Inhaltsverzeichnis

SvxLink

Der SvxLink-Server ist ein universelles, von SM0SVX entwickeltes Sprachrepeater-System.

Der Quellcode ist auf GitHub unter https://github.com/sm0svx/svxlink verfügbar. Der Build-Prozess wird unter SvxReflector beschrieben.

Für die Sprachausgaben sind zusätzlich Sprachdateien notwendig, diese sind unter https://github.com/sm0svx/svxlink-sounds-en_US-heather/releases verfügbar. Unter SvxLink-PTT wird die Ansteuerung der Sende-Empfangsumschaltung beschrieben.

Weitere Infos:

- SvxLink-Wiki: https://github.com/sm0svx/svxlink/wiki
- SvxReflector: Vernetzung von SvxLink
- SvxPortal: Dashboards für SvxLink und SvxReflector
- Audio-Kalibrierung bei SvxLink
- Rundspruchausgabe über SvxLink
- Schwedische Sammlung zu SvxLink: http://www.granudden.info/?page=/Ham/Repeatrar/SM5GXQ_en/
- Diskussionsgruppe zu SvxLink: https://groups.io/g/svxlink
- Installationsanleitung im DARC-Wiki - https://wiki.n18.de/doku.php?id=svxlink:start
- Ausgabe über UDP - SvxLink-UDP
- Tetra-DMO-Vernetzung mit Svlink
- SvxLink in Südtirol:
  - Südtirol analog - Übersicht: https://drc.bz/betriebsarten/linksuedtirol/
  - Orange-Pi: https://drc.bz/technik/analog-digitaltechnik/svxlink-mit-orange-pi-zero/
  - Integration von Discord in Svxlink (über SvxReflector): https://pkg.go.dev/gitlab.com/galberti/svxcord#section-readme
- SvxLink in SE https://svxportal.sm2ampr.net/
- SvxLink inRO https://rolink.network/
- SvxLink in FR http://rrf4.f5nlg.ovh:82/
- SvxLink in PL https://fm-poland.pl/dashboard/
- SvxLink in UK https://portal.svxlink.uk:8443/
- SvxLink in DE
- Roger-Bee anpassen
- Ein Fork von SvxLink von DL1HRC mit zusätzlichen Features wie TetraLogic und usrpLogic - https://github.com/dl1hrc/svxlink
  - Präsentation bei der HamRadio2023
  - Diskussion zu usrpLogic im SvxLink-Forum
- Sprechgruppen (Talk Groups) im SvxLink

test

SvxLink Roger Beep

Svxlink kann optional am Ende der Aussendung einen Roger-Beep senden.

Dieser wird in der Konfigurationsdatei "svxlink.conf" im Abschnitt "[RepeaterLogic]" durch die Einstellung RGR_SOUND_DELAY gesteuert.

Der Wert "-1" deaktiviert den Ton, positive Werte definieren die Verzögerung in ms bis der Ton ausgesendet wird.

In der Standard-Konfiguration wird die konkrete Aussendung durch das TCL-Script "Logic.tcl" im Verzeichnis "/usr/share/svxlink/events.d" definiert, konkret durch die Prozedur "send_rgr_sound {}":

```
#
# Executed when the squelch have just closed and the RGR_SOUND_DELAY timer has
# expired.
#
proc send_rgr_sound {} {
  variable sql_rx_id

  if {$sql_rx_id != "?"} {
    # 200 CPM, 1000 Hz, -10 dBFS
    CW::play $sql_rx_id 200 1000 -10
    set sql_rx_id "?"
  } else {
    playTone 440 500 100
  }
  playSilence 100
}
```

Diese kann durch ein eigenes Script ersetzt werden. Dazu wird das Untervereichnis "local" angelegt und dort ein TCL-Script erstellt, etwa "rogerbeep.tcl" im Verzeichnis "/usr/share/svxlink /events.d/local/":

```
namespace eval Logic {

variable is_rf 0;

proc squelch_open {rx_id is_open} {
        variable sql_rx_id;
        variable is_rf;
        set sql_rx_id $rx_id;
        if {!$is_open} {
                set is_rf 1;
        }
}

proc send_rgr_sound {} {
        variable is_rf;
        if {!$is_rf} {
                # Signal wurde vom Netzwerk empfangen
                playTone 500 300 150;
        } else {
```

```
                # Signal wurde von lokal empfangen
                playTone 1633 300 50;
                playSilence 80;
                playTone 1209 300 50;
        }
        set is_rf 0;
        }
}
# end of namespace
```

In obigen Script wird unterschieden, ob über Funk (rf) oder anders (Svxreflector) empfangen wurde, je nachdem wir ein tiefer oder ein hoher Ton ausgesendet.

Alternativ zum integrierten Tongenerator kann auch eine Sounddatei abgespielt werden. Diese ist (sprachabhängig) zu hinterlegen, etwa in "/usr/share/svxlink/sounds/en_US/Core".

Svxlink erwartet die Sprachdatei in einem bestimmten Format (RIFF-Magic, 16k, mono), dieses kann mit "sox" erzeugt werden:

```
sox beep500-10.wav -r 16k -c1 out/beep500-10.wav
```

Nachdem nun Töne in Sprachdateien vorhanden sind, können diese nun in das Skript integriert werden, etwa folgendermaßen:

```
proc send_rgr_sound {} {
        variable is_rf;
        if {!$is_rf} {
                # Signal wurde vom Netzwerk empfangen
                playMsg "Core" "beep500-30";
                playSilence 80;
        } else {
                # Signal wurde von Lokal empfangen
                playMsg "Core" "beep1633-30";
                playSilence 80;
                playMsg "Core" "beep1209-30";
                playSilence 80;
        }
        set is_rf 0;
        }
}
```

Beispielsdateien im korrekten Format finden sich im Anhang (siehe unten).

SvxLink TG

Repeater mit Svxlink können über SvxReflector vernetzt werden.

Bei der Vernetzung ist es möglich Gruppen zu bilden, diese werden in Anlehnung an DMR als TG (Talk Groups) bezeichnet. Alle Mitglieder einer Gruppe hören sich gegenseitig und können miteinander kommunizieren.

Die Auswahl der aktiven TG kann entweder

- statisch erfolgen (der Repeater ist immer Mitglied einer TG, also etwa der "starren" Verbindung von zwei Repeaterstandorten),
- über CTCSS (der Repeater akzeptiert an der Eingabe mehrere CTCSS-Frequenzen, je nach Frequenz wird eine unterschiedliche TG ausgewählt oder
- über DTMF-Auswahl erfolgen.

Die Tatsache, dass SvxLink Talkgroups (TG) unterstützt bedeutet aber keineswegs im Umkehrschluss, dass es generell sinnvoll oder wünschenswert wäre, wenn eine Auswahl an einer Vielzahl an TGs angeboten wird oder Repeater in eine Vielzahl an TGs unterteilt werden. Vielmehr sind TGs ein Werkzeug, dass dazu dient, sinnvolle und für Nutzer verständliche, bedienbare Lösungen zu bauen.

Die mehrfache Mitgliedschaft in Gruppen hat neben der Verständlichkeit und Bedienbarkeit - wie auch bei digitalen Systemen - das Problem, dass zu einem konkreten Zeitpunkt ein einzelner Repeater nur die Kommunikation einer Gruppe wiedergeben kann. Sind mehre TG aktiv, so muss der Repeater entscheiden, welche TG er wiedergibt, typischerweise nach Prioritäten oder lokaler Nutzung.

