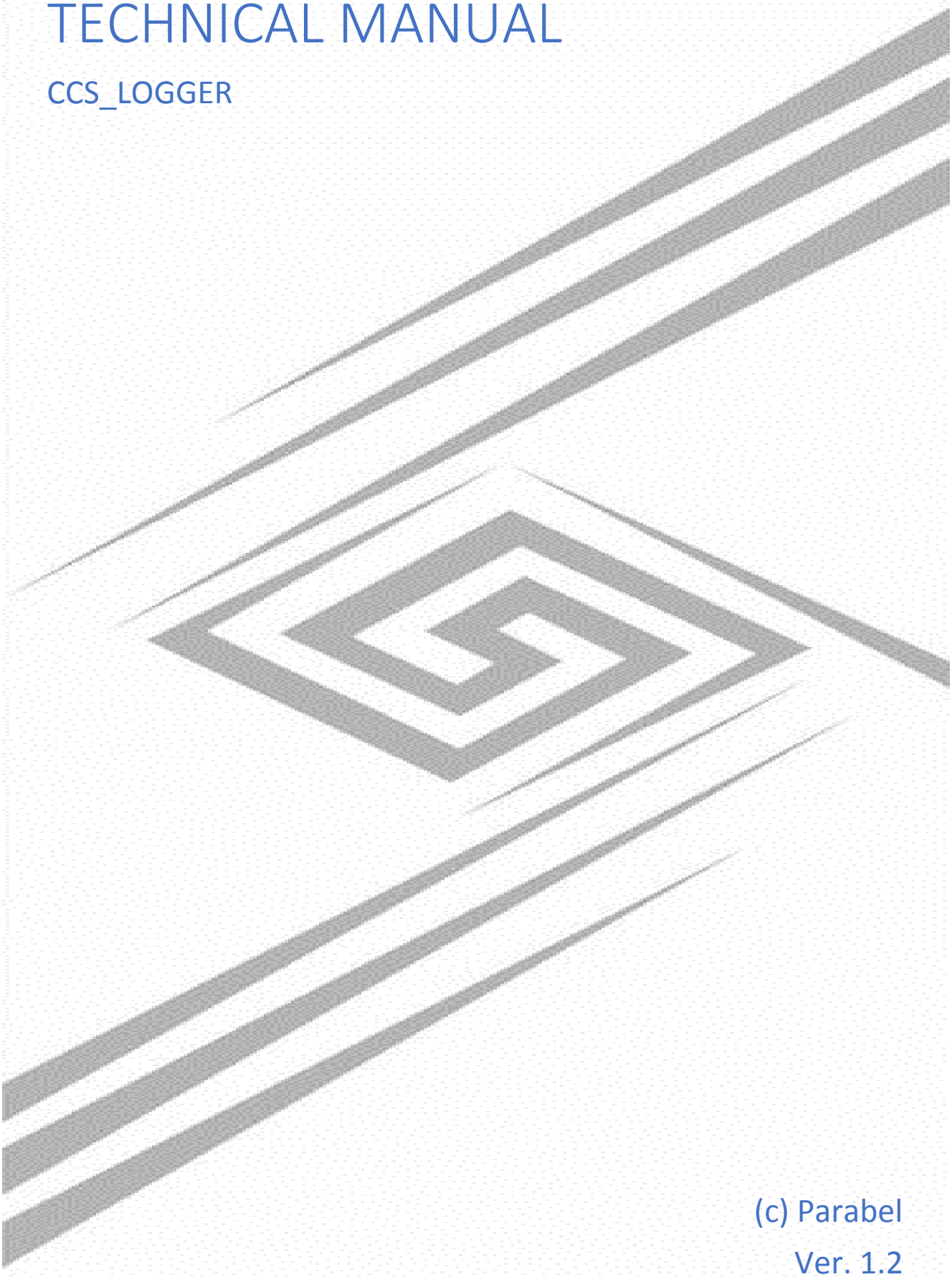
TECHNICAL MANUAL

CCS_LOGGER



(c) Parabel

Ver. 1.2

TABLE OF CONTENTS

1.INTRODUCTION

Purpose of the program ccs_logger

The product is designed to work as part of a hardware and software complex that records information from E1 lines, parsing SS7 and PRI common channel signaling protocols, archiving media messages.

Applications can be:

- diagnostics of communication networks, identification and localization of faults in signaling, problems with setting up telecommunication equipment;

- collection and analysis of connection statistics, identification of loaded lines;

The message base recorded using ccs_logger can be used in conjunction with AI programs to provide intelligent communication services, for example:

- analysis of the quality of work of operators in call centers

- analysis and identification of conflict situations

- search for dialogs by keywords

Output information

As a result of the ccs_logger software, the following files are created:

- SQL database with committed connection information;

- files with audio data in wav format;

- log file with decryption of the exchange protocol;

Output file formats are described on page 16.

