nicht mehr erreichbar ist.

4.3.31 [authentication]

Hier werden Authentication Server konfiguriert. Dies kann ein interner EAP-Server für einen WLAN Access Point sein, die Details eines externen RADIUS Servers, oder auch ein Standalone RADIUS Server. Diese Sektion kann mehrfach vorkommen. So können zum Beispiel ein integrierter EAP-Server für einen WLAN Access Point und ein Standalone RADIUS Server nebeneinander konfiguriert werden.

Attribut	Wert (Default)	Beschreibung
name		Eindeutige Kennung der Sektion. Dieser Wert muss gesetzt sein und an erster Stelle in der Sektion stehen.
start		Die Sektion wird nur ausgewertet, wenn dieser Parameter auf yes gesetzt ist.
standalone		Wenn in dieser Sektion ein Standalone RADIUS Server definiert wird, muss dieser Parameter auf yes gesetzt werden. Wenn die Sektion von einer wlan Sektion referenziert wird, steht hier no.
eap_phase1_id		Definiert die Parameter für die Authentifizierung in Phase 1. Syntax: eap_phase1_id = type [username[:password]] type ist TLS, TTLS oder PEAP Nach Bedarf können Username und allenfalls Passwort angegeben werden.
eap_phase2_id		Definiert die Parameter für die Authentifizierung in Phase 2. Syntax: eap_phase2_id = type [username[:passwort]] type ist MSCHAPV2, MSCHAP, CHAP, PAP für PEAP sowie mit Präfix TTLS- für TTLS
root_cert		Root Zertifikat. Referenziert eine [certificate] Sektion.
server_cert		Server Zertifikat. Referenziert eine [certificate] Sektion.
server_key		Server Private Key. Referenziert eine [certificate] Sektion.
radius_start		Wenn ein Standalone RADIUS Server definiert wird, muss dieser Wert auf yes gesetzt werden. Die folgenden Parameter werden nur verwendet, wenn radius_start auf yes gesetzt ist.
radius_addr	0.0.0.0	Adresse, auf der der RADIUS Server auf Anfragen wartet.
radius_acct_addr	0.0.0.0	Adresse, auf der der RADIUS Accounting Server auf Anfragen wartet.
radius_port	1812	UDP Port, auf dem der RADIUS Server auf Anfragen wartet.
radius_acct_port	1813	UDP Port, auf dem der RADIUS Accounting Server auf Anfragen wartet.

Attribut	Wert (Default)	Beschreibung
radius_client		Liste von IP oder Netzwerk Adressen, welche Zugriff auf den RADIUS Server haben sollen. Mehrere Adressen können durch Leerzeichen getrennt angegeben werden. Wenn mehrere Adressen mit verschiedenen Passwörtern verwendet werden sollen, kann dieser Parameter mehrfach vorkommen.
radius_secret		Das Passwort für den Zugriff auf den RADIUS Server. Dass Passwort gilt jeweils für die letzte vor dieser Zeile stehende Liste von radius_client Einträgen.

4.3.32 [ospf]

In dieser Sektion wird das Open Shortest Path First (OSPF) Routing Protokoll konfiguriert.

Attribut	Wert (Default)	Beschreibung
start	no	Wenn dieser Wert gesetzt ist, wird der OSPF Dienst gestartet.
router-id		Die Router-ID für das OSPF Protokoll. Dieser Wert kann eine IP-Adresse oder der Name eines Interfaces sein. Wenn ein Interface angegeben wird, wird dessen erste IP-Adresse als Router-ID verwendet. Im Falle des Loopback-Interfaces ist das dann 127.0.0.1.
insert-default		Wenn dieser Wert gesetzt ist, wird eine im System vorhandene Default-Route über OSPF verbreitet.
area		Definiert eine OSPF-Area. Die Definition besteht aus der Area-Nummer und einer komma-separierten Liste von Interfaces, die zu dieser Area gehören. Für jede zu definierende Area wird eine eigene Zeile verwendet. Beispiel: area = 0, vlan1, vlan2
stub		Definiert eine OSPF Stub-Area. Die Definition erfolgt gleich wie beim Parameter area.
passive		Das Netzwerk des hier angegebenen Interfaces wird über OSPF verbreitet, aber das OSPF-Protokoll wird auf diesem Interface nicht ausgeführt. Um mehrere passive Interfaces zu definieren, kann dieser Parameter mehrfach vorkommen.
auth		Authentisierungs-Optionen. Die Optionen bestehen aus einer Area Nummer, dem Authentisierungstyp und einer komma-separierten Liste aus Interface:[id:]Key Tupeln. Der Authentisierungstyp kann key oder md5 sein. Der Schlüssel wird definiert als Interface:Key für den key Typ, oder Interface:id:key für den md5 Typ. Die ID muss in allen Routern im Netzwerk konsistent sein. Beispiele: auth = 0, key, vlan1:verysecret, vlan2:evenmoresecret auth = 1, md5, vlan2:2:unbreakable
range		Routen-Zusammenfassung. Dieser Parameter ist nur auf ABR (Area Border Router) gültig. Wenn mehrere kontinuierliche Netzwerke innerhalb einer Area vorhanden sind, kann der ABR diese als ein Netz in andere Areas weitergeben. Die Definition besteht aus einer Area-ID und einer Liste von zusammengefassten Netzwerken. Beispiel: range = 1, 192.168.64.0/22

4.3.33 [snmp]

In dieser Sektion wird das Simple Network Management Protocol (SNMP) definiert. Der

AnyRover enthält einen SNMP Agent, der SNMP Requests beantwortet und auch SNMP Traps auslösen kann.

Attribut	Wert (Default)	Beschreibung
start	no	Wenn dieser Wert gesetzt ist, wird der SNMP Dienst gestartet.
listen		Definiert, auf welchen Interfaces und Ports SNMP Requests empfangen werden sollen. Syntax: listen = [proto[:]][interface/address[:]][port] proto kann entweder tcp or udp sein, als Interface kann der Name eines Interfaces (z.B. Eth0) oder eine IP-Adresse angegeben werden. Default ist udp:0.0.0.0:161 Mehrere Kombinationen können durch Komma getrennt aufgelistet werden.
sysdescr		Text, welcher von SNMP in sysDescription.0 gemeldet wird.
location		Text, welcher von SNMP in sysLocation.0 gemeldet wird.
contact		Text, welcher von SNMP in sysContact.0 gemeldet wird.
services		Liste von Diensten, welche das Gerät anbietet. Wird von SNMP in sysServices gemeldet. Mögliche Werte sind: physical, datalink/subnet, internet, endtoend, application Anstatt der Dienste kann auch eine Zahl angegeben werden, wobei die Werte 1, 2, 4, 8, 64 entsprechend den Diensten addiert werden müssen.
view		Mit Views kann der Zugriff auf einzelne Unterbäume von OIDs limitiert werden. Ein View muss definiert werden, bevor sie in einem User Attribut verwendet wird. Der erste Parameter ist der Name der View, der zweite definiert, ob die angegebene OID der View hinzugefügt (included) oder entfernt (excluded) wird. Der dritte Parameter definiert die OID. Syntax: view = NAME, [included | excluded], OID
user		Benutzer-Management. Mit diesem Parameter werden SNMP-Benutzer angelegt. Syntax: SNMP v1 und v2c: user = SNMP vers,{ro | rw},community[,src[,OID]] SNMP v3: user = SNMP vers,{ro | rw},user:pw1[:pw2][,hash:enc[,priv[,OID | View]]] Für SNMP Version 1 und 2 können noch zwei optionale Parameter angegeben werden: src: Source Adresse oder Netzwerk, von wo die SNMP-Requests zugelassen sind OID: Limitiert den Zugriff auf den Unterbaum, der an dieser OID hängt. Mehrere OIDs können über eine View zusammengefasst werden. SNMP Version 3: Wird nur pw1 angegeben, gilt es für Authentication und Privacy, sonst gilt pw1 für Authentication und pw2 für Privacy. Beide Passwörter müssen mindestens 8 Zeichen lang sein. Mit hash und enc können die Algorithmen für Authentication und Privacy definiert werden. Das Flag priv kennt die Werte priv, auth oder noauth. Damit können Privacy, Authentication oder keines von beiden zwingend gefordert werden. Ohne Angabe des Flags sind Authentication und Privacy freiwillig. hash: MD5 (default), SHA enc: DES, AES (default)

Attribut	Wert (Default)	Beschreibung
process		Prozess-Beobachtung. process = name [, max [, min]] Der Prozess min dem angegebenen Namen muss in der Prozesstabelle zwischen min und max Mal vorhanden sein, ansonsten wird das Error-Flag gesetzt.
exec sh extend		Ausführen von beliebigen Programmen exec = [OID,] name, path [,arg [,arg]] Wenn die OID abgefragt wird, führt SNMP dieses Programm aus und liefert die Parameter als Werte zurück. Wenn eine OID angegeben wird, sind die Informationen an dieser Stelle im Baum verwurzelt, und die gesamte Ausgabe des Programmes wird zurückgeliefert. Ohne OID sind die Informationen in der extTable angesiedelt, und nur die erste Zeile der Ausgabe wird zurückgeliefert. Der Befehlt exec wird für binäre Programme verwendet, sh für Shell-Skripte. Extend ist eine verbesserte Variante von exec und sh, welche die Ausgaben doppelt liefert: einmal als ein String, und einmal zeilenweise.
trapcommunity		Definiert die Default-Community für Traps.
trapagent		Intern wird für die Abfrage der Werte für Traps ein SNMPv3 Request verwendet mit dem hier angegebenen Benutzernamen. Ein solcher Benutzer muss definiert sein.
trapsink trap2sink		Ziel-Adresse für Traps. trapsink = [tcp | udp]:(IP addr | hostname)[:port][,community] Wenn keine Community angegeben wird, benutzt SNMP die Community, welche unter trapcommunity definiert wurde. Das default Protokoll ist UDP, der default Port ist 162.
authfail		Definiert, ob Authentication Failure Traps gesendet werden sollen oder nicht (yes oder no)
updown		Definiert, ob Interface up/down Traps gesendet werden sollen oder nicht (yes oder no)
monitor		Definiert eine Überwachung eines MIB Objektes monitor = name, expr [, action [, user [, freq [, oid [, oid]]]]] Der name muss für jeden Monitor eindeutig sein. Die expr hat die Form: OID | !OID | !=OID | OID OP value | OID min max wobei OP ein Operator aus dieser Liste ist: ==, !=, <, <=, >, >= Action ist der Name einer Aktion, die ausgeführt werden soll, wenn der Monitor anschlägt. Siehe unten. Freq definiert das Intervall in Sekunden zwischen zwei aufeinanderfolgenden Test der expr (Default: 600) Wenn keine Aktion angegeben wird, löst der Daemon eine Standard Notification aus. Wenn die Aktion eine Notification ist (entweder direct oder über eine action), können mit nachfolgenden OIDs weitere Werte zur Übermittlung angefügt werden.
action		Definiert eine Aktion, die durch einen Monitor getriggert werden kann. action = name, type, value [, oid [, oid]] Der Name wird im monitor Attribut verwendet und dient zur Identifizierung. Type ist entweder set oder notify. Wenn type = set ist, dann hat value die Form OID = Wert, und der ensprechende OID wird gesetzt. Weitere Parameter werden ignoriert. Für type = notify ist value der Typ der Notification Meldung:

Attribut	Wert (Default)	Beschreibung
		coldStart, warmStart, linkDown, linkUp, authenticationFailure, egpNeighborLoss, enterpriseSpecific Weiter angegebene OIDs werden in der trap Meldung ebenfalls übermittelt.

