CCUBE Quickstart Guide

Near Real-Time Streaming of

Seismic Data

1 Introduction

This document describes the necessary steps to operate a CCUBE together with a DATA-CUBE³ (type 2) and a Geophone. The setup with other sensors like seismometers, accelerometers, infrasound and others is similar except the DATA-CUBE³ setup (see DATA-CUBE³ user manual) and the cabling for power via breakout box and sensor connection.

The CCUBE is designed to allow live streaming of seismic data recorded by the DATA-CUBE³ via SeedLink in miniSEED format. The following communication types are supported via

- LTE (also UMTS, EDGE, GPRS)
- WiFi (802.11bgn): Through another router with broadband uplink
- Ethernet (100MBit/s): Through another DSL or SAT router with broadband uplink
- WiFi meshing to another router or CCUBE with a broadband connection

The CCUBE can be configured to automatically connect to an existing OpenVPN network. This is necessary in order to access the CCUBE or SeedLink stream when the CCUBE does not have a public IP.

It is highly recommended to perform a test configuration & installation before deployment in the field.

2 Accessories

2.1 Cables

The following cables are available for the CCUBE:

Power cable
This cable is used to feed the CCUBE with external power (10.5-24V DC).

CCUBE ←→ DATA-CUBE³ cable (black)
This cable is used to stream the data from the DATA-CUBE³ to the CCUBE.

Combi cable (ends with Ethernet, terminal & USB connector)
This cable is used for initial configuration via serial terminal (black USB) or Ethernet connection, and for live streaming via Ethernet (e.g. DSL or SAT router). Additionally, the grey USB cable can be used to attach an USB memory stick.

2.2 Antennas & Wireless Communication

The CCUBE is delivered without SIM card, LTE/UMTS and WiFi antennas. Depending on the final setup the user must obtain these additional items:

- LTE/UMTS SIM card in mini-SIM size
- LTE/UMTS antenna and antenna cable with FME (female) connector
- WiFi antenna and antenna cable with RP-SMA (male) connector

There are different types of LTE & WiFi antennas available on the market. The user must decide which antenna fits best for the desired campaign or setup. A *general* recommendation cannot be provided.

2.3 Software

2.3.1 SeedLink & Streaming

A SeedLink client software is necessary to receive the miniSEED streams from a CCUBE. There are different software packages available which vary by the number of features, usability, price and support. Two examples of SeedLink client software packages are

- SeisComP3 (https://www.seiscomp3.org)
- ObsPy (https://github.com/obspy/obspy/)

The slinktool software (https://github.com/iris-edu/slinktool) can easily be used to verify the CCUBE's live streaming. The manual of slinktool describes it as s "SeedLink client for data stream inspection, data collection and server testing".

2.3.2 VPN

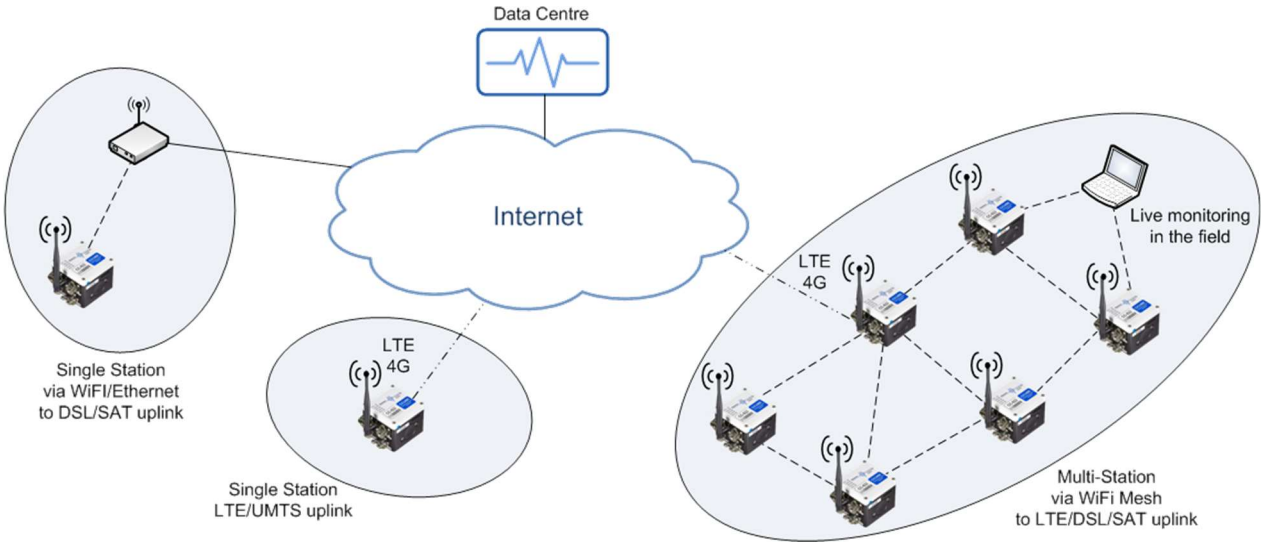
The SeedLink protocol is designed in such a way that the *server* is running on the CCUBE and the *client* in e.g. a central data acquisition system. Server and client are used in the definition of TCP/IP in which the *client* must actively establish the TCP/IP connection to the *server* (CCUBE).