Prerequisites for installation

- computer with Quasar adapter installed

- deployed Linux x64 OS distribution

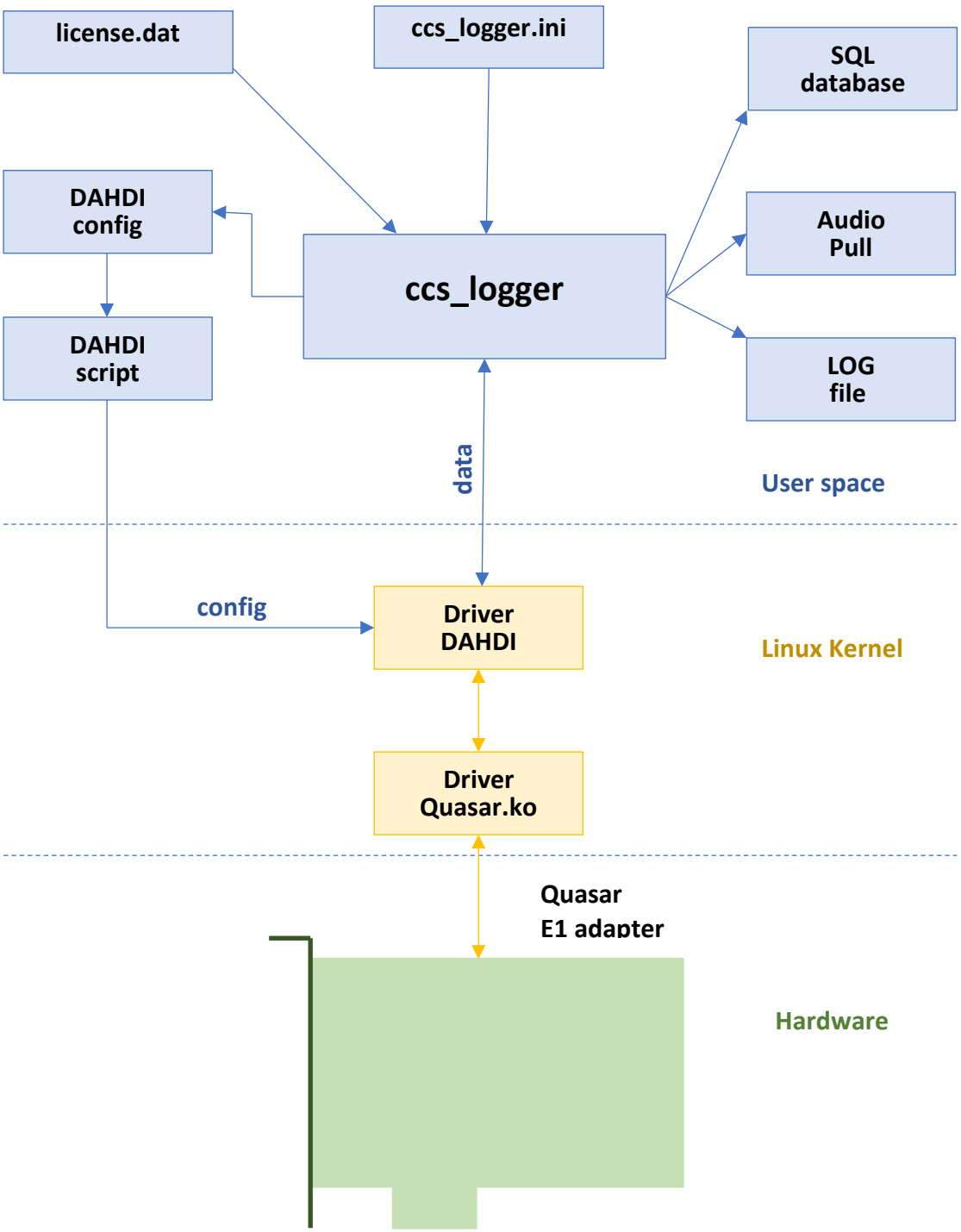
- DAHDI package with Quasar driver

- ccs_logger license.

- standard Linux compilation tools and header files

The software installation process is described on page 4. The process of obtaining a license is described on page 7. Connection of Quasar adapter to E1 lines is described on page 11.

2. SOFTWARE STRUCTURE



Выходными файлами для **ccs_logger** являются база данных (**SQL database**), директория с аудио данными (**Audio pull**), файл с историей соединений (**LOG file**). Конфигурация программы задается в файле **ccs_logger.ini**. На основе этой конфигурации автоматически создается конфигурация драйвера **DAHDI config**, которая скриптом **DAHDI script** загружается при старте программы.

3. INSTALLING THE SOFTWARE

Step by step, to install the ccs_logger software, you must do the following:

1. Install DAHDI package with quasar driver. The finished assembly is located on the manufacturer's website at the link http://parabel.ru/download/. The package is a bz2 archive with source files that you need to unpack and run the **build.sh** script, which will compile the drivers and utilities. For successful compilation, you must first install the following components from the Linux repository:

gcc - a compiler of the same version as the OS kernel.

patch - utility

linux-headers-XXX - kernel headers

linux-kbuild-XXX - kernel build scripts

In addition, the **crc_ccitt** linux driver must be installed on the system. You can check its presence in the system with the commands

lsmod | grep crc

If the driver is not loaded, you need to check if it can be loaded:

modprobe crc_ccitt

Since the **ccs_logger** program works with an absolute link to the **DAHDI** package, there is no need to install the package on the system, the **install.sh** script should not be run. You should also not install **DAHDI** from the standard repository, so as not to get a version conflict.

2. Unpack the **ccs_logger** archive into the working directory. In the **dahdiStart.sh** file, edit the **DAHDI_ROOT** variable, which should point to the directory with the **DAHDI** package.

When launched, the **dahdiStart.sh** script should load DAHDI successfully, which can be checked with the command:

 # lsmod | grep dahdi

dahdi 249856 1 quasar

crc_ccitt 16384 1 dahdi

If this is not the case, several options are possible:

- no quasar adapter installed

- the path to the directory with the **DAHDI** package is incorrect

- **DAHDI** did not compile

- the **linux-headers** headers were taken from another kernel

3. Install the **ODBC** library from the Linux distribution

In Debian, for example:

apt-get install unixodbc unixodbc-dev

ODBC is an interface for accessing networked or file-based SQL. The appropriate database driver must be installed.