4.3.34 [dns]

Diese Sektion definiert die DNS Proxy und Server Einstellungen. Ein DNS Server wird aktuell noch nicht unterstützt.

Attribut	Wert (Default)	Beschreibung
start_proxy	no	Wenn dieser Wert gesetzt ist, wird der DNS Proxy gestartet.
proxy_basic		Wenn der Wert auf yes gesetzt ist, werden ein paar grundlegende Parameter eingeschaltet: Der Proxy blockiert unnötige Anfragen von Windows, und Adressen im privaten Adressbereich sowie reine Hostname ohne Domain werden nicht weitergeleitet. Wenn Kerberos, SIP, XMMP oder Google-talk verwendet wird, muss dieser Wert auf no gesetzt werden.
proxy_interface		Liste (mit Komma getrennt) aller Interfaces, auf denen der DNS Proxy Anfragen entgegennehmen soll. Wenn die Liste mit einem '/' anfängt, spezifiziert sie die Interfaces, auf denen keine Anfragen entgegengenommen werden sollen. Wenn nichts definiert wird, nimmt der Proxy alles an. Dieser Parameter sollte nich zusammen mit proxy_address verwendet werden.
proxy_address		Liste der IP-Adressen, auf denen der DNS Proxy Anfragen entgegennimmt. Wenn nichts definiert wird, nimmt der Proxy alles an. Dieser Parameter sollte nicht zusammen mit proxy_interface verwendet werden.
proxy_port		Port, auf dem der Proxy DNS Anfragen entgegennimmt. Default: 53
proxy_domain		Wenn gesetzt, wird dieser Domainname allen reinen Hostnamen angehängt, bevor sie zum DNS-Server geschickt werden.
proxy_param		Zusätzliche Parameter können hier gesetzt werden. Ein paar nützliche Parameter sind: - strict-order: Die Name-Server werden strikt in der Reihenfolge, wie sie im /etc/resolv.conf stehen, abgefragt. - all-servers: Alle Name-Server werden gleichzeitig angefragt, anstatt nacheinander, bis einer die Antwort liefert.
static_host		Mit diesem Parameter können statische DNS Einträge definiert werden, so dass kein Lookup stattfindet. Syntax: static_host = hostname, IP address Beispiel: static_host = google-public-dns-a.google.com, 8.8.8.8

4.3.35 [serports]

Diese Sektion konfiguriert die seriellen Schnittstellen.

Attribute	Value (Default)	Description
enable	no	Aktiviert oder deaktiviert die seriellen Schnittstellen.

4.3.36 [openconnect]

In dieser Sektion wird der OpenConnect Client konfiguriert, der eine VPN Verbindung zu einem Cisco AnyConnect SSL VPN Gateway erlaubt.

Openconnect wird nicht offiziell von Cisco unterstützt und ist nicht mit Cisco verbunden.

Attribute	Value (Default)	Description
start		Definiert, ob openconnect gestartet werden soll oder nicht.
remote		Adresse des VPN Endpunktes. URL in dier Form https://server.example.org oder https://192.168.20.12
username		Benutzername für das Login auf dem Server
password		Passwort für das Login auf dem Server
check_certificate		Openconnect kontrolliert das Server-Zertifikat und fragt um Bestätigung, wenn es dieses nicht selber verifizieren kann. Mit diesem Parameter kann diese Kontrolle ausgeschaltet werden (check_certificate = no).

4.3.37 [mobileip]

Der AnyRover kann als Mobile Node eine MobileIP Verbindung zu einem Home Agent (HA) aufbauen. MobileIP ist eine VPN-Technologie, welche den Tunnel sehr schnell (innerhalb von Sekunden) auf einen neuen Link umschalten kann, wenn der bisher verwendete Link nicht mehr verfügbar ist, oder wenn ein höher priorisierter Link verfügbar wird.

Die Daten im MobileIP Tunnel sind nicht verschlüsselt, für diese Zwecke wird in der Regel ein zusätzlicher IPsec Tunnel verwendet.

Der Mobile Node führt eine Liste von verfügbaren Schnittstellen für die Verbindung zum HA. Wenn die aktuell verwendete Verbindung nicht mehr funktioniert, probiert er die anderen möglichen Routen der Reihe nach durch, bis er eine neue findet. Daher kann es bei wachsender Liste möglicher Uplinks auch länger dauern, bis MobileIP umgeschaltet hat. Es ist mit etwa 3 Sekunden pro getestetem Link zu rechnen.

MobileIP started alle Hook-Skripte in /etc/scripts.d/mip-hooks/ wann immer es sich erfolgreich beim Home Agent registriert hat. Die Skripte werden mit den Parametern „(RE)CONNECT“ und dem Namen des verwendeten Interfaces gestartet. Bei der ersten Registrierung ist der erste Parameter „CONNECT“, und IPsec wird angekickt, bei jeder weiteren Registrierung ist der erste Parameter „RECONNECT“. Diese Skripte können verwendet werden, um gewünschte Routen über den MobileIP Tunnel aufzusetzen.

Attribute	Value (Default)	Description
start		Definiert, ob MobileIP gestartet werden soll oder nicht.
mode	mn	Definiert den Modus, in dem MobileIP laufen soll. Zur Zeit wird nur

Attribute	Value (Default)	Description
		Mobile Node unterstützt. mode = mn
ha		Die IP-Adresse des Home Agent
hoa		Die Home Address des Mobile Nodes.
ign_interface		Eine Liste von Netzwerk-Schnittstellen (mit Komma separiert), welche von MobileIP nicht verwendet werden, um den HA zu kontaktieren. Die Schnittstellen lo, tunl0 und gre0 werden per Default ignoriert. Wenn sie trotzdem verwendet werden sollen, können sie mit einem führenden Slash gelistet werden (/gre0).
routing	default	Definiert das Routing, welches nach erfolgreichem Tunnel-Aufbau konfiguriert werden soll. Mögliche Werte: default, none, {Netzwerk}. Default: Eine Default-Route über den Tunnel wird eingerichtet. None: Keine zusätzlichen Routen werden konfiguriert, allfällig nötige Routen müssen über zusätzliche Scripte aufgesetzt werden. Wenn eine Netzwerk-Adresse angegeben wird, setzt das System eine Route zu diesem Netzwerk über den MobileIP Tunnel.
spi		Security Parameter Index, definiert die Security Association auf dem HA für diese Verbindung. Kann dezimal oder hexadezimal (mit führendem 0x) angegeben werden. Beispiel: spi = 0x10a
auth	hmac-md5	Authentisierungs-Algorithmus. Mögliche Werte: md5-prefix-suffix, hmac-md5, sha1, hmac-sha1 Hinweis: md5-prefix-suffix funktioniert nicht bei Verbindungen zu Cisco Home Agents, in diesem Fall ist hmac-md5 zu verwenden. Zudem hat md5-prefix-suffix bekannt Schwachstellen.
secret		Das Shared Secret für die Tunnel Authentisierung beim Home Agent. Gemäss RFC2002 ist das Shared Secret 16 oder 32 Byte lang, allerdings werden auch andere Längen unterstützt. Das Secret kann als Text oder als Hexadezimalzahl (mit Prefix 0x) angegeben werden.
replay	timestamp	Methode für Replay Protection. Mögliche Werte: none, timestamp, nonces
lifetime	3600	Tunnel Life Time: Zeit in Sekunden bis zur nächsten Re-Registrierung. Werte >= 65535 bedeuten unendlich, d.h. Keine neue Registrierung.
udpport	434	UDP Port an den die Registrierungsanfragen gesendet werden. Standard-Port ist 434.
udpsrcport		UDP Source Port für die Kommunikation mit dem Home Agent. Wenn der Parameter nicht gesetzt ist, wird ein zufälliger Port benutzt.
interval	200	Tunnel-Keepalives. Ein aktiver Tunnel wird regelmässig geprüft auf Verfügbarkeit. Dieses Intervall definiert das minimale Intervall zwischen zwei Keepalive-Pings in Millisekunden.
linkdown	3	Ein Tunnel wird als nicht mehr verfügbar angeschaut, wenn die hier definierte Anzahl an aufeinanderfolgenden Pings nicht beantwortet wurde.
tunnel_rtt	500	Die initiale Round-Trip-Time im Tunnel in Millisekunden. Die RTT wird im laufenden Betrieb ständig den aktuellen Werte angepasst, allerdings nie unter 200ms gesenkt.

Attribute	Value (Default)	Description
percentage	120	Wenn eine Antwort auf ein Keep-Alive Paket nicht innerhalt des hier angegebenen Prozentsatzes der RTT empfangen wird, dann wird es als verloren betrachtet.
link_priority		Der Mobile Node führt eine Liste der aktuell verfügbaren Default Routen, sortiert nach der Routing Metrik. Wenn die Link Priority aktiviert wird, dann prüft der Mobile Node höher priorisierte Routen (tiefere Metrik) als die aktuell verwendete auf Verfügbarkeit, und schaltet gegebenenfalls um. Wenn die Link Priority nicht verwendet wird, dann schaltet der Mobile Node nur auf einen anderen Link um, wenn der aktuell verwendete nicht mehr verfügbar ist.
link_prio_icmp	yes	Dieser Parameter hat nur einen Effekt wenn link_priority=yes gesetzt ist. Wenn dieser Parameter gesetzt ist, werden ICMP Echo Requests zum HA geschickt, um zu prüfen, ob eine besser priorisierte Route verfügbar ist.
link_prio_reg_valid	no	Dieser Parameter hat nur einen Effekt, wenn link_priority=yes und link_prio_icmp=no gesetzt sind. Für die Prüfung der Verfügbarkeit einer höher priorisierten Route zum HA werden Registration Requests gesendet. Dieser Parameter definiert, ob diese gültig sein sollen oder nicht. Gültige Registration Requests führen dazu, dass der HA den Tunnel sofort auf den neuen Link umschaltet, und der Tunnel somit nicht verfügbar ist, bis der Mobile Node ebenfalls umgeschaltet hat. Dieses Verhalten des HA verträgt sich nicht mit der möglichen Umschalt-Verzögerung, welche mit den nächsten zwei Parametern konfiguriert werden kann. Ungültige Registration Requests haben einen 10 Jahre alten Zeitstempel und werden vom HA mit einer Fehlermeldung beantwortet. Für den Test, ob der Link verfügbar ist, reicht dies jedoch vollkommen aus. Sobald der Mobile Node eine solche Fehlermeldung erhält, schaltet er aktiv um und schickt einen gültige Registration Request über diesen Link.
link_count	2	Dieser Parameter definiert, wieviele erfolgreiche Antworten der Mobile Node über einen höher priorisierten Link erhalten muss, bis er umschaltet. Zusammen mit dem nächsten Parameter kann so definiert werden, wie schnell der Mobile Node umschalten soll, wenn ein neuer und besserer Link verfügbar wird.
link_interval	2	Dieser Parameter definiert das Intervall für Keep Alive Meldungen über höher priorisierte Links. Zusammen mit dem vorherigen Parameter kann so definiert werden, wie schnell der Mobile Node umschalten soll, wenn ein neuer und besserer Link verfügbar wird.

