sysmocom

sysmocom - s.f.m.c. GmbH

OsmoTRX User Manual

by Pau Espin Pedrol

The Asciidoc source code of this manual can be found at http://git.osmocom.org/osmo-gsm-manuals/

HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
1	March 6, 2019	Initial version.	PE

Contents

Foreword

Digital cellular networks based on the GSM specification were designed in the late 1980ies and first deployed in the early 1990ies in Europe. Over the last 25 years, hundreds of networks were established globally and billions of subscribers have joined the associated networks.

The technological foundation of GSM was based on multi-vendor interoperable standards, first created by government bodies within CEPT, then handed over to ETSI, and now in the hands of 3GPP. Nevertheless, for the first 17 years of GSM technology, the associated protocol stacks and network elements have only existed in proprietary *black-box* implementations and not as Free Software.

In 2008 Dieter Spaar and I started to experiment with inexpensive end-of-life surplus Siemens GSM BTSs. We learned about the A-bis protocol specifications, reviewed protocol traces and started to implement the BSC-side of the A-bis protocol as something originally called `bs11-abis`. All of this was *just for fun*, in order to learn more and to boldly go where no Free Software developer has gone before. The goal was to learn and to bring Free Software into a domain that despite its ubiquity, had not yet seen any Free / Open Source software implementations.

`bs11-abis` quickly turned into `bsc-hack`, then *OpenBSC* and its *OsmoNITB* variant: A minimal implementation of all the required functionality of an entire GSM network, exposing A-bis towards the BTS. The project attracted more interested developers, and surprisingly quickly also commercial interest, contribution and adoption. This allowed adding support for more BTS models.

After having implemented the network-side GSM protocol stack in 2008 and 2009, in 2010 the same group of people set out to create a telephone-side implementation of the GSM protocol stack. This established the creation of the Osmocom umbrella project, under which OpenBSC and the OsmocomBB projects were hosted.

Meanwhile, more interesting telecom standards were discovered and implemented, including TETRA professional mobile radio, DECT cordless telephony, GMR satellite telephony, some SDR hardware, a SIM card protocol tracer and many others.

Increasing commercial interest particularly in the BSS and core network components has lead the way to 3G support in Osmocom, as well as the split of the minimal *OsmoNITB* implementation into separate and fully featured network components: OsmoBSC, OsmoMSC, OsmoHLR, OsmoMGW and OsmoSTP (among others), which allow seamless scaling from a simple "Network In The Box" to a distributed installation for serious load.

It has been a most exciting ride during the last eight-odd years. I would not have wanted to miss it under any circumstances.

— Harald Welte, Osmocom.org and OpenBSC founder, December 2017.

Acknowledgements

My deep thanks to everyone who has contributed to Osmocom. The list of contributors is too long to mention here, but I'd like to call out the following key individuals and organizations, in no particular order:

- Dieter Spaar for being the most amazing reverse engineer I've met in my career
- Holger Freyther for his many code contributions and for shouldering a lot of the maintenance work, setting up Jenkins - and being crazy enough to co-start sysmocom as a company with me ;)
- Andreas Eversberg for taking care of Layer2 and Layer3 of OsmocomBB, and for his work on OsmoBTS and OsmoPCU
- Sylvain Munaut for always tackling the hardest problems, particularly when it comes closer to the physical layer
- Chaos Computer Club for providing us a chance to run real-world deployments with tens of thousands of subscribers every year
- Bernd Schneider of Netzing AG for funding early ip.access nanoBTS support
- On-Waves ehf for being one of the early adopters of OpenBSC and funding a never ending list of features, fixes and general improvement of pretty much all of our GSM network element implementations
- sysmocom, for hosting and funding a lot of Osmocom development, the annual Osmocom Developer Conference and releasing this manual.

- Jan Luebbe, Stefan Schmidt, Daniel Willmann, Pablo Neira, Nico Golde, Kevin Redon, Ingo Albrecht, Alexander Huemer, Alexander Chemeris, Max Suraev, Tobias Engel, Jacob Erlbeck, Ivan Kluchnikov

May the source be with you!

— Harald Welte, Osmocom.org and OpenBSC founder, January 2016.

Endorsements

This version of the manual is endorsed by Harald Welte as the official version of the manual.

While the GFDL license (see Appendix C) permits anyone to create and distribute modified versions of this manual, such modified versions must remove the above endorsement.