For **sqlite3** database, install the following packages:

apt-get install sqlite3 libsqliteodbc

For postgre database:

apt-get install odbc-postgresql

After installing the database drivers, the file **/etc/odbcinst.ini** will contain the actual links to the libraries on your system. Copy the odbcinst.ini file to the **ccs_logger** directory.

Edit the local **odbc.ini** file according to the **odbcinst** content and the attributes of the existing database:

Driver - specify driver name from odbcinst file

Database - specify the path to the database (for sqlite) or the name of the network DB

UserName – specify the username of the network database

Password – password to access the database

Port - tcp port of network db

Servername – the IP address or network name of the database server

Access to most popular databases can be configured in the same way - Microsoft Access, MySQL, Oracle, Microsoft SQL Server etc.

Note that the **ccs_logger** program takes the current **odbc.ini** and **odbcinst.ini** files from the local directory where the program was launched. The system-wide files **/etc/odbc.ini** and **/etc/odbcinst.ini** are ignored.

4. If necessary, you can additionally configure **ccs_logger** by changing the default paths in the **ini** file: to the **dahdi.conf** configuration file; to the dahdi startup script; name of the connection database; path to the directory with audio files.

5. If there is a **license.dat** file, place it in the **ccs_logger** working directory, otherwise obtain a license following the procedure described on page 7.

6. According to the connection of the E1 lines, configure the linkset sections in the ini file as described on page 9.

7. Run the program for execution:

#./ccs_logger -s

4. STARTING THE PROGRAM, COMMAND LINE

Running **ccs_logger** without parameters displays a brief help, the program exits:

ccs_logger [options]

-l {file} generate file with license request, exit

-s start recordering

-v print version (build number)

-h help

Launching with the -l switch creates a license request file, the program exits. The licensing features are described in the corresponding paragraph.

Running with the -v switch (without parameters) displays the program version (build number).

Launching with the -s switch (without parameters) puts the program into the main operating mode, recording information and saving it to files is carried out in accordance with the **ccs_logger.ini** configuration.

You can end the program by pressing **Ctrl-C**.

Pressing any key leads to the output of work statistics:

Linkset 0

Counters:

Setup 5, Progress 5, Answered 4, Release 5, Release complete 5

Errors:

Format 0, Proto 1, Fops 0, Queue 0

The statistics counters record the number of events since the start of the program:

Setup – protocol packet Setup (Q.931) or Initial address (SS7)

Progress – protocol package Call proceed (Q.931) or Call progress (SS7)

Answered – protocol packet Connect ack (Q.931) or Answer (SS7)

Release – Release protocol packet (Q.931 or SS7)

Release complete – Release complete protocol packet (Q.931 or SS7)

Error counters:

Format – format error, such as an out-of-bounds parameter or unexpected end of packet

Proto – protocol error, such as an unsupported parameter or packet

Fops – file I / O error

Queue – an internal queue overflow is likely to result in packet loss

5. LICENSE AND LICENSE REQUEST PROCEDURE

The license entitles you to use one copy of the ccs_logger program in a fixed hardware environment. By installation we mean one or more E1 adapters of the Quasar series installed in one server, the ccs_logger program and the license for it.

The license is issued by the manufacturer for a specific installation of the program that works with one or more registered Quasar adapters..

Moreover, the following is true:

- the license is issued for ccs_logger to work with specific instances of Quasar adapters, with specific serial numbers.

- it is not allowed to add unregistered Quasar adapters to the installation, in this case the license must be obtained again by the request procedure.

- under the current license, it is allowed to remove one (several) adapters from the installation. At least one registered Quasar must remain on the system.

- it is allowed to transfer the installation from one server to another under one license.

License request procedure

The manufacturer can supply the ccs_logger software either with a ready-made installation or separately. In the latter case, the license must be obtained by sending a letter with a request to the manufacturer's technical support address. The letter must contain a binary license request file that is generated by the ccs_logger program when launched with the -l switch:

ccs_logger -l {file}

where **file** is the name of the output file, which can be arbitrary.

for example,

ccs_logger -l a.dat

will write to the current directory a binary file a.dat with a license request.

The program may fail:

No quasar adapter detected or no driver

This will mean that the program did not find any Quasar adapter installed in the server. The reason may be not only the physical absence of adapters, but also the absence of a loaded driver for it. In this case, you need to install the entire software package and start ccs_logger again.

The file with the license license.dat is sent by the manufacturer and must be copied to the working directory, from where the ccs_logger program itself is launched. When launched, the program will check if the installation is compliant with the license and will allow further operation.

6. CONFIGURATION

6.1. INI FILE

The configuration is described by the user in the **ccs_logger.ini** file in the format of a standard **ini** file and is reduced to setting values for variables in the form:

Parameter=value

where **Parameter** – the name of variable, **value** - the value of variable.

The content of the ini file is split into sections:

[common] – paths to working files and directories

[log] – control output to log file

[linkset N] - description of group E1 with number N

The configuration is read from the ini file once at the start of the program. To apply the configuration changes, you need to restart **ccs_logger**.

Comments can be added on each line, after the symbol «;».

6.2. COMMON SECTION

The section describes the paths to working files and directories.

dahdiConf = xxx - the name of the file with the configuration of the dahdi driver. The name can contain a path. The file is created automatically at the start of the program and then used by the script specified in the **dahdiScript** parameter.