In Österreich gibt es heute - Stand Februar 2025 - mehrere Vernetzungen analoger Repeater. Das größte Netz innerhalb Österreichs ist OE-LINK, eine Vernetzung, welche ausschließlich über IP angebundene, analoge Hytera-Repeater nutzt und einen an IPSC2 angelehnten Server zur Vernetzung verwendet. Diese System würde theoretischen eine Vielzahl von Gruppen unterstützen, tatsächlich sind alle Repeater ausschließlich Mitglied der Gruppe "99". Somit ergibt sich auch keine Notwendigkeit diese Zahl zu kommunizieren, die Gruppe ist ausschließlich ein technisches Merkmal, um die einzelnen Repeater zu vernetzen.

Ein großes, für Österreich relevantes Svx-Link-Netz ist Südtirol-Link, eine Vernetzung zahlreicher Repeater in Südtirol, wie auch einzelner Repeater in Tirol und Bayern. Während dieses Netz mehrere TGs unterstützt, so läuft der Großteil der Kommunikation auf der "Defaultgruppe" 88. Weiters gibt es in OE9 mehrere über SvxLink vernetzte analoge Repeater. Diese Repeater verwenden intern die Gruppe "2329" und sind an das deutsche https://fm-funknetz.de/ angebunden über das auch ein "Hotspot"-Einstieg ins Repeater-Netz ermöglicht wird.

Konzept

Der derzeitige Testaufbau sieht vor, dass Repeater entweder vernetzt oder lokal betrieben werden. Eine manuelle Auswahl der TG ist damit normalerweise nicht notwendig. Intern werden mehrere TalkGroups verwendet, inbesondere die TG 232.

In Anlehnung an das schwedische SvxLink-Netz wird **TG 232 für österreichweite Vernetzung** definiert - nach dem schwedischen Vorbild https://svxportal.sm2ampr.net/.

Rundsprüche werden über die TG 232+3+<zwei Ziffern> übermittelt, etwa 301 für den OE-Rundspruch.

Um die Adressierung eines einzelnen Repeaters zu ermöglichen, erhält jeder Repeater eine eindeutige TG und ist Mitglied dieser TG. Die TG wird aus "5/6+<zwei Ziffern> gebildet, etwa 501 für 2m auf OE3XPA. Werden unter einem Rufzeichen mehrere Repeater (z.B. 2m und 6m) betrieben, so können auch mehrere TG zugeordnet werden. Im Bereich 500-699 sind demnach 200 Repeater-Kennungen möglich.

Für lokale/regionale Vernetzungen ist 8+<zwei Ziffern> reserviert. Dies kann etwa zur Kopplung mehrerer Repeater an einem Standort dienen oder einer einer starren Kopplung zweier Repeater.

Die TG 4+<zwei Ziffern> sind temporäre TG, dh. auf diese TG wird gewechselt, sobald ein QSO länger als 5 min auf der TG 232 läuft.

Die TG 9+<zwei Ziffern> sind für Experimente reserviert und können ohne weitere Koordination lokal verwendet werden.

Dzt werden nur dreistellige TGs verwendet. Die TG 000-199 und 700-799 werden dzt. nicht genutzt und sind für künftige Erweiterungen des Adressierungsschemas vorgesehen, ebenso längere oder kürzere TG.

SVX\-TG\-Liste

(Stand 20.2.2025)

- TG 112 keine Vergabe (Überlappung mit der einheitlichen europ. Notrufnummer, es wird allerdings kein Notruf angeboten)
- **TG 232** - österreichweite Vernetzung der teilnehmenden Repeater
- TG 301 OE-Rundspruch
- TG 302 OE1-Rundspruch
- TG 400 - 499 temporäre TG von 232 (nach 5 min Aktivität)
- TG501 OE3XPA 2m
- TG 502 OE3XNR 70cm
- TG 800-899 lokale Vernetzungen
- TG 900 - 999 ist reserviert für Experimente

Konfiguration

In SvxLink ist in der Datei svxlink.conf folgende Konfiguration im Abschnitt [ReflectorLogic] notwendig:

```
[ReflectorLogic]
TYPE=Reflector
HOSTS=<IP SVXREFLECTOR>
HOST_PORT=5300
CALLSIGN="OE3XPA-2m"
AUTH_KEY="re-CHANGE-ME-Noo"
DEFAULT_TG=232
```

```
MONITOR_TGS=232,501,+301,+302
MUTE_FIRST_TX_LOC=0
AUDIO_CODEC=OPUS
QSY_PENDING_TIMEOUT=10
VERBOSE=1
```

Die Einstellung "MUTE_FIRST_TX_LOC=0" verhindert, dass die erste Aussendung ignoriert wird. "+" vor der Gruppe gibt der Gruppe Priorität. Mit "VERBOSE" wird das Logging detailreicher. Wird auf eine temporäre TG gewechselt, so kann mit Drücken der PTT-Taste an "unbeteiligten" Repeatern innerhalb von 10s mitgewechselt werden (QSY_PENDING_TIMEOUT). Die temporären TG brauchen am svxlink ansonsten nicht extra konfiguriert werden. Hier ein Beispiel für ein QSY:

```
Wed 22 Nov 2023 03:48:44 AM CET: OE3XPA: Talker stop on TG #232
Wed 22 Nov 2023 03:48:44 AM CET: Requesting auto-QSY from TG #232 to TG #401
Wed 22 Nov 2023 03:48:44 AM CET: OE3XER: Select TG #401
Wed 22 Nov 2023 03:48:44 AM CET: OE3XNR: Select TG #0
Wed 22 Nov 2023 03:48:44 AM CET: OE3XPA: Select TG #401
Wed 22 Nov 2023 03:48:51 AM CET: OE3XER: Talker start on TG #401
```

Würde der Repeater normalerweise lokal betrieben, so wäre "DEFAULT_TG=501" eingestellt.

Auch wenn in der Bespielkonfiguration mehrere TG angeführt werden, aus Nutzersicht unterstützt der Repeater lediglich österreichweite oder lokale Kommunikation, eine manuelle Auswahl der TG ist nicht notwendig.

SvxLink-PTT

Um Svxlink mit einem Transceiver zu verbinden ist neben den Audiosignalen (RX, TX) auch die Sende-Empfangsumschaltung (PTT) notwendig. Der Squelch wird üblicherweise in Software erkannt (CTCSS).

Es gibt mehrere Möglichkeiten die PTT zu verbinden:

GPIO

Auf Raspberrys ist die einfachste Variante die Verwendung eines GPIO-Pins. Typischerweise wird bei bei der PTT ein Pin gegen des Funkgeräts gegen Masse geschaltet. Ein Transistor mit Basis über einen Spannungsteiler am GPIO und Open-Collector am PTT-Pin des Funkgeräts ist eine einfache Lösung.

Hier ein kurzer Test-Code:

```
import RPi.GPIO as GPIO
import time

# Set up the GPIO pin for PTT
PTT_PIN = 17   # Change this to your pin number
GPIO.setmode(GPIO.BCM)
GPIO.setup(PTT_PIN, GPIO.OUT)

# Function for PTT
def ptt_on():
    GPIO.output(PTT_PIN, GPIO.LOW)  # Pulling to LOW to key the radio

def ptt_off():
    GPIO.output(PTT_PIN, GPIO.HIGH)  # Releasing to HIGH to unkey

try:
    while True:
        ptt_on()  # Key the radio
        time.sleep(5)  # Duration for which to key
        ptt_off()  # Unkey the radio
        time.sleep(5)  # Pause before the next key

except KeyboardInterrupt:
    pass
finally:
    GPIO.cleanup()
```

Die entsprechende Konfiguration in svxlink.conf lautet:

```
PTT_TYPE=GPIO
GPIO_PATH=/sys/class/gpio
PTT_PIN=!gpio0
```

USB\-Serial\-Adapter

Sofern ein gewöhnlicher PC verwendet wird, kann eine über USB hinzugefügte serielle Schnittstelle als PTT verwendet werden. Dabei wird die Handshake-Leitung "RTS" (Request to Send) verwendet. RTS ist Pin 7 am 9-poligen SubD-Stecker. Masse (GND) findet sich auf Pin 5. Die Verbindung zum Funkgerät erfolgt wie bei der Verwendung eines GPIO-Pins. RTS wechselt bei USB-Serial-Adaptern typischerweise zwischen -7 V ("off") und 7 V ("on"). Die Belastung des Pins sollte 5 mA nicht überschreiben, 0.5 mA ist mehr als ausreichend an der Basis eines Transistors, damit genügt ein Spannungsteiler von jeweils 10 kOhm.