In many real world installations the CCUBE (server) will be installed with an internet uplink behind a firewall or in a private network where it is not directly reachable by the data acquisition system (client). This is true for most DSL and almost all LTE/UMTS connected devices. In a DSL setup it might still be possible to work around by port forwarding if the DSL router receives a public IP address from the provider. But in a LTE/UMTS setup this will not work. The reason is that most if not all LTE/UMTS providers worldwide do not provide public IP addresses to the devices of their customers and a forwarding rule would not work in this configuration.

Therefore, it is recommended or even necessary to connect the CCUBE to a Virtual Private Network (VPN) in order to receive the seismic data stream. The CCUBE supports OpenVPN which requires an OpenVPN server operated by the user.

3 Hardware setup

The CCUBE was designed primarily as an add-on to the DATA-CUBE³. It is capable to operate in different scenarios as depicted below



This quickstart guide assumes one single station comprising a DATA-CUBE³ and a geophone or another sensor. The aim of this quickstart guide is to descript the necessary setup steps to enable near real-time data streaming of a single station with a CCUBE via Ethernet, WiFi or LTE/UMTS.

3.1 DATA-CUBE³

First, the DATA-CUBE³ must be running:

1. Connect an external GPS antenna to the “GPS-ANT.” connector (type 2 only)
2. Connect the Geophone to the “SIG. INPUT” connector
3. Connect the black data cable to the “USB/TERM” connector (the cable between DATA-CUBE³ and CCUBE)
4. Connect the 12V power supply to the “POWER” connector

This quickstart guide assumes that the operation of the DATA-CUBE³ and the geophone or another sensor has been established according to the DATA-CUBE³ user manual and it is verified that seismic data is being recorded to the internal DATA-CUBE³ memory. If an active sensor is used than the power supply of the DATA-CUBE³ is most likely provided through the breakout box (BOB).

3.1.1 DATA-CUBE³ Configuration

The configuration of a DATA-CUBE³ must be adapted for each experiment and uploaded to the DATA-CUBE³ via USB cable (gray) before deployment in the field. For data streaming via SeedLink it is required to operate the DATA-CUBE³ in **continuously GPS mode**. The corresponding part of the configuration can be found below. Please refer to the DATA-CUBE³ configuration file manual for a detailed description of the configuration file.

```
** GPS Mode 0 = cycled // 1 = continuously
GPS_ON=1
```

The SeedLink CUBE plugin requires several parameters to correctly decode the data stream coming from the DATA-CUBE³. These parameters are normally found in the file header of each file which is written by the DATA-CUBE³ to its internal memory. These parameters are also found in the data stream but only if the DATA-CUBE³ starts writing measurements to a new file (at start-up or at midnight). To be able to start the SeedLink CUBE plugin independent of the DATA-CUBE³, a copy of all necessary parameters is stored in a separate configuration file. In particular the following configuration parameters are needed:

Parameter	Description	Default
S_RATE	Sample-rate	100
CH_NUM	Number of active channels	3
D_FILT	Filter delay (in milliseconds / 10)	31

The correct parameters (especially D_FILT which is the “Filter Delay” parameter used for the CUBE plugin configuration below) can be determined after the configuration of the DATA-CUBE³ was completed and a first data file was written to the internal memory. The Cube-Tools the program `cubinfo` can then be used to extract the parameters from the file header of a raw data file:

```
$ cubeinfo 11061538.AVF
# --- CUBE FILE ----------------------------------------------------------
#                file name:  11061538.AVF
# --- SYSTEM -------------------------------------------------------------
#          firmware version:  V2.0T
#            recording unit:  c0AVF
#           experiment name:  N/A
#              file segment:  #0
# --- DIGITIZER ----------------------------------------------------------
#         recorded channels:  p0 p1 p2
#               sample rate:  100 samples per second
#         optimization mode:  low power consumption
#      amplification factor:  16
#        chopping amplifier:  on
# digital high pass filter:  off
#              filter delay:  0.31 seconds
#      time base correction:  off
#           begin recording:  on valid GPS signal
# --- GPS ----------------------------------------------------------------
#                 GPS mode:  powered on for 5 minutes every 30 minutes
#          initial position:  N/A
#        GPS backup battery:  3.15 V
....
```

The default file for DATA-CUBE³ parameters is located on the CCUBE at `/etc/seedlink/cube.txt` and will be configured later in this quickstart guide.

3.2 CCUBE

In the second step, the CCUBE hardware is set up:

1. Connect an external Wi-Fi antenna to the “WLAN” connector (optional)
2. Connect an external GPRS/UMTS/LTE antenna to the “UMTS/LTE” connector (optional)
3. Connect the data cable coming from the DATA-CUBE³ to the “CUBE” connector
4. Connect the combi cable to the “Data out” connector
5. Connect a 12V power supply or 12V battery to the “POWER” connector (**don’t mix + and -**)

The CCUBE starts up automatically as soon as it is powered. The LEDs will indicate that power is available.

4 Initial Access & Configuration

The CCUBE runs a full featured Debian Stretch Linux which can be accessed for configuration and monitoring purpose as well as file exchange. There are two ways to access the device which are described in the next subsections.