DahdiScript = xxx – the name of the script that reloads the dahdi driver and configures it. The script is created by the user and must have execute permission. You can use the dahdiStart.sh script included with the program.

audioPath=xxx – path to the directory with audio files. The directory must be user created and have write access. The directory path can be either absolute or relative.

saveAudio=1 - record audio files, = 0 - do not record

saveMeta = 1 - write metafiles, = 0 - do not write

startRecord = setup – start recording an audio file immediately after the start of the connection

startRecord = answer – start recording an audio file after receiving a response from the subscriber

6.3. LOG SECTION

timestamp=1 - print a timestamp with each message to the log. The label will look like

2020-12-03 17:59:43.22>

timestamp=0 – time stamp is not printed.

info=1 – print info messages

warn=1 print warnings

lev2=1 print messages of the second level of the protocol. In case of PRI it is Q.921, in case of SS7 protocol it is MTP2.

lev3=1 print protocol layer 3 messages.

If info/warn/lev2/lev3 = 0, then the corresponding message type is not printed in the log.

console=1 - a copy of messages in the log file will be displayed in the console.

filename=xxx – the name of the log file. If the file does not exist, it will be created. Existing file will be appended from the end.

6.4. LINKSET SECTION

The **linkset** section describes a group of target E1 lines monitored by Quasar equipment. A group can include one or several (up to 4) E1 lines controlled by a common signaling channel. The target E1 lines are connected to the ports of the Quasar adapter.

The section header must contain the group number separated by a space, for example

[linkset 3]

Several groups can be described, the maximum number is determined by the license.

This section describes the following parameters

numslots = xxx – the number of timeslots in the group.

> The parameter can take fixed values - 32,64,96,128, which corresponds to 1 to 4 lines connected to the monitoring, which, in turn, corresponds to 2 to 8 ports on the Quasar adapter (taking into account two directions - rx, tx). For more connection diagram, see page 11.

dslot0 = xxx – timeslot number with the main signaling channel.

> Time slots are numbered starting from zero, each E1 corresponds to 32 timeslots. Timeslots 0,32,64 and 96 correspond to the E1 markup channel containing the label of the beginning of the frame. End-to-end numbering from the zero timeslot of the first E1 line in the group. For example, if the signaling is on the 17th time slot of the third line E1, then you need to specify dslot0 = 81.

dslot1 = xxx – timeslot number with backup signaling channel.

> The numbering rules are the same as for dslot0. This parameter is optional - if there is no backup channel, the dslot1 line can be commented out.

proto = xxx –signaling protocol

> There may be two options **pri** or **ss7**.

law = xxx

> It can take the values **a** or **m**. The parameter is used only for configuring the DAHDI driver.

ifid0=xxx

ifid1=xxx

ifid2=xxx

ifid3=xxx – the number of the E1 interface in the group, the parameter refers **only** to the PRI protocol. Ifid0 corresponds to the lowest E1 port in the group, ifid3 - to the highest port in the group.

The parameter value corresponds to the interface number in the Q.931 protocol (PRI). The interface number is assigned administratively when configuring the external PBX and can take the values 0..127. On single E1 lines, the indication of the "default" interface is most often used, in this case the ifid parameters should not be specified. If a group contains two E1s on a common signaling channel, only the parameters ifd0, ifd1 are indicated, the rest are ignored. If the group contains three E1s, you need to specify ifid0, ifid1, ifid2. Accordingly, for a group of 4 E1s, you need to specify ifid0..ifid3.

6.5. DB SECTION

The **db** section contains parameters:

dbName = xxx

where xxx is the name of the database specified in the odbc.ini file

saveDb = 1 – save information to the database, =0 – don’t save

If saveDb = 1 and an error occurred while writing to the database, ccs_logger exits.

7. CONNECTING THE E1 PORTS OF THE ADAPTER TO E1 LINES

For the ccs_logger program to work correctly, the ports of the Quasar adapters must be connected to the E1 lines in compliance with certain rules. In this section, we will consider several schemes:

- connection to one line E1

- connection to two E1 lines with separate signaling channels

- connection to two E1 lines with a common signaling channel

- connection to four E1 lines with a common signaling channel and a backup channel

Under the **E1 line**, we will further understand the line connecting the PBX1 and PBX2 switching equipment, with which it is necessary to record information. Physically, the line consists of two twisted pairs, transmitting information in two directions. The direction PBX1 -> PBX2 is denoted as TX, the direction of PBX2 -> PBX1 is denoted as RX. In the figures, one twisted pair will be denoted by one graphic line.

By **E1 port** we mean the port of the Quasar adapter installed in the system under the control of the ccs_logger program. The port has one transmitter and one receiver. To retrieve information from the E1 line, the port transmitter is not used, but two receivers are required for the RX, TX directions. Accordingly, one E1 line has two ports of the Quasar adapter.

The ini file of the ccs_logger program describes the configuration of the **E1 lines** (section **linkset**), it is assumed that the connection of these lines to the ports will follow the rules:

- if the **linkset** describes one line, the P port of the adapter corresponds to RX, the P + 1 port corresponds to TX;

- if the **linkset** describes n lines (with a common signal channel), 2 * n ports are used for connection, and the RX direction (RX1, RX2 ... RXn) is connected to the odd ports P1, P3 .. Pn, to the even ports P2, P4, Pn + 1 connect direction TX (TX1, TX2 .. TXn).

- higher numbered linkset matches higher numbered ports.