4.3.38 [scep]

Das Erfassen von Zertifikaten (z.B. für IPsec) direkt in einer Zertifikats-Sektion ist unpraktisch, wenn das Gerät über die Gültigkeitsdauer der Zertifikate hinaus betrieben werden soll, und das Ersetzen der Zertifikate von Hand unpraktisch ist. Dann können die Zertifikate automatisch via SCEP (Simple Certificate Enrollment Protocol) bezogen und entsprechend auch erneuert werden, bevor sie ablaufen.

Pro Satz Zertifikate wird eine [scep] Sektion verwendet; entsprechend kann die Sektion

mehrfach vorkommen.

Wenn ein SCEP-Request gestartet wird, dann lädt er zuerst die CA-Zertifikate, erzeugt einen privaten Schlüssel, und daraus dann einen Certificate Signing Request (CSR), welcher er dem SCEP-Server zur Signierung vorlegt und so das eigentliche Zertifikat erhält.

Nachdem ein SCEP-Request ausgeführt wurde, werden die Hook-Skripte ausgeführt, welche in /etc/scripts.d/scep-hooks/ liegen. Für jede Sektion spezifisch werden die Skripte in /etc/scripts.d/scep-hooks/<name>/ ausgeführt, wobei <name> dem Namen der Sektion entspricht.

Den Hook-Skripten werden über Umgebungs-Variablen verschiedene Werte mitgegeben. Folgende Parameter sind gesetzt:

SCEP_NUMCERT = Anzahl zu erneuernder Zertifikate
SCEP_SUCCESS = Anzahl erfolgreich erneuerte Zertifikate
SCEP_TIMEOUT = Anzahl Zertifikate, welche infolge von Server-Timeout nicht erneuert werden konnten
SCEP_SKIPPED = Anzahl Zertifikate, welche noch nicht erneuert werden müssen.

Das Überprüfen von vorhandenen Zertifikaten auf das Ablaufdatum ist eine sehr günstige Operation, welche nur ein paar Zehntelssekunden dauert und daher problemlos auch regelmässig ausgeführt werden kann.

Erst das Erneuern selber ist dann eine aufwändige Operation, welche schnell über eine Minute dauern kann. Allerdings ist die Aufgabe nicht besonders rechenintensiv; bloss das erzeugen des neuen Private Keys dauert, weil genügend Zufallszahlen vorhanden sein müssen.

Attribute	Value (Default)	Description
name		Name der Sektion. Dieser Name wird für das Config-File verwendet und muss für jede [scep] Sektion eindeutig sein. Dieser Parameter muss vor dem start-Parameter stehen.
start		Definiert, ob diese Sektion aktiv ist.
check		Definiert den Zeitplan, wann der SCEP-Client ausgeführt werden soll. Grundsätzlich sind die Werte Einträge in die Crontab und haben dieselbe Syntax. Der Cron Daemon wird automatisch gestartet, auch wenn in [crontab] start = no gesetzt ist. Zusätzlich können folgende Einträge verwendet werden: - daily TIME: tägliche Kontrolle zur vorgegebenen Zeit. - weekly DAY TIME: wöchentliche Kontrolle am angegebenen Tag zu einer bestimmten Zeit. Wochentage sind auf englisch und in Kleinbuchstaben anzugeben. - Bestimmte Ereignisse: ppp-up wenn die 3G/4G Verbindung erstellt wurde (nur ppp), mip-up wenn MobileIP verbunden hat (nur beim ersten erfolgreichen Verbinden, nicht bei jedem Medium-Wechsel), ipsec-up wenn der IPsec Tunnel aufgebaut worde ist, dhcp <if> wenn das Interface <if> eine IP-Adresse bezogen hat, boot wenn der Bootprozess abgeschlossen ist, wlan <if> wenn das Wireless Client-Interface <if> mit dem AP

Attribute	Value (Default)	Description
		verbunden ist. Syntax: on <EVENT> Beispiele: check = 1 15 * * * check = daily 9:30 check = weekly thursday 11:23 check = on boot check = on wlan wlan0
action		Vordefinierte Aktionen, welche bei erfolgreichem Enrollment ausgeführt werden sollen: - ipsec: Lädt alle IPsec Verbindungen neu, so dass neue Zertifikate übernommen und verwendet werden. - bondix: Lädt alle Bondix Tunnel neu
debug		Wenn dieser Wert auf yes gesetzt wird, dann loggt der SCEP Client ziemlich intensiv. Wird nur für Debugging-Zwecke empfohlen.
directory	/etc/certs/ipsec	Verzeichnis, wo die neuen Zertifikate gespeichert werden. Der Request kann auch gestartet werden, wenn dieses Verzeichnis leer ist; dann werden zuerst die CA-Zertifikate geladen.
days	7	Anzahl Tage vor Ablauf, wann eine Erneuerung gestartet werden soll.
key_size		Grösse in Bit des privaten Schlüssels, sofern nicht vorhanden. Mögliche Werte: 768, 1024, 2048, 4096. Für PKCS#11 (TPM) Keys sollte 2048 verwendet weden.
key_type		Definiert den Typ der Private Keys. Mögliche Werte: rsa: File-basierter Schlüssel tpm: TPM basierter Schlüssel (geht nicht mit IPsec und WLAN) pkcs11: Schlüssel im PKCS#11 Store (TPM basiert). Funktioniert mit IPsec und WLAN.
signature		Algorithmus für die Key-Signierung. Mögliche Werte: md5, sha1, sha224, sha256, sha384, sha512
server		URL des SCEP-Servers. Für Microsoft-Server sieht die in der Regel wie folgt aus: http://<IPADDR>/certsrv/mscep/mscep.dll
fetch_ca		Wenn dieser Wert auf yes gesetzt ist, lädt SCEP bei jedem Request die CA-Zertifikate vom Server. Ansonsten nur, wenn sie lokal nicht verfügbar sind.
encryption		Verschlüsselung, die bei der Kommunikation mit dem SCEP-Server verwendet wird. des, 3des, blowfish
ca-file		Name der Datei, in welche das CA-Zertifikat geschrieben wird. Wenn der Server mehrere Zertifikate liefert, werden sie mit -2, -3 usw ergänzt. Zudem wird eine Datei namens enc-<file> erzeugt.
password		Challenge-Passwort, welches für die Kommunikation mit dem SCEP-Server verwendet wird.
old_cert_auth		Wenn dieser Wert auf yes gesetzt ist, werden die aktuellen Zertifikate anstelle eines Passwortes für die Authentisierung beim SCEP Server verwendet.
CA-DN		Distinguished Name des zu verwendenden CA-Zertifikats.
cert-file		Name der Datei, in welche das Zertifikat abgelegt wird.
key-file		Name der Datei für den privaten Schlüssel. Wird mit Mode 0600 erzeugt.
keep		Wenn dieser Wert auf yes gesetzt ist, bleibt der Certificate Signing Request (CSR) nach Erhalt der Zertifikate erhalten,

Attribute	Value (Default)	Description
		ansonsten wird er gelöscht.
Country State Location Organization OrgUnit CommonName Email		Felder für den Distinguished Name des Zertifikats.
altname		Alternativer Name des Zertifikats.
quote_altname		Wenn dieser Wert auf yes gesetzt ist, wird der subjectAlternativeName im CSR in Anführungszeichen gesetzt.

4.3.39 [pelix]

Der AnyRover kann Positionsdaten an einen Pelix-Server der Firma LogObject übermitteln. Weitere Funktionen im Zusammenhang mit Pelix Servern werden noch nicht unterstützt.

Daten, die an den Pelix-Server gesendet werden, sind zwar mit dem Pelix-Protokoll kodiert, aber nicht verschlüsselt. Eine Verschlüsselung der Daten muss mit anderen Methoden sichergestellt werden (z.B. IPsec).

Attribute	Value (Default)	Description
start		Definiert, ob diese Sektion aktiv ist.
listen		Definiert eine Schnittstelle, auf der GPRMC Daten empfangen werden können. Default-Wert: listen = tcp, 127.0.0.1:13181 Um diese Werte liefern zu können, muss analog ein Konfigurations-Eintrag für die Übermittlung der Daten vorhanden sein: [gps] target = 0, 127.0.0.1:13181
target		IP-Adresse und TCP-Port des Pelix-Servers. Zur Zeit kann nur ein Server angegeben werden. Beispiel: target = 192.168.1.1:11310
source		Source-Adresse und Port, mit denen die Verbindung zum PELIX Server hergestellt werden soll. Syntax: source = addr[:port] Beispiel: source = 192.168.1.3
retry		Wartezeit in Sekunden, bis im Fehlerfall der nächste Verbindungsversuch gestartet wird. Default: 5
interval		Intervall in Sekunden für die Übermittlung der Position. Default: 10
coordinates		Definiert das Format der Koordinaten, wie es an den Pelix-Server übermittelt wird. Mögliche Werte: CH1903, WGS84 microdegrees, WGS84 Default: WGS84 microdegrees
id		ID des Gerätes gegenüber dem Pelix-Server. Hier wird

Attribute	Value (Default)	Description
		üblicherweise die IMEI des Modems verwendet. Die IMEI muss zurZeit noch manuell eingetragen werden.
username		Username für das Login beim Pelix-Server.
password		Passwort für das Login beim Pelix-Server.
retransmit		Definiert, ob Meldungen, welche vom Pelix-Server nicht bestätigt wurden, erneut gesendet werden. Default: no

4.3.40 [login]