Hier ein kurzer Test-Code:

```
#!/usr/bin/python3

# pip install pyserial

import serial
import time

# Configure your serial port and baud rate
serial_port = '/dev/ttyUSB0'  # Change this to your serial port
baud_rate = 9600

# Open the serial port
with serial.Serial(serial_port, baudrate=baud_rate, timeout=1) as ser:
    try:
        while True:
            # Set RTS (key the radio)
            ser.setRTS(True)
            print("PTT ON")
            time.sleep(5)  # Keep PTT on for 5 seconds

            # Clear RTS (unkey the radio)
            ser.setRTS(False)
            print("PTT OFF")
            time.sleep(5)  # Wait for 5 seconds before the next key

    except KeyboardInterrupt:
        print("\nStopped by User")
```

Die entsprechende Konfiguration in svxlink.conf lautet:

```
PTT_TYPE=SerialPin
PTT_PORT=/dev/ttyS0
PTT_PIN=!RTS
```

PTT\-Pins auf USB\-Sound\-Karten

Manche Sound-Chips (z.B. CM108) haben unbenutzte GPIO-Pins, welche als PTT verwendet werden können. Allerdings ist es dazu -abhängig vom Leiterplatten-Design -notwendig. einen Draht direkt am Sound-Chip anzulöten.

SvxLink-UDP

svxlink erlaubt die Anbindung von Audio (sowohl als "Mic", wie als "Lautsprecher") über UDP.

Dazu ist etwa folgender Abschnitt in der Konfiguration von svxlink notwendig:

```
[TxUDP]
TYPE=Local
AUDIO_DEV=udp:44.143.100.200:4300
AUDIO_CHANNEL=0
PTT_TYPE=NONE
CARDS_CHANNELS =1

[RxUDP]
TYPE=Local
AUDIO_DEV=udp:0.0.0.0:4301
AUDIO_CHANNEL=0
SQL_DET=VOX
VOX_FILTER_DEPTH=100
VOX_THRESH=10
```

44.143.100.200 ist die fiktive Adresse des Ziels der UDP-Daten auf Port 4300. Daten für Rx werden am Rechner (Adresse 0.0.0.0) am Port 4301 entgegengenommen. Nachdem es bereits RX und TX gibt ist es nun notwendig die neuen Quellen und Senken zu integrieren. Für "RX" wird ein Voter verwendet, für "TX" ein Multiplexer:

```
[MultiTX]
TYPE=Multi
TRANSMITTERS=Tx1,TxUDP

[Voter]
TYPE=Voter
RECEIVERS=Rx1,Rx1,RxUDP
VOTING_DELAY=200
BUFFER_LENGTH=0
```

An anderer Stelle - etwa in der RepeaterLogic - ist nun die neue Definition statt "Rx1" bzw "Tx1" einzufügen:

```
[RepeaterLogic]
TYPE=Repeater
RX=Voter
TX=MultiTX
```