Information about the ports, as the program assumes to use them, is displayed at startup in the console and in the log file, for example:

LINKSET 0 (Rx) ->PORT 1

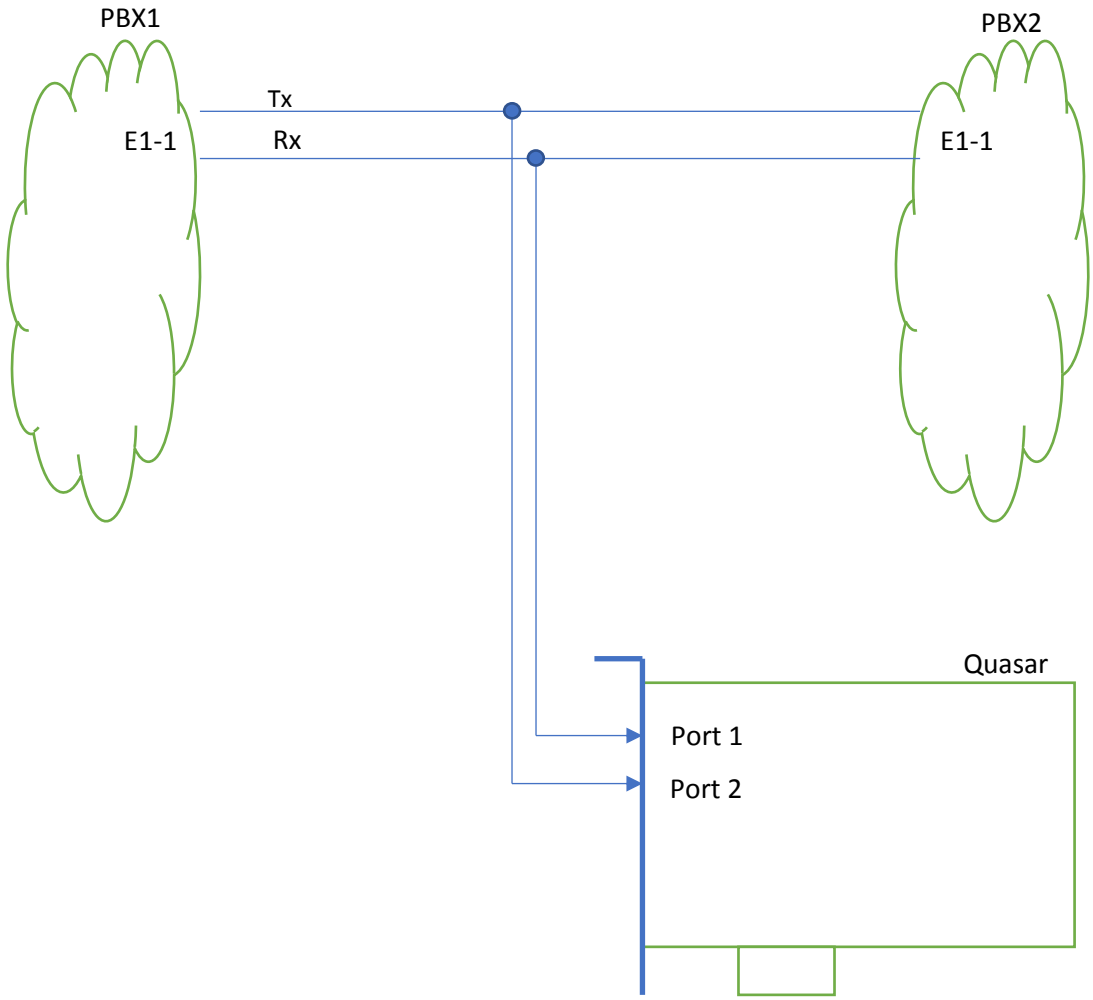
LINKSET 0 (Tx) ->PORT 2

LINKSET 0 (Rx) ->PORT 3

LINKSET 0 (Tx) ->PORT 4

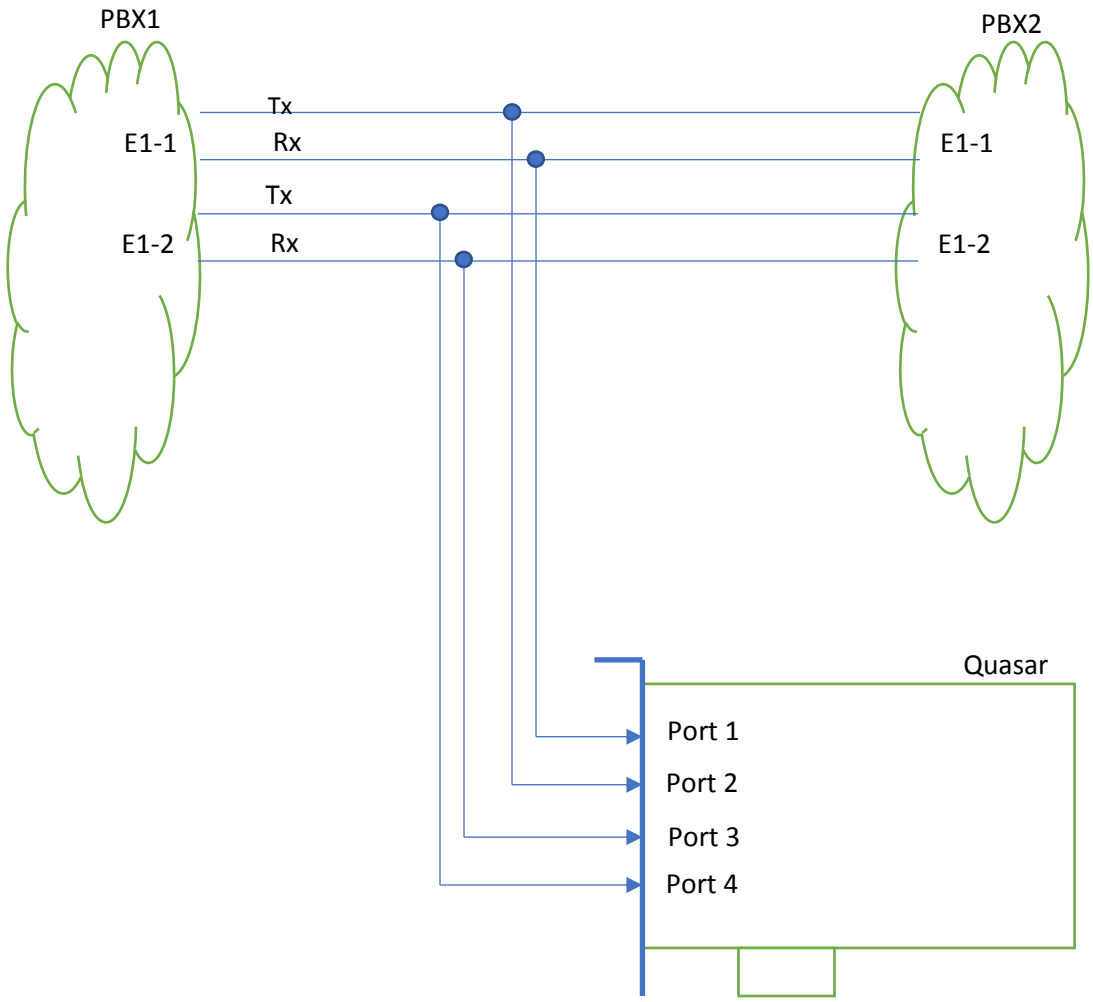
Next, let's move on to considering specific connection examples.

7.1. CONNECTING TO ONE E1 LINE




```
[linkset 0]