4.1 Local Terminal Access

The local terminal access is done via a direct serial connection between the host computer and the CCUBE. The method requires a terminal program. Depending on the operating system of the computer one of the following programs can be used:

- Linux terminal programs: screen, minicom, Kermit, PuTTY
- Windows terminal programs: Hyperterminal, PuTTY

Connect the black USB cable of the combi cable to your computer. Depending on your operating system the cable internal serial converter will be recognised differently:

- Linux: ttyUSBx usually /dev/ttyUSB0
- Windows: COMx usually COM10

Setup your terminal program for the recognised serial converter device with the following settings:

- **Serial Settings: 115200, 8N1**

If the above steps are carried out before the CCUBE is powered on, the boot process can be monitored in the terminal. Otherwise, if the CCUBE was already powered on before the terminal connection was made; the user **should press "Enter" once to see the login prompt**. The initial login credentials are:

- **Login: ccube-admin**
- **Password: ccube$%009**

```
Login: ccube-admin
Password: ***********
Linux CC0XX 4.9.40-ccube #13 Sat Jan 02 20:51:30 UTC 2018 armv5tejl
Have a lot of fun...

Last login: Tue Jan 02 14:53:57 2018 from 192.168.1.12
ccube-admin@CC0XX:~$
```

4.2 Network access

For network access the IP address of the device must be known. Initially, the CCUBE is configured to obtain an IP address automatically via DHCP through the Ethernet connection (combi cable).

If a DHCP server is present in the network, the IP address can be determinate by the following steps:

1. Connect the Ethernet cable from the combi cable to your local network (a DHCP server in this network should automatically offer an IP address to the CCUBE)
2. Establish a terminal connection according to the description in the previous section of this manual
3. Obtain the current IP address of the CCUBE with the following command:

```
ccube-admin@CC0XX:~$ ip addr show
...
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group de-
fault qlen 1000
    link/ether 00:04:25:04:00:56 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.25/24 brd 192.168.1.255 scope global eth0
...
```

In this example the IP address is 192.168.1.25 as highlighted. Using the IP address a remote terminal can be established via SSH and files can be transferred via SCP/SFTP.

4.2.1 Remote Terminal

Depending on the operating system of the host computer one of the following programs can be used:

- Linux: ssh
- Windows: PuTTY

The initial default credentials are the same as for the local terminal access

- **Login: ccube-admin**
- **Password: ccube$%009**

Using a standard OpenSSH client under Linux a connection can be established as follows:

```
ssh ccube-admin@192.168.1.25
Password: ***********
Linux CC0XX 4.9.40-ccube #13 Sat Jan 02 20:51:30 UTC 2018 armv5tejl
Have a lot of fun...

Last login: Tue Jan 02 14:53:57 2018 from 192.168.1.12
ccube-admin@CC0XX:~$
```

4.2.2 Transferring Files

Files from and to the CCUBE can be transferred via SCP/SFTP. Depending on the operating system of the host computer one of the following programs can be used:

- Linux: scp, sftp
- Windows: WinSCP

Connection details are the same as for the remote terminal access.

4.3 System Software Update

The software running on the CCUBE can be updated in the following way:

```
ccube-admin@CC0XX:~$ sudo apt-get update
Get:1 http://deb.ccube.digos.eu/debian stretch InRelease [1207 B]
Get:3 http://security.debian.org stretch/updates InRelease [63.0 kB]
Ign:2 http://cdn-fastly.deb.debian.org/debian stretch InRelease
Get:4 http://cdn-fastly.deb.debian.org/debian stretch Release [118 kB]
Get:5 http://cdn-fastly.deb.debian.org/debian stretch Release.gpg [2434 B]
Get:6 http://deb.ccube.digos.eu/debian stretch/main armel Packages [2532 B]
Get:7 http://security.debian.org stretch/updates/non-free armel Packages [1268 B]
Get:8 http://security.debian.org stretch/updates/main armel Packages [263 kB]
Get:9 http://security.debian.org stretch/updates/contrib armel Packages [1092 B]
Get:10 http://cdn-fastly.deb.debian.org/debian stretch/contrib armel Packages
Get:11 http://cdn-fastly.deb.debian.org/debian stretch/non-free armel Packages
Get:12 http://cdn-fastly.deb.debian.org/debian stretch/main armel Packages
Fetched 7466 kB in 40s (185 kB/s)
Reading package lists... Done

ccube-admin@CC0XX:~$ sudo apt-get upgrade
Reading package lists... Done
Building dependency tree... Done
Calculating upgrade... Done
The following packages will be upgraded:
  ccube-tools  libdns-export162  libirs-export141  libisc-export160  libisccc-ex-
port140 libisccfg-export140 libssl1.0.2 libxml2 rsync sensible-utils
10 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 3800 kB of archives.
After this operation, 48.1 kB disk space will be freed.
Do you want to continue? [Y/n] Y
```

This also updates the ccube tools, libraries and other programs.

Alternatively, the update process can be also done with the following commands:

```
ccube-admin@CC0XX:~$ sudo apt update
...
ccube-admin@CC0XX:~$ sudo apt list --upgradable
...
ccube-admin@CC0XX:~$ sudo apt upgrade
...
```

In case of a boot loader or the Linux kernel update, the internal boot partition must be synchronised by calling **ccube-update-boot.** This script will mount the boot partition and copy all necessary files.

4.4 Logging

The underling Linux system uses **journald** for logging. Log message can be displayed and searched via **journalctl**.