Die Daten werden mit zwei Kanälen mit jeweils 16 Bits kodiert. Pro Sekunde werden 48 UDP-Pakete mit einer Länge von 4032 Bytes übermittelt, dh. je Paket 1008 Samples mit einer Sample-Rate von 48.000 Samples/s, der wohl internen Sample-Rate von svxlink.

```
[2025-03-10 21:12:45.603] Received 4032 bytes from ('127.0.0.1', 53786):
8eff00006eff000037ff0000fafe0000befe000099fe000094fe0000a7fe0000cefe0000fbfe00
001eff000034ff00003cff00003dff000044ff00005bff000086ff0000c2ff0000000000003500
0000510000004800000001f000000deff000094ff00005aff00003fff00004cff000089ff0000e9
ff00005b000000d90000004e010000af010000f80100002602000039020000370200002202000
```

```
ff010000d5010000a90100008101000063010000500100004901000048010000480100004101000
00310100001501000f1000000ce000000b3000000a6000000a4000000ab000000b4000000a800
0000840000004600000e6ff000077ff00000cff0000b0fe000079fe000072fe000095fe0000dc
fe00003dff0000a0ff0000faff000043000000730000008c0000009700000092000000860000000
730000005400000028000000efff0000a4ff00004fff0000f9fe0000acfe000075fe00005bfe00
0067fe000095fe0000d8fe00002dff000080ff0000c2ff0000eeff000000000000f9ff0000e2ff
0000c5ff0000abff000096ff00008aff000085ff00007fff000076ff00006cff000065ff00006b
ff000086ff0000c4ff00001e000000900000000f01000083010000d6010000fe010000f1010000
b60100005a010000ef000000870000002f000000ebff0000b5ff000084ff000050ff000010ff00
00c9fe000084fe00004afe00002efe000034fe00005afe000097fe0000d8fe00000bff00001cff
000007ff0000d2fe00007ffe000028fe0000e1fd0000b1fd0000a3fd0000b7fd0000e3fd000020
fe000065fe0000affe0000fffe000053ff0000b6ff000022000000920000000501000006b010000
b7010000ec01000002020000fe010000ee010000d9010000c6010000c0010000c4010000c90100
00d7010000e1010000e2010000e7010000ec010000f50100000b0200002c020000520200007a02
000099020000ab020000a702000091020000710200004302000017020000f3010000ce010000ac
0100008901000058010000180100000ce000000760000002000000dbff0000a6ff00008bff0000
8bff000094ff00009fff0000a2ff000087ff000054ff000011ff0000befe00006ffe000037fe00
000efe0000fbfd0000f8fd0000eefd0000d7fd0000abfd000064fd000010fd0000befc00007dfc
00005afc00005bfc00007afc0000abfc0000dcfc000000fd00000efd000006fd0000f4fc0000e6
fc0000e9fc00000cfd00004dfd0000a0fd000004fe000064fe0000b3fe0000f8fe000032ff0000
66ff0000aeff000008000000720000000f70000007c010000ef0100005402000097020000b50200
00c5020000c5020000bd020000c2020000cc020000d3020000e2020000e6020000da020000c802
0000aa020000840200005d020000340200000702000d8010000a30100006c01000030010000f8
000000ca000000a00000007f000000670000004200000014000000dcff00008cff000036ff0000
e8fe00009ffe000072fe00006afe000076fe00009bfe0000d1fe000001ff00002cff000050ff00
0066ff00007bff000097ff0000b9ff0000e9ff000023000000600000009a000000cb000000eb00
0000fd000000020100000201000002010000050100001601000002f0100004c0100007101000091
010000a6010000b3010000b0010000a0010000870100006701000044010000230100000501000
ea000000cf000000b1000000910000006800000039000000a000000d9ff0000adff00008aff00
006dff000058ff000048ff000035ff000020ff00000bff0000effe0000d5fe0000c1fe0000acfe
00009afe000087fe00006afe000040fe00000bfe0000cbfd00008bfd000057fd000039fd00003b
fd000060fd0000a3fd0000f8fd000051fe0000a0fe0000d6fe0000edfe0000e6fe0000c4fe0000
8ffe000056fe00001efe0000eefd0000cefd0000bffd0000bcfd0000cefd0000edfd000014fe00
0048fe00007efe0000acfe0000dcfe000002ff00001bff000036ff000050ff00006aff00008eff
0000b6ff0000ddff0000020000001e000000290000002b000000250000001a0000001e00000033
000000590000009b000000ee00000047010000a7010000010200004c0200008a020000b6020000
cc020000d1020000c6020000aa02000085020000 5d0200003b020000260200002602000003d0200
006b020000a7020000ea020000280300005803000077030000810300007d030000720300006103
0000530300004d030000410300003303000026030000070300 00de020000b3020000790200003d
02000006020000ca010000930100006601000039010000120100 00f9000000dc000000c2000000
ac000000810000004900000006000000a8ff00003cff0000d0fe000059fe0000eafd00008afd00
0029fd0000cffc00007dfc000024fc0000cefb000084fb000043fb00001afb000010fb00001efb
000044fb000078fb0000a8fb0000cefb0000e3fb0000e4fb0000dafb0000cefb0000cafb0000d5
fb0000f0fb00001ffc000056fc000089fc0000c0fc0000f1fc000017fd00004dfd00008dfd0000
d6fd000041fe0000bcfe000038ff0000c5ff000046000000ad0000000a0100004a010000690100
007d0100007d0100006801000059010000460100002e01000290100002a0100002e0100004601
0000660100008701000 0b8010000ee0100002502000065020000a5020000de0200000c03000024
03000020030000f7020000ad0200004a020000d30100005c010000f7000000aa00000082000000
7f00000097000000c2000000f8000000290100005501000 07f010000a3010000cc010000000200
00320200006302000 08d020000910200007002000027020000a90100000a01000062000000bfff
00003dff0000f2fe0000e6fe00001cff00008bff00001d000000c100000060010000e301000041
020000770200008002000064020000 2f020000da01000071010000fa0000006b000000d2ff0000
3cff0000aefe00003dfe0000fbfd0000f0fd00001cfe000076fe0000eefe000067ff0000c8ff00
00feff000000000000d1ff000077ff000008ff00009bfe000030fe0000d3fd000082fd000029fd
0000c5fc000051fc0000cdfb000049fb0000d7fa000093fa00008afa0000c0fa00003dfb0000ea
fb0000adfc000071fd00001cfe00009afe0000e4fe0000fafe0000e2fe0000a7fe000056fe0000
f7fd0000a1fd00005cfd000031fd00003dfd000082fd000000fe0000cafe0000c8ff0000dc0000
000b020000240300000904000 0c30400003b0500007205000083050000740500005105000 02d05
000006050000db040000b10400007e040000440400001204 0000e8030000ce030000d9030000fe
030000360400007c040000b5040000d4040000bf04000073040000f70300003a0300005a020000
6d0100006500000063ff000073fe000083fd0000a7fc0000eefb000047fb0000cafa000084fa00
0064fa000076fa0000b4fa0000fafa000042fb00007efb00008bfb000076fb00004afb0000fdfa
0000b7fa00008cfa000078fa00008efa0000c8fa000019fb000078fb0000d8fb000040fc0000af
fc000024fd0000c2fd000081fe000056ff000054000000590100004402000 01d030000c4030000
2c040000710400009204000096040000a7040000be040000d70400000c050000420500006d0500
009e050000c1050000d1050000e4050000ea050000e3050000d7050000b9050000840500003805
```

```
0000d30400005d040000d10300003e030000b30200001f020000920100001301000088000000ff
ff00007cff0000e7fe000053fe0000cdfd00003efd0000bcfc000053fc0000e9fb000091fb0000
53fb00000ffb0000d5fa0000abfa000072fa00003ffa000017fa0000ebf90000cef90000c9f900
00d3f90000f5f900002bfa000069fa0000aafa0000e2fa000010fb000033fb00004afb00006ffb
0000a5fb0000f2fb00006efc00000cfd0000bcfd000088fe000051ff000000000000a60000002e
0100008d010000e70100002d020000580200008a020000af020000bd020000d2020000dd020000
da020000e9020000ff0200001a030000540300009c030000e90300004a040000a9040000fc0400
00470500007905000008e0500007c05000042050000e504000060040000c6030000290300008e02
00000b020000ae0100006c0100004d0100004901000048010000 4b0100004f0100004301000034
0100002a01000014010000ff000000ed000000bd0000007d00000034000000cbff000060ff0000
0aff0000c2fe0000a6fe0000befe0000f7fe000052ff0000bfff0000250000007f000000c70000
00f70000001601000 02f0100003f0100004c010000560100004d0100002e010000f40000009800
00001f00000094ff000001ff000075fe0000fcfd0000a4fd00006bfd00004ffd00004afd00004b
fd000045fd00002bfd0000f5fc0000a2fc000034fc0000bafb000046fb0000defa000092fa0000
67fa000052fa00004cfa000049fa00003cfa000021fa0000fcf90000dbf90000cdf90000dbf900
001afa000082fa000003fb00009afb00002bfc0000a0fc000003fd00004efd000087fd0000d3fd
000034fe0000a9fe00004bff0000fdff0000a500000051010000e401000050020000bb0200001b
03000074030000ea03000067040000dc04000058050000b9050000f30500001b0600002c060000
2d0600003f0600005f0600008c060000c9060000000700001f0700001c070000ec060000930600
0017060000890500 00ff040000790400000804000 0b303000064030000220300 00ee020000a902
0000620200002002000 0c6010000660100000b01000095000000 1a000000a9ff000022ff0000a6
fe000044fe0000defd00008efd00005dfd000025fd0000fafc0000defc0000b9fc00009dfc0000
94fc000093fc0000a9fc0000dafc000015fd00005cfd0000a7fd0000e3fd000011fe000030fe00
0041fe00004dfe00005dfe00007dfe0000affe0000effe000042ff000098ff0000e3ff00002700
0000560000006c0000007c000000800000007a0000007d0000007d0000007100000063000000 3f
00000001000000b6ff00005aff0000f7fe0000a3fe00005ffe00002ffe00001afe000014fe0000
10fe00000bfe0000f8fd0000d8fd0000adfd00007efd000052fd00002afd00000afd0000ebfc00
00c7fc00009afc000066fc000030fc000007fc0000f9fb000016fc000062fc0000d6fc00006bfd
000007fe00008ffe0000fefe000045ff000066ff000079ff00008bff0000aaff0000ebff000048
000000b40000002d0100009a010000ed0100003302000067020000 8f020000cb02000016030000
6b030000d80300004304000 09c040000e704000016050000280500002805000 01a050000ff0400
00db040000a60400005c040000fa0300007f030000f6020000670200 00e40100007f0100003401
00000c0100000301000000010000fc000000f3000000cc0000009400000056000000 01000000af
ff00006bff00001fff0000e0fe0000
```

Der digitale Kanal kann auch parallel zu einem bestehenden Kanal erstellt werden, etwa durch eine Multi-TRX-Konfiguration:

```
[MultiTX]
TYPE=Multi
TRANSMITTERS=Tx1,TxUDP
```

Einfacher Python-Code zur Ausgabe der UDP-Daten:

```
import socket
from datetime import datetime

def start_udp_server(host='0.0.0.0', port=4300):
    # Create a UDP socket
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    # Bind the socket to the address and port
    server_address = (host, port)
    sock.bind(server_address)

    print(f"Starting UDP server on {host}:{port}. Waiting for data...")

    try:
        while True:
            # Receive data from the client
            data, address = sock.recvfrom(65535)
```

```
            # Get the current timestamp with milliseconds
            timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S.%f')[:-3]

            # Convert the data to hexadecimal format
            hex_data = data.hex()

            # Print the timestamp and the received hex data
            print(f"[{timestamp}] Received {len(data)} bytes from {address}: {
hex_data}")

    except KeyboardInterrupt:
        print("\nServer is shutting down.")
    finally:
        sock.close()

if __name__ == '__main__':
    start_udp_server()
```