Attribute	Value (Default)	Description
passwd		Wenn dieser Wert auf yes gesetzt ist, wird ein Login Passwort gegen das lokale Passwort-File geprüft, allerdings erst nach Prüfung von RADIUS und TACACS+ Servern. Achtung: wenn der Wert auf no gesetzt und keine andere Login Methode aktiviert wird, ist ein Login auf dem AnyRover nicht mehr möglich.
passwd_console		Wie passwd, aber nur für Konsolen-Logins.
passwd_ssh		Wie passwd, aber nur für SSH Logins.
tacacs_blacklist radius_blacklist	root, config	Blackliste für Radius oder TACACS+ Logins. Dieser Parameter muss zwingend an erster Stelle stehen. Alle User, die hier angegeben werden, werden nicht vom Radius oder TACACS+ Server überprüft, sonder nur lokal. Der Parameter wird nur angewendet, wenn weiter unten ein Radius oder TACACS+ Server konfiguriert wird.
radius		Login Informationen für Radius-Server. Syntax: radius = server, key[, timeout] server: IP Adresse oder Hostname des Radius Servers key: Shared Secret des Radius Servers timeout: Wenn der Server in dieser Zeit (in Sekunden) nicht antwortet, wird der nächste Eintrag probiert.
radius_console		Login Informationen für Konsole-Login via Radius-Server
radius_ssh		Login Informationen für SSHLogin via Radius-Server
tacacs		Login Informationen für TACACS+ Server Syntax: tacacs = server, key[,[timeout][,source[:port]]] server: IP Adresse oder Hostname des TACACS+ Servers. key: Shared Secret des TACACS+ Servers timeout: Wenn der Server in dieser Zeit (in Sekunden) nicht antwortet, wird der nächste Eintrag probiert. source: Source IP-Adresse für die Kommunikation mit dem TACACS+ Server. port: Source Port für die Kommunikation mit dem TACACS+ Server.
tacacs_console		Login Informationen für Konsole-Login via TACACS+ Server
tacacs_ssh		Login Informationen für SSH-Login via TACACS+ Server
allow_unknown	yes	Mit diesem Parameter kann bestimmt werden, ob ein User, der nicht auf dem System konfiguriert wurde, jedoch über TACACS+ oder Radius authentisiert wird, einloggen kann. Der User wird

Attribute	Value (Default)	Description
		beim Login angelegt und nachdem die letzte Session beendet wurde, wieder gelöscht.
allow_unknown_console		Beschränkt das Login mit unkonfiguriertem User auf die Konsole.
allow_unknown_ssh		Beschränkt das Login mit unkonfiguriertem User auf SSH.
tally_count	5	Wenn dieser Parameter gesetzt wird, wird nach der konfigurierten Anzahl fehlgeschlagener Logins der Account blockiert. Die Funktion ist aktiv, sobald ein Wert >0 gesetzt ist. Der Zähler wird bei erfolgreichem Login oder mit dem Befehl pam_tally2 zurückgesetzt.
tally_root	yes	Definiert, ob der oben definierte Zähler auch für den User root gilt.
tally_timeout	30	Definiert für wieviele Sekunden der Account gesperrt bleibt. Wenn kein Wert gesetzt wird, wird der Account für immer gesperrt, oder solange bis ein reset mit „pam_tally2“ durchgeführt wird.

4.3.41 [802.1X]

Attribute	Value (Default)	Description
name	vlan2	Definiert das Interface, auf welchem 802.1x aktiv sein soll. Zusätzlich muss im betreffendem Interface bei ipaddr das keyword 8021x stehen.
mode	client	Definiert, ob 8021x als Client (Supplicant) oder Server (Authenticator) genutzt wird.
eapol_version	1	Definiert die Eapol-Version
key_management	EAP	Schlüssel-Management Protokoll. Mögliche Werte sind: PSK und EAP
eap	PEAP	Liste möglicher EAP -Methoden. Mehrere Werte werden mit einem Komma getrennt. Mögliche Werte sind: PEAP, TLS, TTLS, MD5, MSCHAPV2.
pre_shared_key		Pre Shared Key bei PSK-Authentifizierung
anonymous_identity		Username bei EAP Phase 1 Authentisierung.
identity		Username bei EAP Phase 2 Authentisierung.
password		Passwort bei EAP Phase 2 Authentisierung.
phase1		Phase 1 Parameter (bei TLS Tunnel). Beispielwert: peapver=0
phase2		Phase 2 Parameter (bei TLS Tunnel). Beispielwert: auth=MSCHAPV2
root	8021x-root	Root-Zertifikat für die Identifikation. Referenziert eine [certificate] Sektion.
cert	8021x-cert	Zertifikat für die Identifikation. Referenziert eine [certificate] Sektion.
key	8021x-key	Privater Schlüssel für die Identifikation. Referenziert eine [certificate] Sektion.
pkcs11		Privater Schlüssel für die Identifikation im PKCS#11 (TPM) Store. Syntax gleich wie in der [wlan] Sektion.

Attribute	Value (Default)	Description
route		Nur für Authenticator, wenn das Interface Teil einer Bridge ist. Wenn dieser Wert auf yes gesetzt ist, werden EAPOL Pakete auf dem physikalischen Interface empfangen. Bei no werden sie auf dem Bridge-Interfaces empfangen. Beispiel: vlan1 (Switch Port 1) und vlan2 (Switch Port 2) werden über 802.1X geschützt. Die Schnittstellen sind im gleichen IP Subnetz und darum beide Teil der Bridge br0. Der Authenticator läuft auf der vlanX Schnittstelle, der DHCP-Server auf der br0 Schnittstelle. In diesem Fall wird route=yes gesetzt, dann ercheinen die EAPOL Pakete auf vlanX, alle anderen Pakete (v.a. IP Daten) auf br0.
port_mode		Definiert den Modus, in dem der Port laufen soll (nur für Authenticator). Mögliche Werte: single-host: Nur ein Host kann authentisiert sein. Wenn sich ein zweiter Host authentisiert, wird der erste verworfen. multi-host: Wenn sich ein Host erfolgreich authentsiert hat, ist der Port für alle Clients geöffnet. Beispiel: ein weiterer 802.1X Switch ist am Port angeschlossen. Der Switch authentisiert sich am AnyRover, die weiteren Clients am Switch. multi-auth: Mehere Clients können sich unabhängig voneinander authentsieren. Beispiel: ein nicht-802.1X Switch ist am Port angeschlossen. Wenn das Netzwerk-Kabel entfernt wird, ist der Port für alle Clients wieder geschlossen.
mab		Mac Authentication Bypass. Hier können MAC Adressen angegeben werden, die keine Authentisierzung benötigen, weil der Client kein 802.1X unterstützt (z.B. ein Drucker).
standalone		Wenn dieser Parameter auf yes gesetzt wird, können im Anschluss lokale Anmelde-Daten konfiguriert werden
eap_phase1_id		Daten für EAP Phase 1 Authentisierung. Syntax: TYPE [user[:password]] Mögliche Werte für TYPE: PEAP, TTLS, TLS Wenn kein Username angegeben wird, gilt die Methode für alle eintreffenden Anfragen.
eap_phase2_id		Daten für EAP Phase 2 Authentisierung. Syntax: TYPE [user[:password]] Mögliche Werte für Type: MSCHAPV2 (für PEAP), TTLS-MSCHAPV2 (für TTLS). TLS hat keine Phase 2.
use_radius_server	yes	Definiert, ob ein externer Radius Server für die Authentifizierung benutzt wird. Wenn der Parameter aktiv ist darf die Authentifizierung oben nicht aktiv sein.
source_addr	192.168.1.3	Definiert eine IP-Adresse, die als Source für die Kommunikation mit dem Radius Server benutzt wird. Als Default-Wert wird die IP-Adresse gemäss Routing-Tabelle benutzt.
radius_ipaddr	192.168.1.3	IP-Adresse vom Access-Point. Wenn nichts angegeben wird, wird die IP-Adresse vom der WLAN-Karte benutzt.
radius_server	192.168.1.1:1812	IP-Adresse und Port vom Radius-Server. Wenn kein Port angegeben wird, wird 1812 als Default-Wert benutzt. Es können mehrere Server konfiguriert werden. Diese werden benutzt, wenn der erste keine Antwort gibt.
radius_secret		Definiert das Passwort für den Zugriff auf den Radius-Server.

Attribute	Value (Default)	Description
radius_accounting		IP-Adresse und Port des Accounting-Servers. Wenn kein Port definiert wird, wird 1813 verwendet.
radius_acct_secret		Passwort für den Zugriff auf den Accounting-Server.
radius_retry		Intervall (in Sekunden) um zu versuchen, zum ersten Radius-Server zurückzukehren, auch wenn der aktuelle Server immer noch funktioniert. Wenn der Parameter nicht gesetz wird, bleibt das System auf dem verwendeten Radius-Server, bis dieser nicht mehr funktioniert und wechselt erst dann auf einen anderen.
attribute		Hier können weiter Radius Attribute angegeben werden, die dann zum Radius-Server geschickt werden. Syntax: attribute = attr_id[:syntax:value] attr_id: Radius Attribute Typ. syntax: s = String, d = integer, x = octet string value: Attribut-Wert im Format wie von syntax spezifiziert. Beispiele: attribute = 6:d:2 → Service-Type = 2 (Framed) attribute = 61:d:15 → NAS-Port-Type = 15 (Ethernet)

4.3.42 [tpm]

Das Trusted Platform Module (TPM) kann verwendet werden um Zertifikate sicher auf dem AnyRover zu speichern. Ein private Key, der im TPM abgespeichert wird, verlässt den TPM nicht mehr. Bei der Auslieferung ist der TPM mit einem owner Passwort und einme SRK Passwort initialisiert die beide „abc123“ lauten. Das Passwort kann mit tpm_changeownerauth geändert werden. Das Initialisieren des TPM wird im File /etc/tpm/HOWTO beschrieben. Das owner-Passwort ist nötig um die Eingentümerschaft des TMPs zu übernehmen und wird nicht im Konfigfile abgespeichert. Das SRK Passwort wird benötigt um auf gespeicherte Daten und private keys zugreifen zu können. Dieses wird im cablynx.conf abgelegt.

Attribute	Value (Default)	Description
srk	abc123	SRK Passwort
pkcs11	abc123	Private Keys werden in PKCS#11 Kontainern gespeichert, welche ein Addon vom TPM sind. Der PKCS#11 Kontainer hat zwei Passworter: - SO pin: security officer pin. Dies ist das Masterpasswort des Kontainers und wird nicht im Konfigfile abgespeichert. - user pin: dieses Passwort wird benötigt um auf gespeicherte Zertifikate zuzugreifen und wird hier definiert.

4.3.43 [power]

Die CPU kann mit drei verschiedenen Takt-Frequenzen betrieben werden, was sich im Stromverbrauch niederschlägt. Die Takt-Frequenz kann dynamisch zur Laufzeit angepasst werden, und zwar beliebig oft.

Es gibt vordefinierte Governors, welche ein fixe Einstellung vornehmen, oder die Einstellung kann einem Userspace Programm überlassen werden.

Der Governor Performance lässt die CPU stets auf der maximalen Takt-Frequenz laufen, der Governor Powersave stets auf der minimalen Frequenz.