4.5 Configuration

The CCUBE provides a number of command line tools for configuration. These tools cover the most common use cases. For specialised use cases the generated configuration files can be used as a starting point for a custom setup. The ccube command line tools consist out of the following programs:

- **ccube-network**
- **ccube-datalogger**
- **ccube-sensors**
- **(ccube-hardware)**

All ccube tools must be executed as user “root” (administrator). The following command is used to switch from user “ccube-admin” to user “root”:

```
ccube-admin@CC0XX:~$ sudo su -

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for ccube-admin:
root@CC0XX:~# id
uid=0(root) gid=0(root) groups=0(root)
```

4.6 Network & Communication Settings

The network and communication settings are configured by the program “**ccube-network**”. This program can be executed with the parameter “**–h**” to display helpful usage information:

```
root@CC0XX:~# ccube-network -h
usage: ccube-network [-h] [-v] [-S] <component> ...

CCUBE network management

optional arguments:
  -h, --help     show this help message and exit
  -v, --verbose  increase output verbosity
  -S, --syslog   Output to syslog only

Network subsystems:
  <component>
    wifi         WiFi management
    mobile       Mobile network management
    ethernet     Ethernet management
    openvpn      Open VPN management
```

Please note: It is recommended to shutdown a network interface (if running) with ifdown before performing a reconfiguring of the interface.

Please note: The terminal access can be used to recover from a broken, unknown or wrong network configuration. Additionally, the configuration program automatically saves a backup of old configurations in /var/lib/ccube/

4.6.1 Ethernet Configuration

The CCUBE Ethernet interface can be configured either with a static or dynamic IP address. The user can also decide to configure the CCUBE to acts as a dedicated DHCP server.

```
root@CC0XX:~# ccube-network ethernet -h
usage: ccube-network ethernet [-h] <command> ...

optional arguments:
  -h, --help  show this help message and exit

Subcommands:
  <command>
    status    Queries ethernet status
    off       Disable ethernet interface
    dhcp      Setup ethernet interface for DHCP
    static    Setup interface with static IP
    server    Setup CCUBE as DHCP server
```

Again, the usage for particular settings will be shown with parameter "**-h**":

```
root@CC0XX:~# ccube-network ethernet dhcp -h
usage: ccube-network ethernet dhcp [-h] [--metric METRIC]

optional arguments:
  -h, --help       show this help message and exit
  --metric METRIC  Network metric (default: 10)
```

The following command is to set the Ethernet configuration to automatically retrieve the IP, routing and DNS configuration via DHCP:

```
root@CC0XX:~# ccube-network ethernet dhcp
Configure ethernet interface for DHCP
Wrote backup of configuration file to: /var/lib/ccube/01_eth0_201801012053
Wrote new configuration to /etc/network/interfaces.d/01_eth0
Use reboot or ifdown/ifup to activate changes
```

The configuration is set permanently, but is not activated immediately. As the output explains, either a reboot or the ifup command shall be used to apply the configuration to the interface "eth0":

```
root@CC0XX:~# ifup eth0
ifup: interface eth0 already configured
```

In this example, the Ethernet interface was already configured in DHCP mode and no changes are necessary.

Once the configuration was changed by **ccube-network** it is saved and will be used at the next reboot too.

Please note: By default the Ethernet interface will get the lowest network metric and therefore the highest priority.

4.6.2 WiFi Configuration

The CCUBE WiFi can be configured either with a static or dynamic IP address. The user can also decide to configure the CCUBE to acts as a dedicated DHCP server.

```
root@CC0XX:~# ccube-network wifi -h
usage: ccube-network wifi [-h] <command> ...

optional arguments:
  -h, --help  show this help message and exit

Subcommands:
  <command>
    status    Queries WiFi status
    off       Disable WiFi network configuration
    dhcp      Add CCUBE to existing WiFi network (DHCP)
    static    Add CCUBE to existing WiFi network (static)
    adhoc     Add CCUBE to a peer-to-peer network
    mesh      Add CCUBE to a mesh network
```

The most common configuration to attach a CCUBE to an existing WiFi network is via DHCP:

```
root@CC0XX:~# ccube-network wifi dhcp -h
usage: ccube-network wifi dhcp [-h] [--metric METRIC] essid passwd

positional arguments:
  essid            WiFi network ESSID
  passwd           Network password

optional arguments:
  -h, --help       show this help message and exit
  --metric METRIC  Network metric (default: 20)
```

The following example shows how to connect to a WiFi network with the ESSID (name) “Hotspot” with the password “supersecret”:

```
root@CC0XX:~# ccube-network wifi dhcp "Hotspot" "supersecret"
Configure WiFi in managed mode using DHCP...
Wrote backup of configuration file to: /var/lib/ccube/02_wlan0_201801232112
Wrote new configuration to /etc/network/interfaces.d/02_wlan0
Use reboot or ifdown/ifup to activate changes
```

The ifup command is use to activate the WiFi interface “wlan0”:

Please note: By default the WiFi interface will get the second lowest network metric and therefore prioritised over mobile data connections.

```
root@CC0XX:~# ifup wlan0
Enable WiFi module...
Power up WiFi module
Waiting for WiFi module...
Internet Systems Consortium DHCP Client 4.3.5
Copyright 2004-2016 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/wlan0/7c:dd:90:ce:17:c9
Sending on   LPF/wlan0/7c:dd:90:ce:17:c9
Sending on   Socket/fallback
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 3
DHCPDISCOVER on wlan0 to 255.255.255.255 port 67 interval 5
DHCPREQUEST of 192.168.1.3 on wlan0 to 255.255.255.255 port 67
DHCPOFFER of 192.168.1.3 from 192.168.1.1
DHCPACK of 192.168.1.3 from 192.168.1.1
bound to 192.168.1.3 -- renewal in 40631 seconds.
```

The status of the WiFi can be retrieved in the following way:

```
root@CC0XX:~# ccube-network wifi status
Query WiFi status...
Interface wlan0
        ifindex 5
        wdev 0x100000001
        addr 7c:dd:90:ce:17:c9
        ssid Hotspot
        type managed
        wiphy 1
        channel 6 (2437 MHz), width: 20 MHz, center1: 2437 MHz
        txpower 20.00 dBm
```

Once configured this setup is saved and will be used at the next reboot too.