Einfacher Pyhton-Code um die Daten auf 8000 Samples/s zu konvertieren:

```
import socket
import numpy as np
from scipy import signal
import wave

def low_pass_filter(data, cutoff, fs, order=5):
    nyquist = 0.5 * fs
    normal_cutoff = cutoff / nyquist
    b, a = signal.butter(order, normal_cutoff, btype='low', analog=False)
    return signal.lfilter(b, a, data)

def start_udp_server_with_processing(host='0.0.0.0', port=4300, output_file='o
utput.wav'):
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.bind((host, port))

    print(f"Listening on {host}:{port}")

    out_wave = wave.open(output_file, 'wb')
    out_wave.setnchannels(1)
    out_wave.setsampwidth(2)
    out_wave.setframerate(8000)

    try:
        while True:
            data, _ = sock.recvfrom(4032)
            samples = np.frombuffer(data, dtype=np.int16)[::2]

            filtered_samples = low_pass_filter(samples, cutoff=3400, fs=48000)

            downsampled_samples = signal.resample_poly(filtered_samples, up=1,
 down=6)

            out_wave.writeframes(downsampled_samples.astype(np.int16).tobytes(
))

    except KeyboardInterrupt:
        print("\nServer shutting down.")
    finally:
```

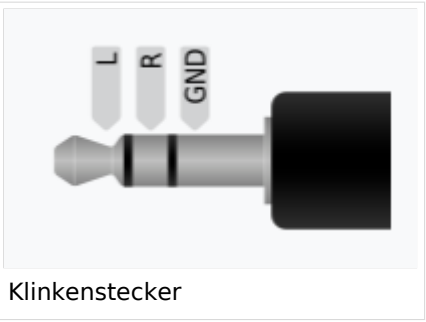
```
        sock.close()
        out_wave.close()

if __name__ == '__main__':
    start_udp_server_with_processing()
```

Das Downsampling und die Ratenadaption basiert auf Samples von lediglich 20ms, damit kommt es zu Artifakten.

SvxLinkAudio

Die Anbindung des Funkgeräts erfolgt üblicherweise über eine analoge Audioschnittstelle. Dazu ist eine Soundkarte am Rechner erforderlich.

Anbindung 3,5 mm Klinke (L = "channel" 0 bei SvxLink üblicherweise verwendet.)



Klinkenstecker

Der Audio-Abgleich ist durchaus herausfordernd, eine ausführliche Beschreibung findet sich in der "man"-Page zu devcal unter

https://github.com/sm0svx/svxlink/blob/master/src/doc/man/devcal.1

Inhaltsverzeichnis

NAME

devcal - An FM deviation calibration utility for the SvxLink system

SYNOPSIS

- devcal [-?|--help] [-h|--usage] [-f|--modfqs=**//frequencies in Hz//**] [-d|--caldev=** //deviation in Hz//**] [-m|--maxdev=**//deviation in Hz//**] [-H|--headroom=**//Headroom in dB//**] [-r|--rxcal] [-F|--flat] [-M|--measure] [-w|--wide] [-a|--audiodev=**//type:dev//**] <**//config file//**> <**//config section//**>**

DESCRIPTION

- devcal** is a utility that is used to calibrate the input and output sound levels (deviation) on an FM SvxLink system. The idea is that it should be possible to calibrate multiple parts of a SvxLink system to the exact same sound levels. This will ensure that a system using multiple receivers have the same sound level for all receivers and for a repeater it means that the audio being retransmitted is at the same level as the received audio. For an EchoLink system it will guarantee that audio received by the local node will be at a proper level to be retransmitted on remote EchoLink nodes, and the other way around.

An RTL2832U based DVB-T USB dongle can be used to measure deviation and the devcal utility can also be used to calibrate the receiving frequency correction for such a dongle.

Deviation calibration can be done both with and without a DVB-T USB dongle. Both methods are described below.

OPTIONS

```
 * ****-?|--help**** Print a help message and exit.
 * ****-h|--usage**** Display a brief help message and exit.
 * ****-f|--modfqs**//frequencies in Hz//** Use this command line option to
set at what frequencies you want to calibrate the deviation. Separate
multiple frequencies using commas. The default frequency is 1000Hz.
 * ****-d|--caldev=**//deviation in Hz//** The deviation value at where to
perform the calibration. The default value of 2404,8Hz may seem strange but
it will be explained below.
 * ****-m|--maxdev=**//deviation in Hz//** The maximum deviation used on the
channel. This is the deviation value where transmitters start to limit the
deviation so as not to cause interference in neighbouring channels. The
default value is 5000Hz.
 * ****-H|--headroom=**//headroom in dB//** The headroom is the margin to add
above the maximum deviation level. Adding a headroom will allow SvxLink to
handle levels above the maximum deviation level without causing immediate
distortion. The default is 6dB which mean that SvxLink can handle twice the
specified maximum deviation. If both maxdev and headroom are left at their
default this will mean that SvxLink can handle 10kHz deviation without
distortion.
```

Changing the headroom cause a lot of different effects so don't do that unless you are prepared to deal with the problems. For example, increasing the headroom will cause the TX level to get lower and the RX level to get higher, which then must be compensated. The announcement levels will get lower so they also need to be compensated. The EchoLink RX/TX levels will get unbalanced and at the moment there is no way to fix that.

```
 * ****-r|--rxcal**** Specify this command line option to perform receiver
calibration.
 * ****-t|--txcal**** Specify this command line option to perform transmitter
calibration.
 * ****-F|--flat**** Perform calibration on a transmitter or receiver that
has a flat frequency response (no pre- or de-emphasis). If the transceiver
has flat frequency response but you have enabled pre-/de-emphasis in SvxLink,
this option should NOT be specified.
 * ****-M|--measure**** The measurement mode requires the use of a RTL2832U
based DVB-T USB dongle and will measure the received deviation.
 * ****-w|--wide**** Use wide FM (broadcast) instead of narrow band FM
 * ****-a|--audiodev=**//type:dev//** Use this command line option to set an
audio device to use for playing back the received audio. The default is to
use "alsa:default". Disable audio output by setting the audio device to the
empty string (i.e. --audiodev="").
```

CALIBRATING AN RTL2832U BASED DVB-T USB DONGLE

All RTL2832U based DVB-T USB dongles requires calibration so that the specified receiving frequency is correct. Most dongles are way off in frequency. 50-60ppm is not uncommon which translates to more than 20kHz on 434MHz.

Most dongles are sensitive to temperature change so start by plugging the dongle into the computer and let it warm up for like 15 minutes before doing the calibration.

Devcal need a receiver configuration file to work so such a file must be created before this utility can be used. Read the Ddr Receiver Section and the Wideband Receiver Section in the **svxlink. conf**(5) manual page for information on how to set it up.

A first coarse calibration should be done in wideband mode so that the calibration carrier is easy to find. You can use any FM carrier to calibrate on that you know is correct in frequency. It does not matter if the carrier is modulated or not. Use your own transceiver or a repeater for example. If the frequency calibration is unknown for the transmitter being used it may be wise to test with multiple transmitters. The command line may look like this:

devcal -Mw /path/to/svxlink.conf RxRtl

The utility will print, among other things, the carrier frequency error. It may look something like:

Tone dev=7.30 Full bw dev=39198.90 Carrier freq err=-27713.53(-64ppm)

We ignore the first two values for now and concentrate on the carrier frequency error. The -64 ppm mean that the dongle is receiving the transmitted carrier about 27kHz below the set frequency. To compensate for this, set the FQ_CORR configuration variable in the wideband receiver configuration section to 64. Do another measurement to verify that the received carrier frequency is now closer to the expected one.