Eine Mischung der beiden ist der Governor Ondemand, welcher die Takt-Frequenz der aktuellen Last des Prozessors anpasst. Solange der Prozessor praktisch idle ist, läuft die CPU auf der minimalen Frequenz, doch sobald die Last steigt, wird auch die Takt-Frequenz angepasst.

Schliesslich erlaubt der Governor Userspace, dass die Frequenz durch eigene Programme angepasst werden kann.

Die aktuelle Takt-Frequenz kann mit dem Befehl

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

abgefragt werden. Eine neue Takt-Frequenz wird mit diesem Befehl eingestellt:

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

Die Liste mit unterstützten Takt-Frequenzen anzeigen geht so:

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

Die Frequenzen sind somit 396MHz, 792MHz und 996MHz.

Eine interne Batterie (BBU: Battery Backup Unit) ist als Option erhältlich. Ein zusätzlicher Service läuft auf dem Gerät, der diese Batterie kontrolliert. Über zusätzliche Optionen kann das Verhalten des Systems bei Ausfall der Speisung beeinflusst werden.

Attribute	Value (Default)	Description
governor	performance	Stellt den aktuellen Governor ein, welcher die CPU-Frequenz betreut. Mögliche Governors sind: performance: CPU läuft immer auf der maximalen Frequenz powersave: CPU läuft immer auf der minimalen Frequenz ondemand: CPU läuft auf dem Minimum und wird bei Last hochgetaktet. userspace: Frequenz wird duch ein Userspace-Programm eingestellt. Ein solches muss noch bereit gestellt werden. Mittels eines zweiten Parameters kann eingestellt werden, dass die Einstellung erst am Schluss des Boot-Prozesses durchgeführt werden soll. Somit bootet das Gerät noch mit maximaler Takt-Frequenz und schaltet dann für den Betrieb zurück. Syntax: governor = <type> [when] when kann die Werte preboot (Default wenn nichts angegeben) oder postboot annehmen.
bbu_start		Wenn dieser Wert auf yes gesetzt wird, dann startet der Service zur Überwachung des BBU.
bbu_governor		Mit diesem Parameter wird definiert, auf welchen Governor das System umschaltet, wenn es im Batterie-Betrieb ist. Wenn der Wert leer gelassen wird, findet keine Umschaltung statt. Default: powersave

Attribute	Value (Default)	Description
bbu_voltage		Damit wird die Spannung in Volt definiert, mit der die Batterie das Gerät speisen soll. Dieser Wert muss mit der Nennspannung des Netzteils übereinstimmen. Das System wählt eine Spannung ca. 10% unter der angegebenen Spannung als Grenzwert, unter dem die Batterie aktiviert wird. Aus der Batterie wird das System dann mit dieser Spannung gespiesen. Hinweis: Wenn dieser Wert und die Spannung des Netzteils nicht übereinstimmen, geht nichts kaputt, aber das System verhält sich möglicherweise nicht wie erwartet. Mögliche Werte: 12...24 Default Wert: 12
bbu_regulation_voltage		Dieser Wert definiert die Spannung in Millivolt, bis zu welcher die Batterie geladen wird. Die Batterie kann bis 4200mV geladen werden, aber ein Wert von mehr als 4000mV verkürzt die Lebensdauer der Batterie signifikant. Default und empfohlener Wert: 4000
bbu_disable_poe1 bbu_disable_poe2		Wenn diese Attribute auf yes gesetzt werden, dann schaltet das System die PoE Module ab, wenn auf Batterie-Mode umgeschaltet wird. Default Wert: yes

4.3.44 [bondix]

Die Software von Bondix dient dazu, verschiedene Uplinks zu bündeln oder ein Failover herzustellen. Damit können beide Modem oder auch noch zusätzliche Uplinks über WLAN oder Ethernet gleichzeitig zu einem Uplink gebündelt werden. Damit kann eine hohe Ausfallsicherheit und unter idealen Bedingungen eine Erhöhung der Bandbreite erzielt werden. Die Bondix-Software baut einen Tunnel zu einem Bondix-Gateway auf, dieser muss für diese Funktionalität ebenfalls erworben werden mit dazugehörigen Lizenzen.

Die Bondix-Verbindung verschlüsselt die Daten nicht; wenn Verschlüsselung benötigt wird, kann dazu IPsec verwendet werden.

Der Bondix Client enthält einen transparenten Proxy, der TCP Verbindungen über den Bondix Tunnel optimieren kann. Wenn der Proxy aktiviert wird, werden automatisch entsprechende Firewall-Regeln eingefügt, so dass der Datenverkehr zum Proxy umgelenkt wird.

Attribute	Value (Default)	Description
start		Definiert, ob diese Sektion aktiv ist.
mode	client	Definiert, dass die Software auf dem AnyRover als Client verwendet wird.
tunnelmode	bonding	Definiert, wie der Datenverkehr über die verschiedenen Links verteilt wird. Möglich sind die Werte «failover», «bonding» und «loadbalancer». Bei failover werden die Daten nur über einen Link gleichzeitig verschickt. Dabei wird sehr dynamisch entschieden, welcher Link gerade der bessere ist. Bei bonding werden die Daten gleichzeitig über alle

Attribute	Value (Default)	Description
		verfügbaren Links verschickt. Dieser Parameter wird übersteuert, wenn auf der Serverseite eine Lite-Lizenz aktiv ist. Dann arbeitet die Software in jedem Fall im Failover-Modus.
logfile	var/log/ saneclient.log	Definiert, wohin das bondix-Logfile geschrieben werden soll. Alle Logeinträge von der Bondix-Software werden auch ins zentrale AnyRover-Logfile geschrieben.
server		IP-Adresse vom Bondix Server. Diese muss übers Internet erreichbar sein.
port	443	Port, über welchen die Verbinung zum Server aufgebaut werden soll. Wenn kein Wert angegeben wird, wird der Port 443 verwendet.
backup_server		Falls zwei Server als Redundanz eingesetzt werden, kann hier die IP-Adresse vom Backup-Server definiert werden. Dieser Server wird erst kontaktiert, wenn der erste Server nicht erreichbar ist.
backup_port	443	Port für die Verbindung zum Backup-Server.
use_encryption		Wenn dieser Wert auf yes gesetzt ist, verwendet Bondix die interne Verschlüsselung der Links mittels ChaCha20 Verschlüsselung.
auth_method	password	Definiert, ob sich der Client per Passwort oder Zertifikat beim Server identifiziert. Mögliche Werte sind: password, cert.
name		Hier wird der Tunnel-Name definiert
password		Passwort, dass für den Verbindungsaufbau zum Server benötigt wird. Dies muss mit dem definierten Passwort auf der Serverseite übereinstimmen.
root		Pfad zum Root-Zertifikat. Dieses muss im PEM-Format abgelegt werden.
cert		Pfad zum Zertifikat. Dieses muss im PEM-Format abgelegt werden.
key		Pfad zum Private Key. Dieser muss im PEM-Format abgelegt werden.
interfaces		Hier werden alle Interfaces aufgelistet, die als Bondix-Channel verwendet werden sollen. Jedem Interface muss zusätzlich ein Channel-Name zugeordnet werden. Die Interfaces müssen den tatsächlich auf dem AnyRover verfügbaren Interfaces entsprechen. Channelnamen können belibig definiert werden, aber keine Leerzeichen oder Tabulatoren enthalten. Syntax: interfaces = interface1:channelname1, interface2:channelname2, … Beispiel: interfaces = ppp0:Swisscom, ppp1:Sunrise, wlan0: WLAN
local_iface	bondix0	Definiert den Namen des lokalen Bondix-Tunnel-Interfaces.
bonding_proxy	no	Aktiviert den Bonding Proxy
proxy_iface		Der Bonding Proxy empfängt auf diesem Interface TCP Pakete, welche dann an den Server übermittelt werden. Dieser Parameter muss gesetzt werden.
proxy_host	0.0.0.0	Mit diesem Parameter kann zusätzlich noch eine lokale IP-Adresse definiert werden, auf der der Proxy den Verkehr erwartet. Die Angabe ist optional.
proxy_port	18080	Definiert den Port, auf dem der Proxy die Pakete empfängt. Der Port muss in der Firewall Sektion noch geöffnet werden.
webgui_start		Aktiviert das Bondix-Webgui. Dafür muss hier ein Interface und

Attribute	Value (Default)	Description
		ein Port definiert werden Syntax: interface:port Beispiel eth0:80 Zusätzlich zu dieser Konfiguration muss der lokale Webserver aktiviert und der entsprechende Port in der Firewall geöffnet werden.
webgui_config		Mögliche Werte: yes, no
webgui_monitor		Mögliche Werte: yes, no
webgui_key		Passwort um ins WebGUI einloggen zu können

4.3.45 [voltage]

Hier können Aktionen definiert werden, welche bei Unterschreiten einer bestimmten Eingangsspannung ausgeführt werden. Das kann nützlich sein, wenn der AnyRover an einer Batterie betrieben wird.

Der AnyRover überwacht seine Eingangsspannung und kann Aktionen ausführen, wenn diese unter einen bestimmten Wert sinkt. Diese Aktion kann sein jemanden zu alarmieren, den AnyRover herunterzufahren, oder sonst ein Skript auszuführen. Es ist auch möglich, eine weitere Aktion auszuführen, wenn die Spannung wieder über einen anderen Wert steigt (ausser der AnyRover wurde heruntergefahren…).

Die Eingangsspannung wird mit einem 10 Bit Analog-Digital-Konverter (ADC) gemessen, und umfasst fast den ganzen Eingangsspannungsbereich von 0..60V. Die Schrittgrösse ist ca. 72mV und wird noch auf die nächsten 10mV gerundet.

Hinweis: Der AnyRover ist ein mobiler Router, und kein Präzisions-Voltmeter. Die Messwerte sollten mit entsprechender Vorsicht benutzt werden.

Hinweis: Der AnyRover stellt ab, wenn die Spannung unter ca. 8.3V sinkt.