numslots = 32

dslot0 = 16

proto = ss7

law = a
```

7.2. CONNECTING TO TWO E1 LINES WITH SEPARATE ALARM CHANNELS

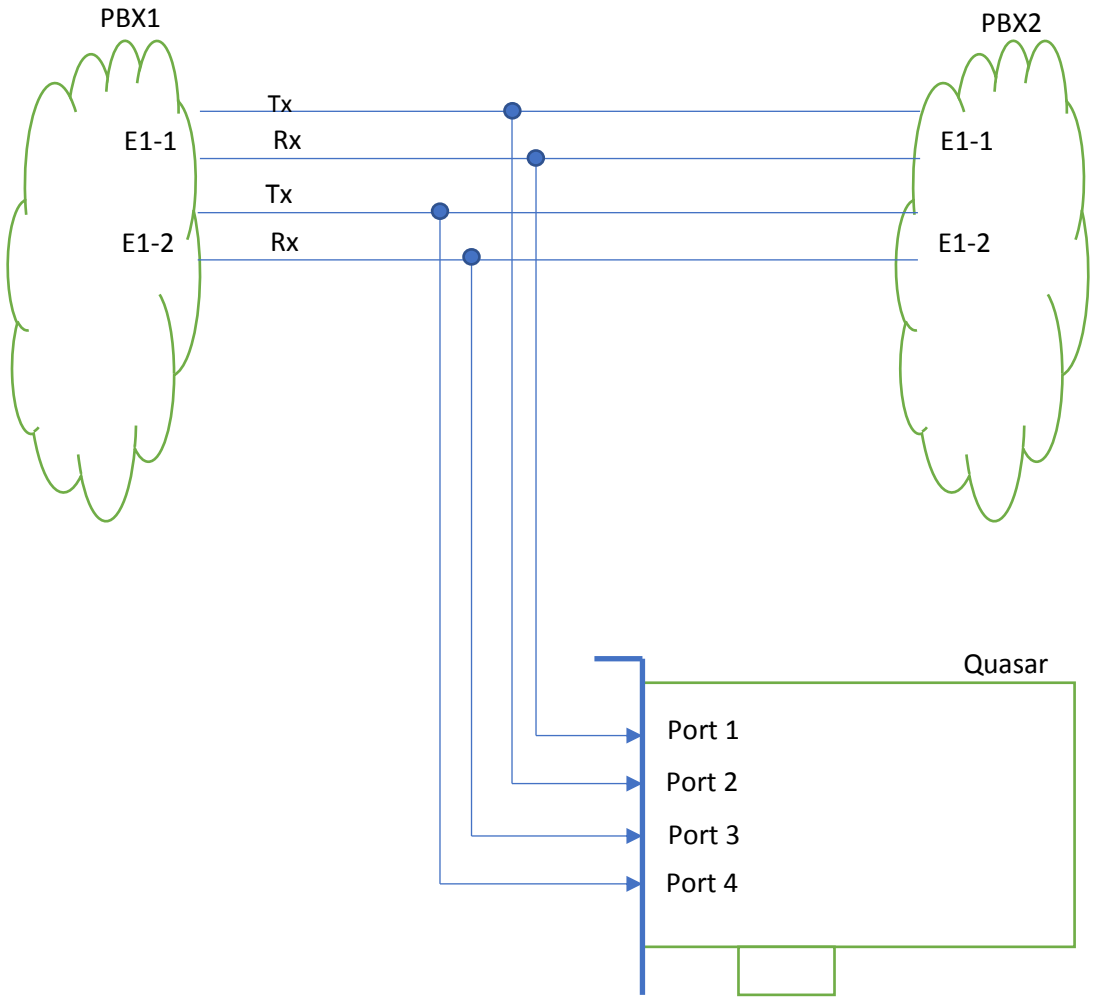


Since both E1 lines have their own signaling channel, two linkset sections need to be described. Scheme of connecting lines to ports RX1-TX1-RX2-TX2.

```
[linkset 0]

numslots = 32

dslot0 = 16

proto = ss7

law = a
```

```
[linkset 1]

numslots = 32

dslot0 = 16

proto = ss7

law = a
```