Now we can do the final calibration in narrow band mode. Remove the "w" command line option from the command above and do another measurement. Adjust FQ_CORR until a value around 0ppm is shown.

CALIBRATING THE TRANSMITTER

In order to calibrate the transmitted deviation we need a way to measure it. There are multiple methods to do that. If you have a deviation meter you're in luck but most people does not own one of those. Fortuantely there are other ways.

One way is to use a RTL2832U based DVB-T dongle with the devcal utility. First calibrate the DVB-T dongle according to the instructions in the previous section. The devcal utility can then be used in measurement mode to measure the transmitted FM deviation. The measurement mode is activated using the "-M" command line option. Use the "-f" command line option to specify the audio frequency to calibrate at if the default of 1000Hz need to be changed.

Another way to measure FM deviation without a deviation meter is to use the Bessel null method. It makes use of the fact that the FM carrier will go down to zero power for certain combinations of modulation frequency and deviation. The deviation divided by the modulation frequency give something called the modulation index. Bessel nulls occur at specific modulation indexes where the first one is at 2.4048. So, if we choose the deviation to 2404.8Hz and the modulation frequency to 1000Hz we should get a Bessel null when the transmitter is calibrated. These are the default modulation parameters in devcal.

So how can we detect when the carrier power goes down to zero? If you own some form of spectrum analyzer the spectrum can be watched to see when the carrier falls down to zero. If you own a CW receiver that covers the frequency of interest you can use that to listen to the carrier. Start devcal in transmitter calibration mode using a command looking like the one below.

devcal -t /path/to/svxlink.conf Tx1

In another window, start the alsamixer utility or some other utility to adjust the audio output level. Set the output level to zero. Go back to the devcal window and press 0 to set MASTER_GAIN to zero. Press T to start transmitting. Since we set the sound output level to zero only a carrier will be transmitted. Adjust the CW receiver to center on the carrier. Use the narrowest filter that the receiver support. Now start increasing the audio output level in alsamixer. Listen to the tone to find the first minimum. Use the +/- keys in deval to fine tune the output level to a minimum. You have now found the calibration values for the transmitter. The printed value of MASTER_GAIN should be entered into the configuration section for the transmitter.

CALIBRATING THE RECEIVER

To calibrate the receiver we need a transmitter with a known calibration. If possible, the transmitter that was calibrated above can be used. If it's not possible, another transmitter can be calibrated in the same way to be used as a calibration transmitter. Start the devcal utility something like this:

devcal -r /path/to/svxlink.conf Rx1

Start alsamixer in another window and adjust the input level to get as close as possible to the expected deviation. Use the 0, + and - keys to adjust PREAMP to fine tune the deviation that is shown. When satisfied, enter the PREAMP value into the configuration file in the receiver section.

EXAMPLE: CALIBRATING USING A DVB-T USB DONGLE

This is an example of how the calibration procedure may be performed using a DVB-T USB dongle and an arbitrary transmitter, like a handie transceiver.

The calibration transmitter, a handie transmitter for example, must be able to send some form of modulated tone for this procedure to work. The 1750Hz tone burst that many transmitters are equipped with is a good one to use. It is also possible to use a DTMF tone but the results is not as good as when using a single tone, it seems. If using DTMF, try using the code "A" (1633Hz) or "3" (1477Hz). In this example, the use of 1750Hz tone burst is described.

Use the DVB-T dongle to measure the deviation of the 1750Hz tone.

devcal -M -f1750 /path/to/svxlink.conf RxRtl

Start transmitting with the calibration transmitter and wait for the values to stablilize. Take a note of the value for "tone dev". It may be something like 3200Hz, which is used in the example below. Leave deval running since we will need it later on.

On the system being calibrated, start the receiver calibration.

devcal -r -f1750 /path/to/svxlink.conf Rx1

Transmit using the calibration transmitter and adjust the input level using alsamixer and PREAMP to the correct level for "tone dev".

Make sure that devcal is started in measurement mode like when the deviation on the calibration transmitter was measured above. Then, on the system being calibrated, start the transmitter calibration.

devcal -t -f1750 -d3200 /path/to/svxlink.conf Tx1

The value of 3200 was the one we measured in the first step. Adjust it to match your own measurements. Adjust the output level using alsamixer and MASTER_GAIN to get the correct reading for "tone dev".

To check the calibration, use devcal in measurement mode to measure the deviation on the transmitter when retransmitting a received signal. For a repeater that is easily achieved by transmitting on the receive frequency using the calibration transmitter. The deviation measurement should show the correct value being retransmitted. For a simplex link the parrot can be used to achieve the same thing.

NOTE: The retransmitted deviation may not be exactly the same for some transceivers. This is an issue that remains to find the cause of.

ENVIRONMENT

```
 * **ASYNC_AUDIO_NOTRIGGER** Set this environment variable to 1 if you get an
error about **ioctl: Broken pipe** during devcal startup when using OSS audio.
 * **ASYNC_AUDIO_ALSA_ZEROFILL** Set this environment variable to 0 to stop
the Alsa audio code from writing zeros to the audio device when there is no
audio to write available. ASYNC_AUDIO_UDP_ZEROFILL Set this environment
variable to 1 to enable the UDP audio code to write zeros to the UDP
connection when there is no audio to write available.
```

AUTHOR

Tobias Blomberg (SM0SVX) <sm0svx at svxlink dot org>

REPORTING BUGS

Bugs should be reported using the issue tracker at https:%%//%%github.com/sm0svx/svxlink.

Questions about SvxLink should not be asked using the issue tracker. Instead use the group set up for this purpose at groups.io: https:%%//%%groups.io/g/svxlink

SEE ALSO

- svxlink**(1), **remotetrx.conf**(5), **svxlink.conf**(5), **siglevdetcal**(1)

SvxPortal

SvxReflector und SvxLink Dashboards

Für SvxLink gibt es zahlreiche Web-Dashboards, hier einige Beispiele:

- Poland dashboard von Waldek SP2ONG, https://github.com/FM-POLAND/hs_dashboard_pi and https://github.com/SP0DZ/hotspot.dashboard.pi
- Dashboard von fm-funknetz.de (end of life wegen veralteter PHP-Version) https://github.com/dl1bz/svxlinkdb4rptr
- Dashboard von DL7ATA / DJ1JAY (verwendet am OE-Tetrareflektor) https://github.com/SkyAndy/svxrdb-server
- Dashboard Schweden von SA2BLV https://github.com/sa2blv/SVXportal Live: [1] https://svxportal.sm2ampr.net/

Weitere Infos: http://www.granudden.info/?page=/Ham/Repeatrar/SM5GXQ_en/

API von SvxReflector

In 'svxreflector.conf' kann mit 'HTTP_SRV_PORT=8080' ein REST-Interface aktiviert werden.

Unter der URL 'http.//127.0.0.1:8080/status' kann ein JSON mit dem aktuellen Status des Reflektors abgerufen werden:


```
{
  "nodes": {
    "OE3XSR": {
      "isTalker": false,
      "monitoredTGs": [
        301
      ],
      "projVer": "24.02-71-gcf2ce04b",
      "protoVer": {
        "majorVer": 3,
        "minorVer": 0
      },
      "restrictedTG": false,
      "sw": "SvxLink",
      "swVer": "1.8.99.13",
      "tg": 0
    },
    "OE3XER": {
      "isTalker": false,
      "monitoredTGs": [
        232,
        301,
        302,
        303,
        304,
        502
      ],
      "projVer": "24.02-71-gcf2ce04b",
      "protoVer": {
        "majorVer": 3,
        "minorVer": 0
      },
      "restrictedTG": false,
```

```
      "sw": "SvxLink",
      "swVer": "1.8.99.13",
      "tg": 0
    },
    "OE3XNR": {
      "isTalker": false,
      "monitoredTGs": [
        232,
        301,
        302,
        303,
        502
      ],
      "protoVer": {
        "majorVer": 2,
        "minorVer": 0
      },
      "restrictedTG": false,
      "sw": "SvxLink",
      "swVer": "1.7.99.91",
      "tg": 0
    }
  }
}
```


Unter https://github.com/sm0svx/svxlink/blob/master/src/svxlink/reflector/svxreflector-status wird ein einfaches Pyhton-Script angeboten welches den Inhalt des JSON auf der Konsole ausgibt.