Attribute	Value (Default)	Description
start		Definiert, ob das Monitoring-Skript gestartet werden soll
limit	11200	Spannung in mV, unter der die Aktion getriggert werden soll.
delta	300	Wenn nach dem Ausführen der Aktion die Spannung wieder über den Wert (limit+delta) steigt, wird der Alarm zurückgesetzt. Der Wert für delta ist auch in mV und muss >=150mV sein.
count	5	Anzahl von aufeinanderfolgenden Messwerten, die unter der Limite sein müssen, damit die Aktion ausgeführt wird. Der Wert 0 wird nicht empfohlen, um Fehlalarme aufgrund einzelner Fehlmessungen zu vermeiden.
interval	30	Intervall in Sekunden zwischen zwei aufeinanderfolgenden Messungen.
ignition		Mit diesem Parameter kann definiert werden, ob vor dem Ausführen der Aktion noch das Ignition Signal geprüft werden soll. Mögliche Werte: - ignore: Keine Prüfung des Ignition Signals

Attribute	Value (Default)	Description
		- on: Aktion wird nur ausgeführt, wenn Ignition auf on ist - off: Aktion wird nur ausgeführt, wenn Ignition auf off ist Wenn das Ignition Signal auf dem falschen Wert ist, wird nur eine Log-Meldung ausgegeben, aber keine Aktion ausgeführt. Wenn die Spannung tief bleibt, und das Ignition Signal wechselt, wird die Aktion schliesslich doch ausgeführt. Dieser Parameter sollte benutzt werden, wenn der AnyRover bei Batteriebetrieb ausschalten soll, weil er bei eingeschaltetem Ignition Signal lediglich einen Reboot ausführt, aber nicht ausschaltet.
action1 action2		Wenn alle Bedingungen erfüllt sind, führt der AnyRover die action1 aus, gefolgt 2s später von der action2. Beide Einträge können auch leer gelassen werden. Als Aktion kann angegeben werden: - LOG: schreibt eine Meldung ins Logfile - POWEROFF: fährt den AnyRover herunter - <script>: Ein beliebiges Skript unter /etc/scripts.d/
recover1 recover2		Die Aktionen, welche ausgeführt werden, wenn nach einen Niederspannungsalarm die Spannung wieder über den zweiten Threshold ansteigt. Das funktioniert nur, wenn der AnyRover nicht heruntergefahren wurde. Verwendung wie actionX oben.
adc_offset		Wenn über den ADC Spannungen gemessen werden (inkl. Einganggspannung), dann wird dieser Wert (Angabe in Millivolt) als Offset addiert. Wenn nur der ADC Wert in Schritten angefragt wird, findet dieser Parameter keine Anwendung. Default Wert: 0
adc_scale		Wenn über den ADC Spannungen gemessen werden (inkl. Einganggspannung), dann wird der Wert mit diesem Faktor muliplizierт (bevor der Offset addiert wird). Wenn nur der ADC Wert in Schritten angefragt wird, findet dieser Parameter keine Anwendung. ADC[mV] = adc_scale * Messwert + adc_scale Default Wert: 1

5 Unterhalt

Nach dem Einschalten startet der AnyRover alle Dienste automatisch und ist nach kurzer Zeit für den Einsatz bereit. Über verschiedene Mechanismen kann zwecks Wartung und Entwicklung neuer Funktionen ins System eingegriffen werden.

5.1 Lock-Files

Mittels Lock-Files kann der Zugriff des Systems auf verschiedene Ressourcen unterbunden werden. Die Lock-Files stehen im Verzeichnis /var/lock, und für die Funktion entscheidend ist einzig, ob das File existiert oder nicht, der Inhalt ist irrelevant. Lock-Files können z.B. mit dem Unix-Befehl touch erzeugt werden.

Lock-File	Geschützte Ressource
/var/lock/sms	Der SMS-Dienst greift nicht mehr auf das Modem zu, d.h. es werden keine SMS mehr gelesen und auch keine mehr verschickt.
/var/lock/ppp	Der PPP Dienst startet keine neue Verbindung mehr. Bereits bestehende Verbindungen werden nicht automatisch beendet.
/var/lock/ppp0	Der PPP-Daemon erstellt dieses File, wenn eine PPP-Verbindung steht.

5.2 Hilfsprogramme

Der AnyRover enthält einige Hilfsprogramme, welche bei der Suche von Fehlern nützlich sein können. Auch können sie in benutzerdefinierten Skripts verwendet werden um automatisiert genutzt werden zu können.

5.2.1 Modem Status

Das Programm *at* (nicht der Unix-Dienst gleichen Namens) dient dazu, AT-Befehle an das Modem zu schicken und die Ausgabe anzuschauen. Dem Programm müssen einer oder mehrere AT-Befehle übergeben werden. Die AT-Befehle werden dann in der angegebenen Reihenfolge ans Modem geschickt, und die Ausgaben des Modems auf der Standardausgabe ausgegeben. Der Befehl at ohne Argumente zeigt die Verwendung und alle Optionen an.

5.2.2 SMS senden

Der AnyRover kann – eine entsprechende SIM-Karte vorausgesetzt – SMS versenden. Dazu dient das Programm *sms*. Eingehende SMS können nicht direkt betrachtet werden, doch sind sie im Log-File /var/log/messages zu finden, wenn der zentrale Dienst sie nicht interpretieren konnte.

Verwendung: *sms Telefonnummer Nachricht* (Beispiel: sms 0791112233 „dies ist eine Testnachricht“)

Mit sms ohne Argumente, werden alle Optionen angezeigt.

5.2.3 Zentraler Dienst

Viele Aufgaben werden vom zentralen Dienst gpio_daemon wahrgenommen. Er füttert den Watchdog, überwacht die GPI Pins, verarbeitet die GPS-Daten , steuert das Modem und verarbeitet eingehende SMS.

Über das Programm *cablynxctrl* kann der zentrale Dienst zur Laufzeit gesteuert werden. Mit help werden alle verfügbaren Befehle aufgelistet.

5.2.4 GPIO

Mit Hilfe des Programms *gpio* können die GPI Pins gelesen und die GPO Pins gesetzt werden. Die Pins, welche vom zentralen Dienst überwacht werden, sind davon momentan noch ausgeschlossen.

Verwendung: *gpio ioNummer [val]*

Wenn val 0 oder 1 ist, wird der gpio auf Output gesetzt und geschrieben. Wenn val ? Ist, wird der gpio-Output gelesen. Ohne val wird der gpio auf Input gesetzt und gelesen. Mit gpio ohne Argumente werden alle Optionen angezeigt.

Die Inputs und Outputs auf dem Multifunktions-Stecker werden vom zentralen Dienst überwacht und können nicht mittels gpio gesetzt und gelesen werden. In der Konfigurationsdatei können aber Aktionen definiert werden, welche bei Änderungen der Inputs ausgeführt werden. Der Output kann entweder über solche Aktionen gesteuert werden, oder mittels des Befehls *cablynxctrl* manuell gesetzt werden.

5.2.5 AD-Wandler

Der AnyRover besitzt mehrere Analog-Digital Wandler. Mit Hilfe des Programms *adc* können die Werte der Wandler ausgelesen werden.

5.2.6 Beschleunigungssensoren

Im AnyRover v3 ist kein separater Beschleunigungssensor mehr enthalten, da das GPS-Modul diesen bereits integriert hat.

5.2.7 Datcom

Mit diesem Tool kann man sich auf einem Datcom-Gateway einloggen und die aktuelle GPS-Position übermitteln. Mit *datcom -h* wird die Beschreibung angezeigt.

5.2.8 PIC-Tool

Ab dem AnyRover V2 ist zusätzlich zum Prozessor noch ein PIC verbaut, der verschiedene Werte unabhängig von der Konfiguration speichert. Diese können mit *pic -option val* ausgelesen und geschrieben werden. Mit *pic -h* werden alle Optionen angezeigt.

5.2.9 Zertifikate und TPM

Zertifikate können in einem PKCS#11 Store abgelegt werden, wodurch der Private Key durch den TPM auf dem AnyRover geschützt ist.

Im Zusammenhang mit dem TPM werden insgesamt vier verschiedene Passwörter definiert, welche alle standardmässig auf abc123 gesetzt sind. Für den TPM selber sind das der Owner Key, der benötigt wird, um den TPM zu konfigurieren. Dies muss nur einmal geschehen, und wird bereits bei der Produktion gemacht. Und dann das SRK-Passwort, welches benötigt wird, um auf die Daten im TPM zuzugreifen. Deshalb muss dieses Passwort im Config-File eingetragen werden.

Der PCKS#11 Store selber hat auch zwei Passwörter. Der Security Officer Pin wird zur Initialisierung benötigt und steht deshalb nicht im Config-File. Der User Pin wird für das Schlüssel-Management benötigt und ist darum im Config File anzugeben.

Der Status des PKCS#11 Stores kann mit

```
show tpm
```

angezeigt werden. Dabei sollte die token flags Zeile wie folgt aussehen:

```
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
```

Der Inhalt des PKCS#11 Stores kann wie folgt angezeigt werden (dieser Befehl benötigt einige Sekunden, bis er eine Ausgabe liefert):

```
# 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)
```

Warnungen mit CKR_ATTRIBUTE_TYPE_INVALID können getrost ignoriert werden.

Wenn PKCS#11 Zertifikate mit IPsec oder WLAN/802.1X verwendet werden sollen, muss die jeweilige Key-ID auf den Fingerprint des Keys gesetzt sein (was bei Bezug des Zertifikates über SCEP automatisch so gesetzt wird).

Es besteht auch die Möglichkeit, Datei basierte Keys in den PKCS#11 Store zu importieren

(nur das Exportieren aus dem PKCS#11 Store ist nicht möglich). Dazu muss der Key im DER Format vorliegen. Der Import eines PEM Keys funktioniert so:

```
# 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
```

Nach dem Import wird die Datei mit dem Private Key sinnvollerweise wieder gelöscht.

5.3 Log-Files

Das System schreibt wichtige Meldungen ins Systemweite Log-File. Die Log-Files werden regelmässig gelöscht, um zu verhindern, dass das Flash (Festplatte) voll läuft.

Das Log-File ist /var/log/messages. Neue Meldungen werden am Schluss der Datei angehängt. Mit dem Befehl

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

können die letzten Zeilen des Log-Files angeschaut werden. Der Schalter „-f“ sorgt dafür, dass sich der tail-Befehlt nicht beendet und neue Meldungen laufend nachführt. Mit „-F“ bleibt der Tail-Befehl sogar aktiv, nachdem das Log-File rotiert wurde.

6 Beispielkonfigurationen

In dieser Sektion werden Beispielkonfigurationen für verschiedene Situationen abgedruckt und erklärt. Dabei werden jeweils nur die Zeilen der Konfigurationsdatei abgedruckt, welche für die gewünschte Funktion relevant sind. Für alle anderen Zeilen wird von der Standardkonfiguration ausgegangen.

Die Kommentare in der Konfigurationsdatei sind hier nicht wiedergegeben.

Für die Beispiele werden die IP-Adressen 172.16.X.Y – 172.24.X.Y als öffentliche Adressen , und die Adresse 192.168.X.Y als private Adressen angenommen.