Preface

First of all, we appreciate your interest in Osmocom software.

Osmocom is a Free and Open Source Software (FOSS) community that develops and maintains a variety of software (and partially also hardware) projects related to mobile communications.

Founded by people with decades of experience in community-driven FOSS projects like the Linux kernel, this community is built on a strong belief in FOSS methodology, open standards and vendor neutrality.

FOSS lives by contribution!

If you are new to FOSS, please try to understand that this development model is not primarily about "free of cost to the GSM network operator", but it is about a collaborative, open development model. It is about sharing ideas and code, but also about sharing the effort of software development and maintenance.

If your organization is benefitting from using Osmocom software, please consider ways how you can contribute back to that community. Such contributions can be many-fold, for example

- sharing your experience about using the software on the public mailing lists, helping to establish best practises in using/operating it,
- providing qualified bug reports, work-arounds
- sharing any modifications to the software you may have made, whether bug fixes or new features, even experimental ones
- providing review of patches
- testing new versions of the related software, either in its current "master" branch or even more experimental feature branches
- sharing your part of the maintenance and/or development work, either by donating developer resources or by (partially) funding those people in the community who do.

We're looking forward to receiving your contributions.

Osmocom and sysmocom

Some of the founders of the Osmocom project have established *sysmocom - systems for mobile communications GmbH* as a company to provide products and services related to Osmocom.

sysmocom and its staff have contributed by far the largest part of development and maintenance to the Osmocom mobile network infrastructure projects.

As part of this work, sysmocom has also created the manual you are reading.

At sysmocom, we draw a clear line between what is the Osmocom FOSS project, and what is sysmocom as a commercial entity. Under no circumstances does participation in the FOSS projects require any commercial relationship with sysmocom as a company.

Corrections

We have prepared this manual in the hope that it will guide you through the process of installing, configuring and debugging your deployment of cellular network infrastructure elements using Osmocom software. If you do find errors, typos and/or omissions, or have any suggestions on missing topics, please do take the extra time and let us know.

Legal disclaimers

Spectrum License

As GSM and UMTS operate in licensed spectrum, please always double-check that you have all required licenses and that you do not transmit on any ARFCN or UARFCN that is not explicitly allocated to you by the applicable regulatory authority in your country.

Warning
Depending on your jurisdiction, operating a radio transmitter without a proper license may be considered a felony under criminal law!

Software License

The software developed by the Osmocom project and described in this manual is Free / Open Source Software (FOSS) and subject to so-called *copyleft* licensing.

Copyleft licensing is a legal instrument to ensure that this software and any modifications, extensions or derivative versions will always be publicly available to anyone, for any purpose, under the same terms as the original program as developed by Osmocom.

This means that you are free to use the software for whatever purpose, make copies and distribute them - just as long as you ensure to always provide/release the *complete and corresponding* source code.

Every Osmocom software includes a file called `COPYING` in its source code repository which explains the details of the license. The majority of programs is released under GNU Affero General Public License, Version 3 (AGPLv3).

If you have any questions about licensing, don't hesitate to contact the Osmocom community. We're more than happy to clarify if your intended use case is compliant with the software licenses.

Trademarks

All trademarks, service marks, trade names, trade dress, product names and logos appearing in this manual are the property of their respective owners. All rights not expressly granted herein are reserved.

For your convenience we have listed below some of the registered trademarks referenced herein. This is not a definitive or complete list of the trademarks used.

Osmocom® and *OpenBSC*® are registered trademarks of Holger Freyther and Harald Welte.

sysmocom® and *sysmoBTS*® are registered trademarks of *sysmocom - systems for mobile communications GmbH*.

ip.access® and *nanoBTS*® are registered trademarks of *ip.access Ltd.*

Liability

The software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the License text included with the software for more details.

Documentation License

Please see Appendix C for further information.

Introduction

Required Skills

Please note that even while the capital expenses of running mobile networks has decreased significantly due to Osmocom software and associated hardware like sysmoBTS, GSM networks are still primarily operated by large GSM operators.

Neither the GSM specification nor the GSM equipment was ever designed for networks to be installed and configured by anyone but professional GSM engineers, specialized in their respective area like radio planning, radio access network, back-haul or core network.

If you do not share an existing background in GSM network architecture and GSM protocols, correctly installing, configuring and optimizing your GSM network will be tough, irrespective whether you use products with Osmocom software or those of traditional telecom suppliers.