The configuration of the other WiFi modes works accordingly and is explained by the usage information shown with the "**-h**" parameter as given in the examples above.

4.6.3 Mobile Network Configuration (LTE/UMTS)

The mobile network LTE configuration requires an activated SIM card (preferably without a PIN code [see below]), the correct APN of the provider and an antenna.

Two Allen key screws must be unscrewed at the side of the CCUBE to access the SIM card slot.
The SIM card must be inserted with the metal contacts to the bottom and the “missing edge” to the front as shown in the picture below:

Next, the LTE/UMTS antenna must be connected to the CCUBE.

The screws must not be fixed too tight. They will only hold the cap in place.

For the initial configuration the LTE hardware module of the CCUBE must be switched on manually by the **ccube-hardware** command:

```
root@CC0XX:~# ccube-hardware mobile enable
Enable LTE module...
Switch USB port to LTE module
Power up LTE module
Waiting for LTE module...
Switching LTE module into WWAN/QMI mode
```

This command may take a while, but will finally active the LTE hardware module and the blue status LED “UMTS” at the top of the CCUBE. Now the ccube-network command can be used for the configuration.

```
root@CC0XX:~# ccube-network mobile -h
usage: ccube-network mobile [-h] <command> ...

optional arguments:
  -h, --help  show this help message and exit

Subcommands:
  <command>
    lock      SIM lock management
    reconnect
              Reconnect network
    connect   Connect network
    disconnect
              Disconnect network
    status    Queries connection status
    off       Disable UMTS/LTE modem connection
    provider  Setup UMTS/LTE network connection
```

First, it is checked if the SIM card has a PIN set, and this PIN will be removed as shown below (PIN in this example is 1234):

```
root@CC0XX:~# ccube-network mobile lock status
Sending: AT+CPIN?

+CPIN: SIM PIN

OK
root@CC0XX:~# ccube-network mobile lock unlock 1234
Sending: AT+CPIN?

+CPIN: SIM PIN

OK
Sending: AT+CPIN=1234

OK
root@CC0XX:~# ccube-network mobile lock disable
Sending: AT+CLCK="SC",2

+CLCK: 1

OK
root@CC0XX:~# ccube-network mobile lock status
Sending: AT+CPIN?

+CPIN: READY

OK
```

In the next step the mobile connection is configured. Therefore, the correct APN is required. A complete list of all mobile providers worldwide and their corresponding APNs can be found on the following web-site:

https://mobilebroadbandprovider.info/

In the example below the provider “Blau” (from Germany) with the APN “internet.eplus.de” will be used:

```
root@CC0XX:~# ccube-network mobile provider internet.eplus.de
Configure UMTS/LTE modem
Wrote new configuration to /etc/network/interfaces.d/03_wwan0
Use reboot or ifdown/ifup to activate changes
```

The connection with the mobile interface wwan0 can be tested the following way:

```
root@CC0XX:~# ifup wwan0
Enable LTE module...
Switch USB port to LTE module
Power up LTE module
Waiting for LTE module...
Switching LTE module into WWAN/QMI mode
Sending: AT+COPS=0

OK
Sending: AT+CFUN=1

OK
Sending: AT+CGDCONT=1,"IPV4V6","internet.eplus.de"

OK
Sending: AT+CREG?

+CREG: 0,1

OK
Sending: AT^NDISDUP=1,1

OK
Sending: AT^NDISSTATQRY?

^NDISSTATQRY: 2,0,,"IPV4",2,0,,"IPV6"

OK
Internet Systems Consortium DHCP Client 4.3.5
Copyright 2004-2016 Internet Systems Consortium.
All rights reserved.
For info, please visit https://www.isc.org/software/dhcp/

Listening on LPF/wwan0/7a:25:f0:08:19:51
Sending on   LPF/wwan0/7a:25:f0:08:19:51
Sending on   Socket/fallback
DHCPDISCOVER on wwan0 to 255.255.255.255 port 67 interval 4
DHCPREQUEST of 10.145.49.212 on wwan0 to 255.255.255.255 port 67
DHCPOFFER of 10.145.49.212 from 10.145.49.209
DHCPACK of 10.145.49.212 from 10.145.49.209
bound to 10.145.49.212 -- renewal in 3265 seconds.
```

The line with NDISSTATQRY and a number 1 or 2 indicates that the connection is successfully established.

From now on, the mobile connection will be automatically re-established after a reboot of the CCUBE.

4.7 VPN Client Configuration

The CCUBE comes with build in client support for OpenVPN. To establish a connection to an existing OpenVPN server, a configuration file is required, which must be provided by the administrator of the OpenVPN server. The setup of an OpenVPN server is beyond the scope of the manual.