7.3. CONNECTING TO TWO E1 LINES WITH A COMMON SIGNALLING CHANNEL

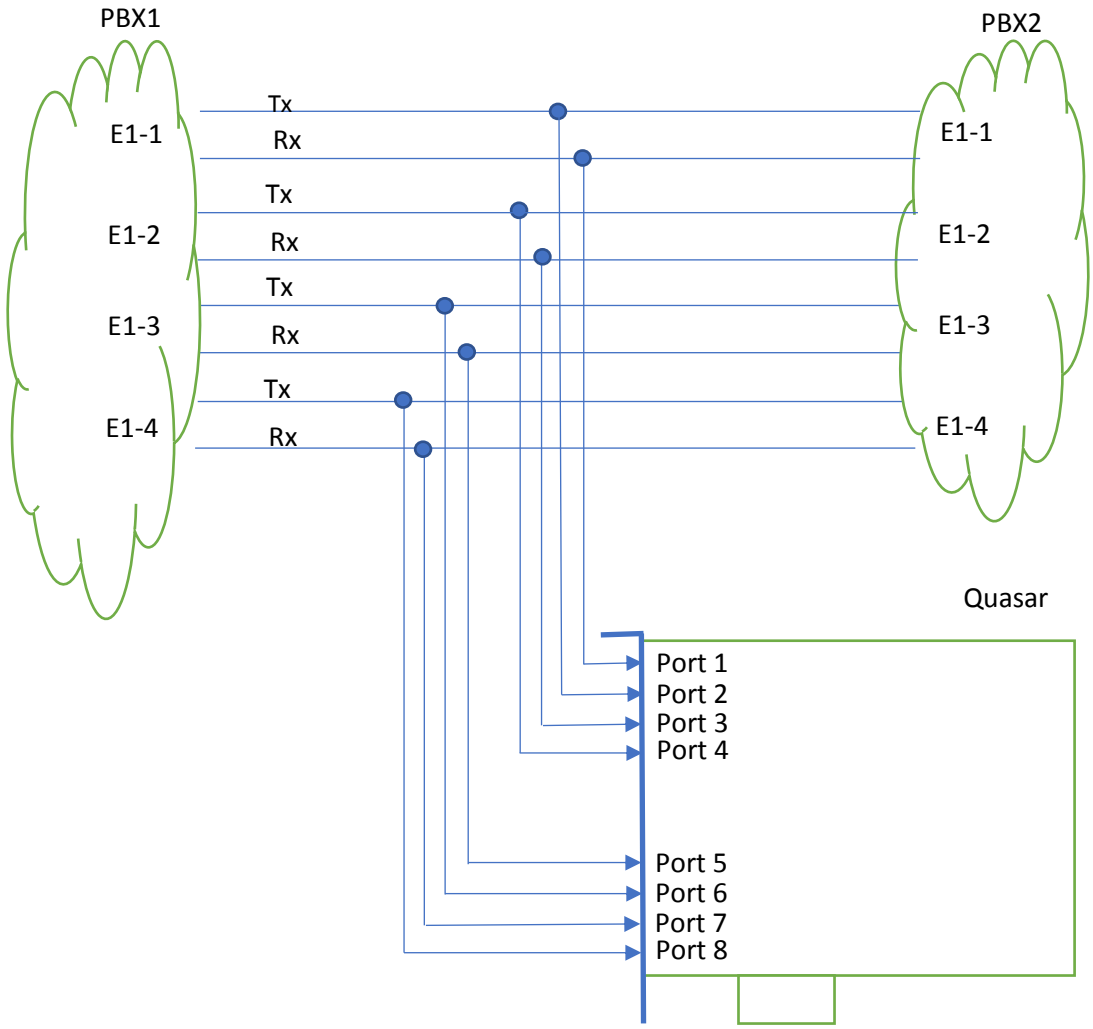


Both E1 lines share a common signaling channel (eg 16 on an E1-1 line), the group must be described in one linkset section. Scheme of connecting lines to ports RX1-TX1-RX2-TX2.

```
[linkset 0]

numslots = 64

dslot0 = 16

proto = pri

law = a

ifid0 = 2

ifid1 = 4
```

If PRI signaling is used, it may be necessary to specify an ifid for each line. The interface id is assigned by the PBX1 or PBX2 administrator.

7.4. CONNECTING TO FOUR E1 LINES WITH COMMON AND BACKUP SIGNALLING CHANNEL




```
[linkset 0]

numslots = 128

dslot0 = 16

dslot1 = 48

proto = ss7

law = a

ifid0 = 2

ifid1 = 4

ifid2 = 6

ifid3 = 7
```

The four lines share a common signaling channel (16) and a redundant signaling channel (48), the group must be described in one linkset section. Scheme of connecting lines to ports RX1-TX1-RX2-TX2-RX3-TX3-RX4-TX4.

If PRI signaling is used, it may be necessary to specify an ifid for each line. Interface id assigned by PBX1 or PBX2 administrator.