SvxReflector

Inhaltsverzeichnis

Installation

Svxreflector 1.0 ist in Debian 12 enthalten. Talkgroups werden erst ab Version 2.0 unterstützt. Aktuell ist inzwischen Version 3.0, diese ist allerdings nicht mit Version 2.0 kompatibel, dh. svxlink mit der Version 3.0 kann sich nicht mit einem svxreflector der Version 2.0 verbinden.

Die folgende Anleitung gilt für Debian und Derivate (etwa Raspberry Pi OS) und erzeugt neben svxlink (Repeater-Software) auch sxvreflector (Vernetzungs-Software).

Svxlink-Code aus Github clonen

```
cd /opt
apt -y install git
git clone https://github.com/sm0svx/svxlink
cd svxlink/
cat INSTALL.adoc
```

Fehlende Pakete installieren

(hier für Debian 12)

```
apt -y install build-essential cmake doxygen pkg-config \
libsigc++-2.0-dev libasound2-dev libspeex-dev libopus-dev libogg-dev \
libpopt-dev libgcrypt20-dev libgpiod-dev librtlsdr-dev libjsoncpp-dev \
tcl-dev libgsm1-dev libcurl4-openssl-dev groff libssl-dev
```

Build

entsprechend INSTALL.adoc:

```
  cd src
  mkdir build
  cd build
  # QT4 not in Debian 12  (only QT5), skip QT UI
  # cmake .. -DUSE_QT=NO
  # Debian-style variant with further options set
  cmake -DCMAKE_INSTALL_PREFIX=/usr -DSYSCONF_INSTALL_DIR=/etc -
DLOCAL_STATE_DIR=/var -DUSE_QT=OFF -DWITH_SYSTEMD=yes ..

  make
  make doc
  useradd svxlink
  # ptt via gpio
  # usermod -a svxlink -G gpio
  # audio
  sudo usermod -a svxlink  -G audio
  # ptt via serial
  sudo usermod -a svxlink  -G dialout
  sudo make install
  sudo ldconfig
```

Nun sollte sowohl svxlink, wie auch svxreflector verfügbar sein.

Migration von svxreflector 2.0 auf 3.0

Seit der Version 3.0 ist eine Public-Key-Infrastructure (PKI) aufbauend auf OpenSSL enthalten.

Dazu muss ein Zertifikat erstellt werden, die passiert automatisch wenn der Hostname (COMMON_NAME) in der Konfiguration (svxreflector.conf) enthalten ist:

```
# version 3.0
[SERVER_CERT]
COMMON_NAME=svx.oe3xnr.hamip.at
```

Der Common-Name kann über die IP-Adresse in der HamnetDB abgefragt werden. Zur Ausgabe der Hamnet-DB ist ".hamip.at" hinzuzufügen. Beim Start wird die erfolgreiche Erstellung des Zertifikats angezeigt:

```
Tue 04 Feb 2025 05:29:38 PM CET: SvxReflector v1.2.99.26@24.02-71-gcf2ce04b
Copyright (C) 2003-2025 Tobias Blomberg / SM0SVX
Tue 04 Feb 2025 05:29:38 PM CET:
Tue 04 Feb 2025 05:29:38 PM CET: SvxReflector comes with ABSOLUTELY NO
WARRANTY. This is free software, and you are
Tue 04 Feb 2025 05:29:38 PM CET: welcome to redistribute it in accordance
with the terms and conditions in the
Tue 04 Feb 2025 05:29:38 PM CET: GNU GPL (General Public License) version 2
or later.
Tue 04 Feb 2025 05:29:38 PM CET:
Tue 04 Feb 2025 05:29:38 PM CET: Using configuration file: /etc/svxlink
/svxreflector.conf
Tue 04 Feb 2025 05:29:38 PM CET: ----------- Root CA Certificate -----------
Tue 04 Feb 2025 05:29:38 PM CET: Serial No.       :
0x7F875321F6F7ACE0C8A4F2374D130F6DB71D5176
Tue 04 Feb 2025 05:29:38 PM CET: Issuer           : CN = SvxReflector Root CA
Tue 04 Feb 2025 05:29:38 PM CET: Subject          : CN = SvxReflector Root CA
Tue 04 Feb 2025 05:29:38 PM CET: Not Before       : Tue Feb  4 17:00:19 2025
Tue 04 Feb 2025 05:29:38 PM CET: Not After        : Sat Jan 29 17:00:19 2050
Tue 04 Feb 2025 05:29:38 PM CET: ------------------------------------------
Tue 04 Feb 2025 05:29:38 PM CET: ---------- Issuing CA Certificate ----------
Tue 04 Feb 2025 05:29:38 PM CET: Serial No.       :
0x47DD2DC96697A6DB5263A95688582C33D57B2F0B
Tue 04 Feb 2025 05:29:38 PM CET: Issuer           : CN = SvxReflector Root CA
Tue 04 Feb 2025 05:29:38 PM CET: Subject          : CN = SvxReflector Issuing
CA
Tue 04 Feb 2025 05:29:38 PM CET: Not Before       : Tue Feb  4 17:00:19 2025
Tue 04 Feb 2025 05:29:38 PM CET: Not After        : Fri Jan 30 17:00:19 2026
Tue 04 Feb 2025 05:29:38 PM CET: ------------------------------------------
Tue 04 Feb 2025 05:29:38 PM CET: Server private key file not found.
Generating '/var/lib/svxlink/pki/private/svx.oe3xnr.hamip.at.key'
Tue 04 Feb 2025 05:29:38 PM CET: Generating server certificate signing
request file '/var/lib/svxlink/pki/csrs/svx.oe3xnr.hamip.at.csr'
Tue 04 Feb 2025 05:29:38 PM CET: -------- Certificate Signing Request -------
Tue 04 Feb 2025 05:29:38 PM CET: Subject          : CN = svx.oe3xnr.hamip.at
Tue 04 Feb 2025 05:29:38 PM CET: Subject Alt Name : DNS:svx.oe3xnr.hamip.at
Tue 04 Feb 2025 05:29:38 PM CET: ------------------------------------------
Tue 04 Feb 2025 05:29:38 PM CET: Generating server certificate file '/var/lib
/svxlink/pki/certs/svx.oe3xnr.hamip.at.crt'
Tue 04 Feb 2025 05:29:38 PM CET: ------------ Server Certificate ------------
Tue 04 Feb 2025 05:29:38 PM CET: Serial No.       :
0x5DCFD4727D7A3C6D749173D0561F747CC7918456
Tue 04 Feb 2025 05:29:38 PM CET: Issuer           : CN = SvxReflector Issuing
```

```
CA
Tue 04 Feb 2025 05:29:38 PM CET: Subject          : CN = svx.oe3xnr.hamip.at
Tue 04 Feb 2025 05:29:38 PM CET: Not Before       : Tue Feb  4 17:29:38 2025
Tue 04 Feb 2025 05:29:38 PM CET: Not After        : Mon May  5 18:29:38 2025
Tue 04 Feb 2025 05:29:38 PM CET: Subject Alt Name : DNS:svx.oe3xnr.hamip.at
Tue 04 Feb 2025 05:29:38 PM CET: ----------------------------------------
```