The configuration file has typically the file extension ".ovpn" and contains configuration, keys and certificates in the following format (simple text file):

```
client
proto udp
remote openvpnserver.example.com
port 1194
dev tun
nobind

key-direction 1

<ca>
-----BEGIN CERTIFICATE-----
# insert base64 blob from ca.crt
-----END CERTIFICATE-----
</ca>

<cert>
-----BEGIN CERTIFICATE-----
# insert base64 blob from client1.crt
-----END CERTIFICATE-----
</cert>

<key>
-----BEGIN PRIVATE KEY-----
# insert base64 blob from client1.key
-----END PRIVATE KEY-----
</key>

<tls-auth>
-----BEGIN OpenVPN Static key V1-----
# insert ta.key
-----END OpenVPN Static key V1-----
</tls-auth>
```

Please note: For an automatic start of the VPN service, the key file must be created without a password; otherwise the password must be given every time the VPN service is starts.

The configuration file must be copied from the host computer to the CCUBE using SCP/SFTP. From a Linux based host computer this can be done using the following command line (see 4.2 for more details):

```
scp client.ovpn ccube-admin@192.168.1.25:/home/ccube-admin/client.ovpn
```

On the CCUBE itself, the file must be copied to the /etc/openvpn directory which requires root privileges (see 4.2 for more details), also the file extension must be renamed to ".conf".' The following command copies and renames the configuration file uploaded by the previous command.

```
root@CC0XX:~# cp /home/ccube-admin/client.ovpn /etc/openvpn/client.conf
```

Finally the configuration must be enabled. The **ccube-network** command provides an easy way to enable/disable configurations located in /etc/openvpn directory.

```
root@CC0XX:~# ccube-network openvpn enable -h
usage: ccube-network openvpn enable [-h] cfg

positional arguments:
  cfg         Config file name from /etc/openvpn/ (e.g client.conf)

optional arguments:
  -h, --help  show this help message and exit
```

If the configuration file name is client.conf then the activation is as follows:

```
root@CC0XX:~# ccube-network openvpn enable client.conf
Enable VPN service...
Created symlink /etc/systemd/system/multi-user.target.wants/openvpn@client.ser-
vice → /lib/systemd/system/openvpn@.service.

root@CC0XX:~# systemctl start openvpn@client
```

After a reboot of the CCUBE the OpenVPN connection will be started automatically as soon as an Internet connection becomes available.

4.8 DATA-CUBE³ Settings

The CCUBE must be configured to *know* the correct settings of the connected DATA-CUBE³ and the corresponding settings for the CUBE SeedLink plugin. This is accomplished by the **ccube-datalogger** command:

```
root@CC0XX:~# ccube-datalogger -h
usage: ccube-datalogger [-h] [-v] [-S] <component> ...

CCUBE data logger management

optional arguments:
  -h, --help     show this help message and exit
  -v, --verbose  increase output verbosity
  -S, --syslog   Output to syslog only

Data logger subsystems:
  <component>
    dcube        Data-Cube management
```

The configuration must be carried out for the dcube and the logger modules:

```
root@CC0XX:~# ccube-datalogger dcube -h
usage: ccube-datalogger dcube [-h] <command> ...

optional arguments:
  -h, --help  show this help message and exit

Subcommands:
  <command>
    logger    Configure DCUBE parameter
    plugin    Configure Seedlink Cube-Plugin

root@CC0XX:~# ccube-datalogger dcube logger -h
usage: ccube-datalogger dcube logger [-h] [--channels CHANNELS]
                                     [--samples SAMPLES] [--delay DELAY]

optional arguments:
  -h, --help           show this help message and exit
  --channels CHANNELS  Number of active channels (default: 3)
  --samples SAMPLES    Sample rate (default: 100 sps)
  --delay DELAY        Filter delay (default: 31)

root@CC0XX:~# ccube-datalogger dcube plugin -h
usage: ccube-datalogger dcube plugin [-h] [--network NETWORK]
                                     [--location LOCATION] [--stream STREAM]
                                     [--channels CHANNELS]
                                     station

positional arguments:
  station              Station name (e.g. DCUBE ID)

optional arguments:
  -h, --help           show this help message and exit
  --network NETWORK    Network ID (default: UP)
  --location LOCATION  Location ID (default: 00)
  --stream STREAM      Stream ID (default: HH)
  --channels CHANNELS  Channel mapping (default: Z,N,E)
```

The following is an example configuration of the DATA-CUBE³ parameters and a subsequent restart of the SeedLink server running on the CCUBE:

```
root@CC0XX:~# ccube-datalogger dcube logger --channels 3 --samples 100
Configure DCUBE parameter...
Number of channels: 3
Sample rate:        100
Filter delay:       31
Wrote backup of configuration file to: /var/lib/ccube/cube.txt_201801232324

Wrote new configuration to /etc/seedlink/cube.txt
Restart seedlink server to active changes

root@CC0XX:~# systemctl restart seedlink-server
root@CC0XX:~#
```

In this example the SeedLink server will capture 3 channels with 100 sps and a filter delay of 31 and convert the raw data from the DATA-CUBE³ in near real-time to miniSEED.

Please refer to subchapter 3.1.1 for additional information on the filter delay parameter.