8. OUTPUT FILE FORMATS

8.1. META FILES

The meta file corresponds to one complete telephone call detected by the system. Each meta file has a unique digital call ID, which is also the name of the file. The meta file also has a paired audio file that contains the contents of the call. The names of the corresponding metafile and media are the same and are equivalent to the identifier. Meta files have a .meta extension, for example 16068878173880240011.meta.

Meta files contain the following fields:

ID 16068878173880240011 – unique call identifier

LINKSET 0 – E1 group number on which the call is recorded

SIG - the type of signaling

BCHAN – timeslot number

NUMCALLED – called party number

NUMCALLING – caller number

TIME – time and date of the start of the call, determined by the first signaling packet

DURATION - call duration (in seconds)

TSETUP – the absolute timestamp of the call start (seconds)

TPROGRESS – absolute timestamp of confirmation of the call by the opposite PBX (seconds)

TRELEASE - absolute timestamp of the end of the call (seconds)

TANSWER - absolute timestamp of the caller's answer (seconds)

CAUSE – call termination reason, in signaling protocol terminology

CAUSECODE – call termination reason code, in decimal

RELEASEDIR – who ended the call, “I” is the initiator of the call, “D” is the recipient of the call

CALLDIR - call direction, “>” - initiator on the RX line, “<” - initiator on the TX line

PROTO – specific fields for the corresponding protocol:

PRI.CRVALUE – call reference value

PRI.CODING – call coding, according to the protocol

SS7.SPC – source point code

SS7.DPC – destination point code

SS7.CIC – circuit identification code

Example meta file:

ID 16294360859244850001

LINKSET 0

SIG SS7

BCHAN 1

NUMCALLED 310

NUMCALLING 13202

TIME 2021-08-20 11:08:05.92

DURATION 0017.94

TSETUP 1629436085.92

TPROGRESS 1629436086.07

TRELEASE 1629436103.86

TANSWER 1629436091.08

CAUSE Normal call clearing

CAUSECODE 16

CALLDIR >

RELEASEDIR I

PROTO SS7.SPC=599/SS7.DPC=650/SS7.CIC=1

8.2. AUDIO FILES

Audio data is saved in wav files with the following characteristics:

- sampling rate 8 khz

- Number of channels - 2

- A-law encoding for SS7 protocol, for PRI is determined by the protocol.

The file name is equivalent to the call identifier and is unique.

The file is created by the system after receiving a connection establishment packet over the signaling channel. From this moment on, there is a continuous recording of audio data to a file from the timeslots specified in the protocol. The receiver side is recorded in one channel of the wav file, the transmitter side is recorded in the other channel. The recording ends after receiving a disconnect packet. Files are written to the directory specified in the audioPath parameter (see the description of the ini-file).

8.3. DATABASE

The information that is recorded in the metafiles is almost entirely duplicated in the SQL database. The name of the database is set in the dbName parameter (see the description of the ini file). The database is accessed through a unified

ODBC interface, which makes it possible to organize writing to most popular databases - SQLite, Postgre, Microsoft Access, MySQL, Oracle, Microsoft SQL Server, etc.

The database allows you to search and sort records by various characteristics - phone numbers, time and duration of the call.

8.4. LOG FILE

Protocol-level messages are displayed in the Log file, which allows you to directly observe the establishment of the connection. An example log file is given below:

```
2020-12-02 11:54:29.53>  Q931 MSG: SETUP (5), CALL REF: 201
2020-12-02 11:54:29.53>    Q931 IE: BEARER (4), length=3
2020-12-02 11:54:29.53>    Q931 IE: speech 64kbit/s, coding a-law
2020-12-02 11:54:29.53>    Q931 IE: CHANNEL ID (24), length=4
2020-12-02 11:54:29.53>    Q931 IE: Chan ID: ifid=2, timeslot=22
2020-12-02 11:54:29.53>    Q931 IE: ? (40), length=4
2020-12-02 11:54:29.53>    Q931 IE: CALLING PARTY (108), length=2
2020-12-02 11:54:29.53>    Q931 IE: Calling number:
2020-12-02 11:54:29.53>    Q931 IE: CALLED PARTY (112), length=3
2020-12-02 11:54:29.53>    Q931 IE: Called number: 99
2020-12-02 11:54:29.53>    Q931 IE: ? (161), length=-1
2020-12-02 11:54:29.54>  Q931 MSG: CALL PROCEEDING (2), CALL REF: 201
2020-12-02 11:54:29.54>    Q931 IE: CHANNEL ID (24), length=4
2020-12-02 11:54:29.54>    Q931 IE: Chan ID: ifid=2, timeslot=22
2020-12-02 11:54:29.57>  Q931 MSG: CONNECT (7), CALL REF: 201
2020-12-02 11:54:29.57>    Q931 IE: CHANNEL ID (24), length=4
2020-12-02 11:54:29.57>    Q931 IE: Chan ID: ifid=2, timeslot=22
2020-12-02 11:54:29.57>    Q931 IE: PROGRESS INDICATOR (30), length=2
2020-12-02 11:54:29.58>  Q931 MSG: CONNECT ACK (15), CALL REF: 201
2020-12-02 11:54:32.04>  Q931 MSG: DISCONNECT (69), CALL REF: 192
2020-12-02 11:54:32.04>    Q931 IE: CAUSE (8), length=2
2020-12-02 11:54:32.04>    Q931 IE: Cause: Normal call clearing
```

Depending on the settings in the ini-file, information of the 2nd or 3rd level of the protocol can be displayed in the log file, the file can be duplicated in the console.