Damit sich Clients verbinden können ist auch auf der Clientseite eine entsprechende Konfiguration notwendig. Im Wesentlichen passiert diese automatisch, doch es ist zu beachten, dass der Verbindungsaufbau über exakt jenen Hostnamen erfolgt, der auch am Server als COMMON_NAME eingetragen wurde. Steht am svxlink-Rechner DNS nicht zur Verfügung, dann kann über einen Eintrag in /etc/hosts die Übersetzung von Hostnamen zu IP erfolgen. Hier ein erfolgreicher Start aus Client-Sicht:

```
Tue Feb  4 17:43:20 2025: Connected to APRS server 109.72.122.50 on port 14580
Tue Feb  4 17:43:20 2025: ReflectorLogic: Connection established to
44.143.55.139:5300 (primary)
Tue Feb  4 17:43:20 2025: ReflectorLogic: Setting up encrypted communications
channel
Tue Feb  4 17:43:20 2025: ReflectorLogic: Encrypted connection established
Tue Feb  4 17:43:20 2025: ReflectorLogic: Sending requested Certificate
Signing Request to server
Tue Feb  4 17:43:20 2025: ReflectorLogic: Authentication OK
Tue Feb  4 17:43:20 2025: ReflectorLogic: Connected nodes: OE3XNR, OE3XER
Tue Feb  4 17:43:20 2025: ------ Opus encoder parameters ------
Tue Feb  4 17:43:20 2025: Frame size           = 320
Tue Feb  4 17:43:20 2025: Complexity           = 9
Tue Feb  4 17:43:20 2025: Bitrate              = 20000
Tue Feb  4 17:43:20 2025: VBR                  = YES
Tue Feb  4 17:43:20 2025: Constrained VBR      = YES
Tue Feb  4 17:43:20 2025: Maximum audio bw     = MEDIUMBAND
Tue Feb  4 17:43:20 2025: Audio bw             = FULLBAND
Tue Feb  4 17:43:20 2025: Signal type          = VOICE
Tue Feb  4 17:43:20 2025: Application type     = AUDIO
Tue Feb  4 17:43:20 2025: Inband FEC           = NO
Tue Feb  4 17:43:20 2025: Expected Packet Loss = 0%
Tue Feb  4 17:43:20 2025: DTX                  = NO
Tue Feb  4 17:43:20 2025: LSB depth            = 16
Tue Feb  4 17:43:20 2025: ---------------------------------------
Tue Feb  4 17:43:20 2025: ------ Opus decoder parameters ------
Tue Feb  4 17:43:20 2025: Gain       = 0dB
Tue Feb  4 17:43:20 2025: ---------------------------------------
Tue Feb  4 17:43:20 2025: ReflectorLogic: Using audio codec "OPUS"
Tue Feb  4 17:43:20 2025: ReflectorLogic: Using UDP cipher id-aes128-GCM
Tue Feb  4 17:43:20 2025: ReflectorLogic: Bidirectional UDP communication
verified
```

Der genaue Ablauf ist dabei folgender (Quelle: https://groups.io/g/svxlink/topic/certificate_authentication/107698563):

Reflector side

1. The reflector is started
2. Checks if there already is a valid root CA key and cert (private/svxreflector_root_ca.key, certs/svxreflector_root_ca.crt). If not, generate those files. The root CA certificate is self-signed.

3. Checks if there already is a valid issuing CA key, csr and cert (private/svxreflector_issuing_ca.key, csrs/svxreflector_issuing_ca.csr, certs/svxreflector_issuing_ca.crt). If not, generate those files. The issuing CA certificate is signed using the root CA.
4. Checks if there is a server key, csr and cert for the reflector (private/myreflector.org.key, csrs/myreflector.org.csr, certs/myreflector.org.crt). If not, generate those files and sign the server certificate using the issuing CA.
5. Checks if there is a ca-bundle.crt file. If not, copy the root CA certificate file to ca-bundle.crt.
6. When a CSR is received from a node it is stored in pending_csrs
7. When the sysop issue the sign command, sign a CSR in the pending_csrs directory. Move the CSR to the csrs directory and store the signed certificate in the certs directory. When the node connects or already is connected, send the certificate to the node.

Node side

1. The SvxLink node is started
2. Checks if there already is key- and csr-files (CALL.key, CALL.csr). If not, generates them.
3. Connects to the reflector
4. Checks if there already is a ca-bundle.crt. If not, downloads it from the reflector server.
5. Initiates encrypted connection
6. Verifies the reflector server certificate against ca-bundle.crt
7. Verifies that the hostname used to connect to the reflector match the hostname (CN) in the certificate
8. If there already is a crt-file (CALL.crt), sends it to the reflector to authenticate. If not, sends the CSR to the reflector. The reflector will store the CSR in the pending_csrs directory.
9. If the reflector has a signed node certificate (CALL.crt), sends it to the node. The node will then disconnect and reconnect authenticating using the certificate.
10. Retry if failed.

Sprachdateien

Für die Nutzung von svxreflector ist eine aktuelle Version der Sprachdateien am Repeater notwendig, zuletzt wurden folgende Sprachdateien ergänzt:

```
- Core/talk_group
- Core/qsy
- Core/ignored
- Core/monitor
- Default/previous
```

Die Sound-Dateien werden je nach Konfiguration abgelegt, etwa unter /usr/share/svxlink/sounds

Diese Dateien sind auch im Download verfügbar.

TETRA-Vernetzung/TETRA prepare svxlink

SVXLINK Installation RASPI mit BUSTER

Aktualisierte Installationsanleitung: SvxReflector

```
 * SD-Karte (16 GByte empfohlen) mit Raspberry "Raspberry Pi OS (32-bit)
Lite" vorbereiten.
   * Link: https://www.raspberrypi.org/downloads/raspberry-pi-os/
```

```
 * SSH Terminal starten
```

```
 * Grundkonfiguration
   * sudo apt-get upgrade
   * sudo apt-get update && sudo apt-get -y install g++ libsigc++-2.0-dev
libgsm1-dev libpopt-dev tcl-dev libgcrypt20-dev libspeex-dev libasound2-dev
make alsa-utils git cmake libqt4-dev libopus-dev opus-tools libcurl4-gnutls-
dev libjsoncpp-dev
   * Fragen jeweils mit "Y" (bzw. "J" wenn auf deutsch installiert)
beantworten
   * Dieser Vorgang dauert länger. Je nach INTERNET Zugangsgeschwindigkeit
```

```
 * User für svxlink-Echolink anlegen
   * sudo useradd -c 'Echolink user' -G audio -d /home/svxlink -m -s /sbin
/nologin svxlink
```

```
 * SVXLINK Installation aus dem GITHUB
   * git clone https://github.com/sm0svx/svxlink.git
   * cd svxlink
   * mkdir src/build
   * cd src/build
   * cmake -DUSE_QT=OFF -DCMAKE_INSTALL_PREFIX=/usr -DSYSCONF_INSTALL_DIR=
/etc -DLOCAL_STATE_DIR=/var -DCMAKE_BUILD_TYPE=Release ..
   * make
   * sudo make install
```

```
 * SVXLINK /etc/svxlink/svxlink.conf anpassen
```

```
 * Sound-Files
   * cd /usr/share/svxlink/sounds/
   * sudo wget https://github.com/sm0svx/svxlink-sounds-en_US-heather/releases
/download/19.09/svxlink-sounds-en_US-heather-16k-19.09.tar.bz2
   * sudo tar xvjf svxlink-sounds-en_US-heather-16k-19.09.tar.bz2
   * sudo ln -s en_US-heather-16k en_US
```