The following example is used to configure the CUBE SeedLink plugin and restart the SeedLink server:

```
root@CC0XX:~# ccube-datalogger dcube plugin --network UP --location 00 --stream HH
--channels Z,N,E A95

Configure Seedlink Cube-Plugin...
Network:  UP
Station:  A95
Location: 00
Stream:   HH
Channels: Z,N,E
Wrote backup of configuration file to: /var/lib/ccube/plugins.ini_201801232330
Wrote new plugin configuration to /etc/seedlink/plugins.ini
Wrote backup of configuration file to: /var/lib/ccube/seedlink.cfg_201801232330
Wrote new server configuration to /etc/seedlink/seedlink.cfg
Restart seedlink server to active changes
root@CC0XX:~# systemctl restart seedlink-server
```

The SeedLink server can be tested with the software tools given in subsection 2.3.1.

4.9 DATA-CUBE³ Memory Access

The DATA-CUBE³ is limited by the size of its internal memory. The 32GB are typically enough to record about 280 days continuously 3 channels at 100sps. Afterwards, the internal memory will be full and the DATA-CUBE³ stops recording.

The CCUBE can be used to cleanup the internal memory of the DATA-CUBE³. This is done by shortly switching to USB “config mode”, accessing the internal memory and removing old measurement files, and then switching back to “measurement mode”.

The current mode of the DATA-CUBE³ connection can be check in the following way:

```
root@CC0XX:~# ccube-hardware dcube status
Query DCUBE interface status...
USB 5V state:     off
USB switch state: stream
```

In this case, the DATA-CUBE³ is in streaming (measurement) mode.

The mode switching is done by the **ccube-hardware** command:

```
root@CC0XX:~# ccube-hardware dcube switch -h
usage: ccube-hardware dcube switch [-h] <command> ...

optional arguments:
  -h, --help   show this help message and exit

Subcommands:
  <command>
    config     Switches DCUBE into configuration/data access mode
    measurement
               Switches DCUBE into measurement mode
```

The following command is used to switch the DATA-CUBE³ to configuration mode and access the internal memory where the measurement data is saved:

```
root@CC0XX:~# ccube-hardware dcube switch config
Switch DCUBE into configuration/data access mode...
Enable 5V for USB-Host access
Switch USB port to DCUBE
Waiting for DCUBE SD-Card...
```

Now the file system of the DATA-CUBE³ can be mounted and files can be removed.

The following command switches the DATA-CUBE³ back to measurement mode and reactivates the near real-time data streaming:

```
root@CC0XX:~# ccube-hardware dcube switch measurement
Switches DCUBE into measurement mode...
Disable 5V for USB-Host access
```

4.10 Temperature & Voltage

The **ccube-sensors** command is used to retrieve sensor values from the CCUBE:

```
root@CC0XX:~# ccube-sensors -h
usage: ccube-sensors [-h] [-v] [-S] <component> ...

CCUBE sensors management

optional arguments:
  -h, --help     show this help message and exit
  -v, --verbose  Increase output verbosity
  -S, --syslog   Output to syslog only

Sensor subsystems:
  <component>
    voltage      Voltage sensors
    temperature  Temperature sensors
```

The following examples request all voltages and the temperature:

```
root@CC0XX:~# ccube-sensors voltage all
Reading all sensors...

# Sensor       Value
========================
  System       12.54 V
  Battery       3.29 V

root@CC0XX:~# ccube-sensors temperature board
Reading temperature sensors...

# Sensor       Value
================================
  Temperature  30.12 °C
  Alert Status -
```

The battery voltage shows the value of the built-in battery used to hold the EEPROM/BIOS information.

4.11 Manual CUBE Plugin Configuration

The CUBE plugin configuration file contains mainly information about the mapping between DATA-CUBE³ channels and SeedLink streams. A template for the plugin configuration is installed with the Seedlink CUBE-Plugin and can be found at:

```
/etc/seedlink/plugins.ini-cube
```

The file contains a section for the SeedLink CUBE plugin instance:

```
[cube01]
mode=1
gps_min_satellites=3
gps_dump_interval=1
gps_init_timestamps=3
max_jitter=-1
network="yy"
station="CC01"
location="00"
stream="HH"
channels=Z,N,E
```

A description of each parameter can be found in the following table:

Parameter	Description
mode	Push mode [0 = STDOUT, 1 = SEEDLINK]
gps_min_satellites	Minimum number of GPS satellite
gps_dump_interval	Interval in seconds to write time/position log files (if enabled)
gps_init_timestamps	Number of GPS timestamps to skip at start up
max_jitter	Max jitter in microseconds between stream and GPS (-1 for no checks)
network	SeedLink network code
station	SeedLink station code
location	SeedLink location code
stream	SeedLink stream code
channels	SeedLink channel mapping

SeedLink must be restarted after a manual modification of the configuration file:

```
root@CC0XX:~# systemctl restart seedlink-server
```

4.12 SeedLink Disk Buffer

Embedded computers like the CCUBE are using SD cards as main storage. Typically, SD cards can only handle a limited number of read/write cycles. Therefore it is recommended to use a Linux ramdisk to store the SeedLink disk buffer. Initially a 100MB ramdisk is created by the following line in the file `/etc/fstab`:

```
tmpfs /var/cache/seedlink tmpfs defaults,size=100M 0 0
```

The following command shows the file disk space usage and displays also the ramdisk:

```
root@CC0XX:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root       1.7G  615M 1006M  38% /
devtmpfs        124M     0  124M   0% /dev
tmpfs           124M     0  124M   0% /dev/shm
tmpfs           124M  5.0M  119M   4% /run
tmpfs           5.0M     0  5.0M   0% /run/lock
tmpfs           124M     0  124M   0% /sys/fs/cgroup
tmpfs           100M     0  100M   0% /var/cache/seedlink
/dev/mmcblk0p3  5.4G   23M  5.1G   1% /media/data
```

Note that SeedLink will use this buffer only after configuring it in `/etc/seedlink/seedling.cfg` and restarting the SeedLink server. It is set by default.