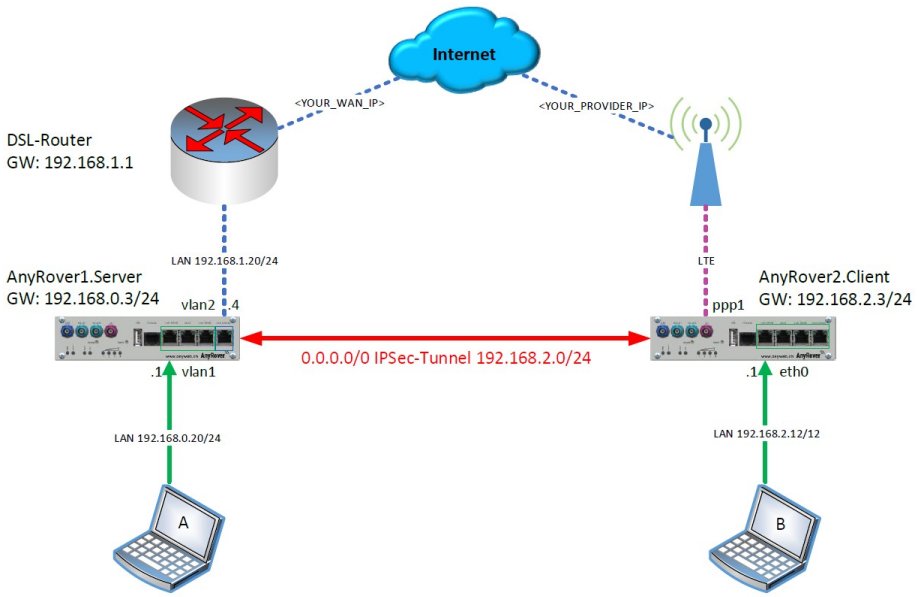
Werte in Grossbuchstaben und spitzigen Klammern (z.B. <YOUR_KEY>) müssen noch durch die korrekten Werte ersetzt werden.

6.1 Permanenter IPsec Server mit mehreren Clients und Pre-Shared Key

Der AnyRover soll über eine 3G/4G und einen Ipsec Tunnel eine permanente Anbindung ins interne Netzwerk herstellen. Die Authentisierung beim Server erfolgt über einen Pre-Shared Key, die Identifikation über den Hostnamen.

Verwendet wird eine SIM-Karte, die vom Provider eine IP-Adresse hinter dem NAT-Gateway erhält. Der IPsec Server hat die öffentliche IP-Adresse <YOUR_WAN_IP> und hängt hinter einem DSL-Router. Das interne Netzwerk ist 0.0.0.0/0.

Alle anderen Internet-Zugriffe sollen direkt ausgeführt werden.



Client-Config:

```
[system]
ipaddr = 192.168.2.3/24
hostname = AnyRover2.Client
[dhcp]
dhcp_start = 192.168.2.11
dhcp_ent = 192.168.2.12
router = 192.168.2.3
dns = 192.168.2.3
[firewall]
accept = ppp1,udp,500
accept = ppp1,esp,
accept = ppp1,udp,4500
accept_fw = eth0,,,
[chat_script]
apn = <YOUR_PROVIDER_APN>
[ipsec]
start = yes
remote = <YOUR_WAN_IP>
local_net = eth0
remote_net = 0.0.0.0/0
my_identifier = fqdn, client1.example.org
peers_identifier = fqdn, server.example.org
psk = <YOUR_PRE_SHARED_KEY>
```

Server-Config:

```
[system]
ipaddr =
hostname = AnyRover1.Server
[switch]
start_vlan = yes
vlan1 = 1,2,3
ipaddr1 = 192.168.0.3/24
vlan2 = 4
ipaddr2 = 192.168.1.20/24
[firewall]
accept = vlan2,udp,500
accept = vlan2,esp,
accept = vlan2,udp,4500
accept_fw = vlan1,,vlan2
accept_fw = ,,vlan2
[ipsec]
start = yes
setup = add
remote = any
local = vlan2
local_net = 0.0.0.0/0
remote_net = any
remote_range = 192.168.0.0/16
my_identifier = fqdn, server.example.org
peers_identifier = fqdn, client1.example.org
psk = <YOUR_PRE_SHARED_KEY>
```

Funktionskontrolle:

«show ipsec»

Client:

```
~ # show ipsec
  10.243.68.184...178.193.119.80  IKEv2, dpddelay=5s
   local:  [client1.example.org] uses pre-shared key authentication
   remote: [server.example.org] uses any authentication
   child:  192.168.2.0/24 === 0.0.0.0/0 TUNNEL, dpdaction=restart
 ESTABLISHED 19 hours ago, <YOUR_PROVIDER_IP>[client1.example.org]...    <YOUR_WAN_IP> [server.example.org]
 IKEv2 SPIs: 728907ac96387e8b_i* 1c36af6cba6484f3_r, pre-shared key reauthentication in 3 hours
 IKE proposal: AES_CBC_256 / HMAC_SHA2_256_128 / PRF_HMAC_SHA2_256 / MODP_1536
  INSTALLED, TUNNEL, reqid 5, ESP in UDP SPIs: c02cf847_i c3138d40_o
  AES_CBC_256/HMAC_SHA2_256_128, 0 bytes_i, 0 bytes_o, rekeying in 48 minutes
   192.168.2.0/24 === 0.0.0.0/0
```

Server:

```
~ # show ipsec
  192.168.1.20...%any  IKEv2, dpddelay=5s
   local:  [server.example.org] uses pre-shared key authentication
   remote: [client1.example.org] uses any authentication
   child:  0.0.0.0/0 === 192.168.0.0/16 TUNNEL, dpdaction=restart
 ESTABLISHED 19 hours ago, 192.168.1.20[server.example.org]...              [client1.example.org]
 IKEv2 SPIs: 728907ac96387e8b_i 1c36af6cba6484f3_r*, pre-shared key reauthentication in 4 hours
 IKE proposal: AES_CBC_256 / HMAC_SHA2_256_128 / PRF_HMAC_SHA2_256 / MODP_1536
  INSTALLED, TUNNEL, reqid 1, ESP in UDP SPIs: c3138d40_i c02cf847_o
  AES_CBC_256/HMAC_SHA2_256_128, 0 bytes_i, 0 bytes_o, rekeying in 46 minutes
   0.0.0.0/0 === 192.168.2.0/24
```

6.2 Permanenter IPsec Server mit mehreren Clients und Zertifikaten

Basierend auf dem Beispiel 6.1 werden folgende Befehle angepasst.

Client-Config:

```
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]
…
```

Server-Config:

```
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=ClientX,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-----
[certificate]
name = ipsec-crl
type = pem
-----BEGIN X509 CRL-----
MII...
-----END X509 CRL-----
```

6.3 IPsec Tunnel bei Bedarf

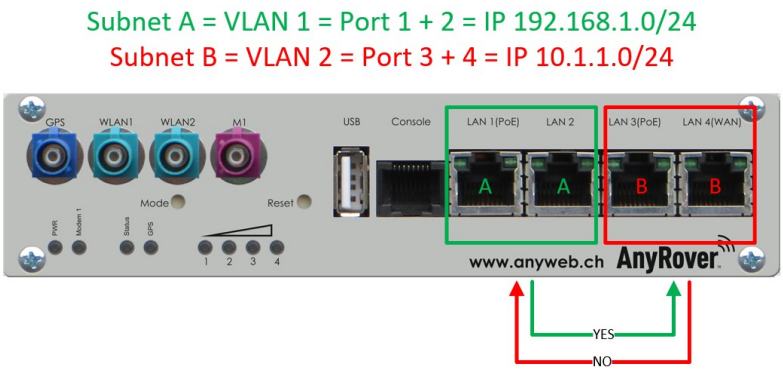
Die 3G/4G Verbindung und der IPsec Tunnel werden nur bei Bedarf aufgebaut. Wenn 5 Minuten kein Verkehr durch den Tunnel fliesst, wird die Verbindung wieder beendet. Dieses Beispiel beruht auf der Konfiguration „6.1 Permanenter IPsec Tunnel mit mehreren Clients und Pre-Shared Key“.

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

6.4 2 lokale Subnetze mit NAT

Der AnyRover soll 2 lokale Subnetze A (Ports 1+2, 192.168.1.0/24) und B (Ports 3+4, 10.1.1.0/24) haben. Zugriff von A nach B ist möglich, der Verkehr wird mittels NAT angepasst. Zugriff von B nach A ist gesperrt.

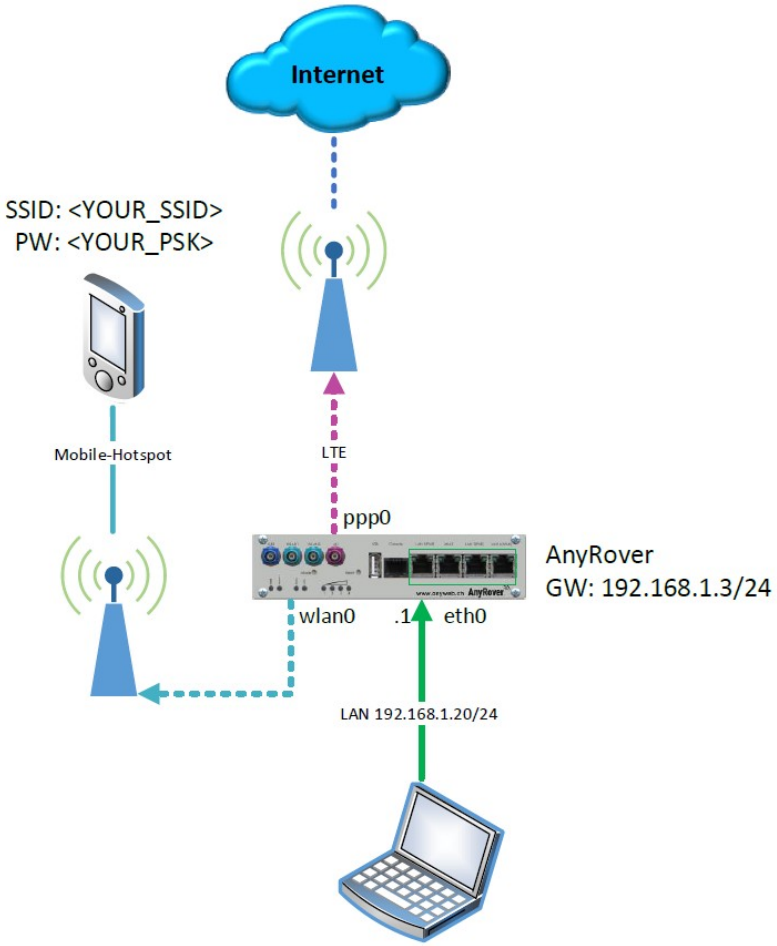
```
[system]
ipaddr =
[switch]
start_vlan = yes
vlan1 = 1,2
ipaddr1 = 192.168.1.1/24
vlan2 = 3,4
ipaddr2 = 10.1.1.0/24
[firewall]
accept_fw = vlan1,,vlan2
nat = vlan2
```



6.5 Wireless Client

Der AnyRover soll sich über Wireless mit einem Mobile-Hotspot verbinden und die

Verbindung seinen Clients zur Verfügung stellen. Über den SIM Slot 1 wird der AnyRover mit dem Internet verbunden. Der Access Point authentifiziert sich mit WPA2-PSK.