Please note: If the disk buffer is located on a ramdisk (default), all data which is not already send, will be lost during a power cycle!

5 Pin-out

Please note: The following pin-out tables are describing the data sockets of the CCUBE!

10-way socket (male): DATA OUT

Pin	A	B	C	D	E	F	G	H	J	K
Ethernet							TX+	TX-	RX+	RX-
Terminal (TTL)	GND				TX	RX				
USB 2.0	GND	5V	D+	D-						

7-way socket (female): CUBE (to DATA-CUBE)

Pin	A	B	C	D	E	F	G
Signal	USB + 5V	USB + D	USB – D	GND	Terminal Tx 3.3V	Terminal Rx 3.3V	NC

4-way socket (male): External Power

Pin	A	B	C	D
Signal	Power 10.5-24V DC	NC	NC	GND

Please note that the corresponding connectors (plugs) must have the opposite gender.

6 LEDs

The LEDs indicate the status of the CCUBE:

LED	Status	Interpretation
ETH	On	Ethernet connected
ETH	Blinking	Ethernet with active data transfer
LOAD	2x blink, off, 2x blink, ...	OK – Heartbeat is active
UMTS	On	UMTS hardware is on
UMTS	Off	UMTS hardware is off
WLAN	On	WiFi hardware is on
WLAN	Off	WiFi hardware is off
DATA	Slow blinking	OK – Receiving valid data stream from DATA-CUBE³

7 General Usage Information

7.1 Safety Instructions

Improper use voids the warranty and can damage or even burn the electronics of the CCUBE! Besides general recommendations of electronics usage, the user must pay attention to the following, additional usage information:

All pins are not protected against overvoltage!

Do not connect RS-232 of the CCUBE directly to a computer!
The CCUBE uses low voltage RS-232.

Do not open the CCUBE! Doing so voids the warranty and can damage internal parts!

Make sure that the SIM cap is tight before deploying the CCUBE in the field. It is recommended to wrap the CCUBE in a plastic bag for long-term outdoor installation. The CCUBE housing provides a short-term protection against moisture & rain, but is not designed to stay permanently underwater.

The program “ccube-hardware” shall only be used to switch the operation mode of the DATA-CUBE³! Wrong usage can lead to an unusable CCUBE.

7.2 Additional Information

All CCUBE units are thoroughly tested by DiGOS prior to shipment. The units have proven reliable and trouble-free operation in deployments. However, DiGOS excludes any liability for data loss due to hardware malfunction or operation errors.

Please note that data transmission via LTE/UMTS, satellite connections, etc. might produce additional costs. DiGOS is in no way responsible or liable for the amount of the transmitted traffic or for the costs resulting from additional data traffic, configuration, updates, etc.

8 Technical Specification

System performance

- System architecture: ARM9
- Operating frequency: 400MHz
- System RAM: 256MB
- Operating system: Embedded Debian Linux
- Recovery: 100MB (buffering up to 1 day @ 100sps, configurable)

Data streaming

- Format: miniSEED via SeedLink server & CUBE plugin
- Live streaming: Yes
- Data buffering: Yes, default 1 day @ 100sps in case of communication outage (STEIM2 compressed)

Time synchronization

- NTP: NTP synchronization via Internet

Communication

- WiFi: 802.11bgn
- WiFi configurations: Client in managed network, ad-hoc or meshing
- WiFi meshing: Supported via B.A.T.M.A.N (see http://open-mesh.org)
- WiFi meshing distance: 800m between CCUBEs with free line-of-sight & omni-directional antennas at 2m height
- WiFi meshing bandwidth: 4MBit/s for 800m distance between CCUBEs (see above)
- Mobile broadband (cellular network): 4G/LTE, 3G/UMTS and 2.5G/EDGE (securely tightened SIM card slot with screwed and sealed cap)
- Ethernet: 100MBit/s
- VSAT & BGAN: Supported via Ethernet interfaces

Remote operation

- VPN access: Yes (OpenVPN)
- Remote login: Yes (SSH)
- Monitoring: Yes (system voltage, temperature, communication status & performance)

Local operation

- RS-232 / USB: Terminal access
- LEDs indicating status of
    - Ethernet connection
    - System load
    - 4G/LTE
    - WiFi
    - Seismic data acquisition

Connectors

- Data out: MIL-C-2684 12-10P (Ethernet, USB, RS-232)
- WiFi antenna: RP-SMA (female)
- Mobile broadband antenna: FME (female)
- DATA-CUBE³ input: MIL-C-2684 10-07S
- Power: MIL-C-2684 08-04P

Power supply

- Input voltage: 10.5-24V DC
- Battery: External battery or power supply required
- Power consumption:
  - Idle: 360mW
  - Streaming via Ethernet: 700mW
  - Streaming via WLAN: 820mW
  - Streaming via 4G/LTE (typical): <1W
    Power consumption rated for CCUBE only. DATA-CUBE³ must be added (~300mW running in continuous GPS acquisition mode).

Physical

- Size: 100 x 100 x 83mm (830ml)
- Weight: 730g
- Operating outdoor temperature: -10 - 60°C (other temperature versions on request)
- Housing: Reinforced plastic
- Waterproof: IP65