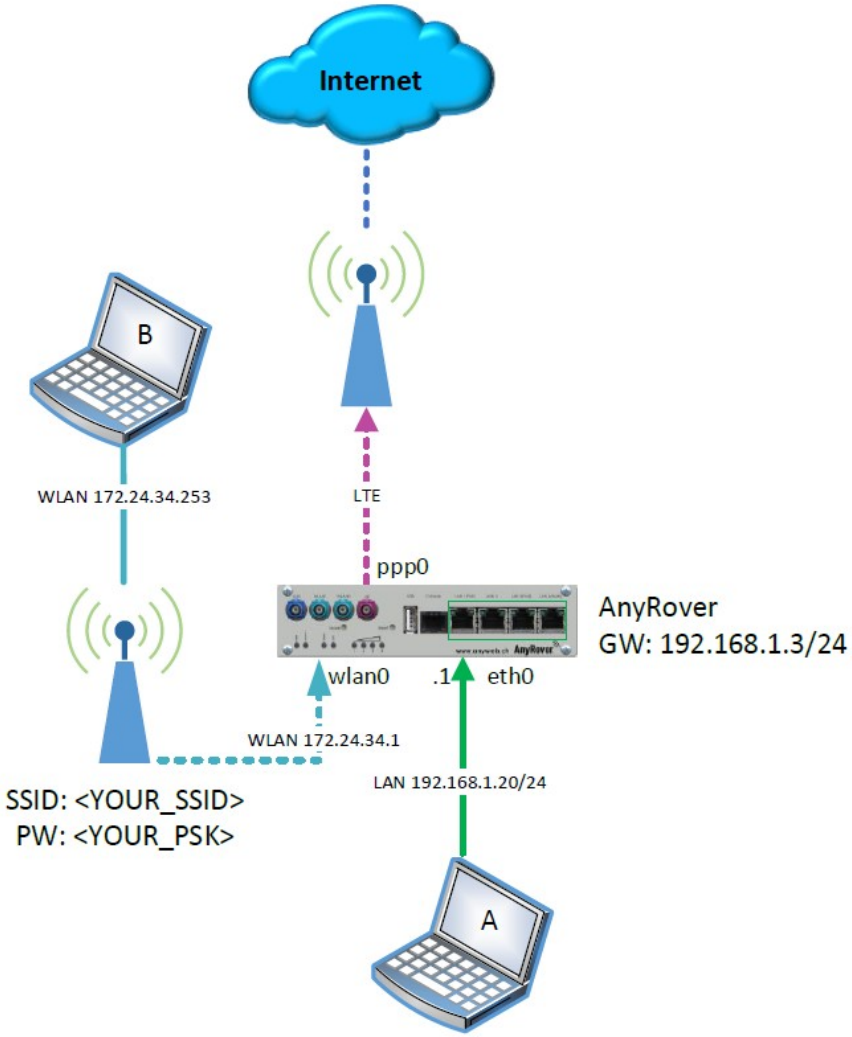



```
[firewall]
accept_fw = eth0,,
nat = ppp0, wlan0
[wlan]
start = yes
country = <YOUR_2_LETTER_COUNTRY_CODE>
ssid = <YOUR_SSID>
key_management = WPA2-PSK
scan_ssid = yes
pre_shared_key = <YOUR_PSK>
```

6.6 Wireless Access Point mit DHCP Server

Der AnyRover soll als Wireless Access Point betrieben werden. Die Authentifikation wird dank WPA2-PSK sichergestellt. Den Clients wird eine IP-Adresse per DHCP zugeteilt die über den SIM Slot 1 ins Internet kommuniziert. Als zweite Auswahl kann man sich über den

lokalen Ethernetport verbinden.




```
[dhcp]
name = wlan0
start = yes
[firewall]
accept = ,udp,:67:68
accept_fw = etho0,,
accept_fw = wlan0,,
[wlan]
start = yes
mode = ap
country = <YOUR_2_LETTER_COUNTRY_CODE>
channel = 1
ipaddr = 172.24.34/24
ssid = <YOUR_SSID>
key_management = WPA2-PSK
scan_ssid = yes
wpa_psk = anyweb350
```

6.7 Mehrere Client-Verbindungen über IPsec mit PSK

Mehrere identische Clients verbinden sich über IPsec mit dem Rechner in der Zentrale. Jeder Client hat sein eigenes kleines lokales Netz (192.168.2.X/30), das über den Tunnel angesprochen werden kann. Die Authentisierung erfolgt über Pre-shared Keys, die Identifikation über Hostnamen.

Die Clients verbinden über den 3G Link und befinden sich hinter dem NAT-Gateway des Providers. Der Verbindungsaufbau muss daher durch den Client erfolgen.

Der Server hat die öffentliche Adresse 172.16.1.1 (vlan1) und das interne Netz 192.168.1.0/24 (vlan2).

Client Konfiguration:

```
[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 = :/
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 Konfiguration:

```
[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 = :/
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 Konfiguration für einen 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
```

6.8 Dateien über Email versenden

Der AnyRover soll regelmässig Log-Files über Email versenden. Das hier gezeigte Skript wird per Cronjob gestartet, jeweils donnerstags um 1:04 Uhr.

```
[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.
```

6.9 IPsec Server für Cisco VPN Clients

Der AnyRover soll als Server für einen Cisco VPN Client aufgesetzt werden. Die Authentisierung muss über Zertifikate erfolgen, die Verwendung von Pre-Shared Keys ist nicht möglich (Grund: dann verwendet der Cisco VPN Client den agressiven Modus, welcher vom AnyRover aus Sicherheitsgründen nicht unterstützt wird).

Momentan ist nur eine Verbindung aufs Mal möglich. Die Erweiterung auf mehrere gleichzeitige Verbindungen folgt in einer späteren Version.

Diese Methode funktioniert auch für Verbindungen von einem iPhone aus. In diesem Fall muss auf dem iPhone als Server ein Hostname angegeben werden, und das Zertifikat auf dem AnyRover muss diesen Hostnamen im CN-Feld drin haben.

Um die Zertifikate ins iPhone zu importieren muss das CA-Zertifikat als .pem Datei und das Client-Zertifikat (mit Key) als .p12-Datei im iPhone eingefügt werden.

Eine .p12-Datei wird mit folgendem Befehl aus den zugehörigen .pem-Dateien erzeugt:

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

Der Befehl fragt nach einem Export-Passwort. Dieses muss angegeben werden, weil das iPhone nicht mit .p12-Dateien ohne Passwort umgehen kann.

```
[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_hash_alg = sha1
ph2_hash_alg = sha1
dh_group  = 5
pfs_group =
```

Dann müssen noch die Zertifikate in den jeweiligen [certificate] Sektionen angegeben werden.

6.10 GPO setzen

Der General Purpose Output (GPO) soll eingeschaltet werden, wenn eine Datei angelegt und ausgeschaltet, wenn sie gelöscht wird.

Als Beispiel dient die Datei /var/lock/ppp0, welche nur existiert, wenn die 3G-Verbindung steht.

```
[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
```

6.11 SCEP Zertifikats-Management

Der AnyRover kann Zertifikate über SCEP (Simple Certifikcate Enrolment Protocol) beziehen. Die Zertifikate werden dabei im Verzeichnis /etc/certs/ abgelegt, von wo sie in der Konfiguration in [certificate] Sektionen referenziert werden können.

Der Private Key eines Zertifikates kann entweder in einer Datei abgelegt werden, oder aber durch den TPM (Trusted Platform Module) geschützt in einem PKCS#11 Store. Effektiv ist der Key dann auch in einer Datei im Filesystem abgelegt (unter /etc/opencryptoki/), aber durch den TPM so verschlüsselt, dass er nur im TPM drin entschlüsselt werden kann, und vom TPM nicht im Klartext herausgegeben wird. Dadurch kann dieser Schlüssel nur auf genau diesem AnyRover (bzw. genau diesem TPM) benutzt werden.

6.11.1 Datei basierte Zertifikate

Bezug von Zertifikate für die Speicherung in Dateien und Verwendung in einer IPsec Verbindung.

```
[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
```

6.11.2 TPM basierte Zertifikate

Gegenüber den Datei basierten Zertifikaten im letzten Abschnitt ändert sich folgendes. Für den Private Key wird trotzdem eine Datei angelegt, aber statt des Keys enthält diese Datei nur die ID des Keys, d.h. einen Hex-String in der Form

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
```

6.12 IEEE 802.1X Port Security

Der AnyRover soll seine Ethernet-Ports via 802.1X schützen, so dass sich ein Client zuerst anmelden muss, bevor er den Port verwenden kann.

6.12.1 Authenticator

Bei der Verwendung von 802.1X auf den LAN Ports sollte jeder Port in einem eigenen

VLAN separiert werden. Weil jedes Mal, wenn ein Client eingesteckt wird, die diesem Interface zugeordnete Liste von authentisierten MAC Adresse gelöscht wird, funktionierts nicht korrekt, wenn ein Authenticator auf eth0 betrieben wird.

Sollen mehrere Ports per 802.1X gesichert werden und im gleichen Subnetz liegen, müssen die Ports in individuellen VLANs liegen und in einer Bridge zusammengefasst werden.

Im Beispiel wird Port 1 für EAP-TLS konfiguriert, also Authentisierung per Zertifikat. Im Multi-Host Modus reicht eine gültige Authentisierung, damit alle Clients kommunizieren können. Die Idee ist, dass an Port 1 ein externer Switch angeschlossen wird, der seine Ports selber auch per 802.1X schützt. Sobald sich der Switch authentisiert hat, dürfen auch alle Clients kommunizieren, nachdem sie sich beim Switch autentisiert haben.

Port 2 ist vorgesehen für Clients, die sich per EAP-PEAP authentisieren. Auf Port 3 ist ein MAC Authentication Bypass eingerichtet, z.B. für einen Drucker. Für 802.1X fähige Clients wird ein externer Radius-Server befragt (über VLAN4).

Der Authenticator kann noch nicht mit PKCS11 Keys umgehen.

```
[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
```

6.12.2 Supplicant

Der Supplikant auf vlan3 authentisiert sich mittels Zertifikat, auf vlan4 mittels Username und Passwort. Als Supplikant kann sich der AnyRover auch mit PKCS11 Zertifikaten authentisieren; das wird grundsätzlich gleich konfiguriert wie für IPsec.

```
[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
```

Mit PKCS11 basierten Keys sieht die Konfiguration so aus:

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

A Kontakt

A.1 Projektverantwortliche

A.1.1 Kommerziell

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

A.1.2 Technische Projektleitung

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

A.1.3 Support und Wartung

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 Konfigurations-Datei

```
################################################################################
### 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]>physIF, 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 = 0x112233445566778899900112233
#wep_key = 0x123456789012345678901234567890123456
#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

Für weitere Informationen zu den GNU-Lizenzen GPL und LGPL siehe 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.
```