GSM knowledge has many different fields, from radio planning through site installation to core network configuration/administration.

The detailed skills required will depend on the type of installation and/or deployment that you are planning, as well as its associated network architecture. A small laboratory deployment for research at a university is something else than a rural network for a given village with a handful of cells, which is again entirely different from an urban network in a dense city.

Some of the useful skills we recommend are:

- general understanding about RF propagation and path loss in order to estimate coverage of your cells and do RF network planning.
- general understanding about GSM network architecture, its network elements and key transactions on the Layer 3 protocol
- general understanding about voice telephony, particularly those of ISDN heritage (Q.931 call control)
- understanding of GNU/Linux system administration and working on the shell
- understanding of TCP/IP networks and network administration, including tcpdump, tshark, wireshark protocol analyzers.
- ability to work with text based configuration files and command-line based interfaces such as the VTY of the Osmocom network elements

Getting assistance

If you do have a support package / contract with sysmocom (or want to get one), please contact support@sysmocom.de with any issues you may have.

If you don't have a support package / contract, you have the option of using the resources put together by the Osmocom community at http://projects.osmocom.org/, checking out the wiki and the mailing-list for community-based assistance. Please always remember, though: The community has no obligation to help you, and you should address your requests politely to them. The information (and software) provided at osmocom.org is put together by volunteers for free. Treat them like a friend whom you're asking for help, not like a supplier from whom you have bought a service.

Overview

About OsmoTRX

OsmoTRX is a C/C++ language implementation of the GSM radio modem, originally developed as the *Transceiver* part of OpenBTS. This radio modem offers an interface based on top of UDP streams.

The OsmoBTS bts_model code for OsmoTRX is called `osmo-bts-trx`. It implements the UDP stream interface of OsmoTRX, so both parts can be used together to implement a complete GSM BTS based on general-purpose computing SDR.

As OsmoTRX is general-purpose software running on top of Linux, it is thus not tied to any specific physical hardware. At the time of this writing, OsmoTRX supports a variety of Lime Microsystems and Ettus USRP SDRs via the UHD driver, as well as the Fairwaves UmTRX and derived products.

OsmoTRX is not a complete GSM PHY but *just* the radio modem. This means that all of the Layer 1 functionality such as scheduling, convolutional coding, etc. is actually also implemented inside OsmoBTS. OsmoTRX is a software-defined radio transceiver that implements the Layer 1 physical layer of a BTS comprising the following 3GPP specifications:

- TS 05.01 "Physical layer on the radio path"
- TS 05.02 "Multiplexing and Multiple Access on the Radio Path"
- TS 05.04 "Modulation"
- TS 05.10 "Radio subsystem synchronization

As such, the boundary between OsmoTRX and `osmo-bts-trx` is at a much lower interface, which is an internal interface of other more traditional GSM PHY implementations.

Besides OsmoTRX, there are also other implementations (both Free Software and proprietary) that implement the same UDP stream based radio modem interface.

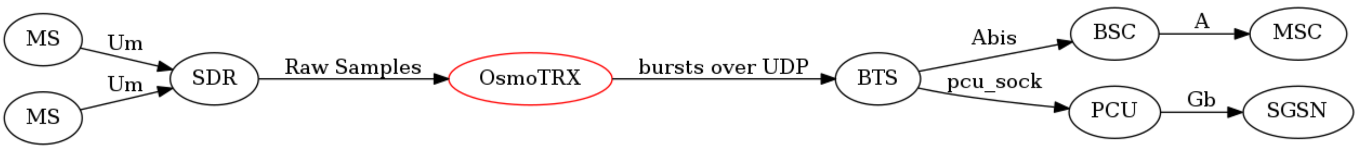

Figure 1: GSM network architecture with OsmoTRX and OsmoBTS

For more information see https://osmocom.org/projects/osmotrx/wiki/OsmoTRX

Running OsmoTRX

The OsmoTRX executable (`osmo-trx`) offers the following command-line options:

SYNOPSIS

osmo-trx [-h] [-C *CONFIGFILE*]

OPTIONS

-h
: Print a short help message about the supported options

-C *CONFIGFILE*
: Specify the file and path name of the configuration file to be used. If none is specified, use `osmo_trx.cfg` in the current working directory.

Osmocom Control Interface

The VTY interface as described in Section 8 is aimed at human interaction with the respective Osmocom program.

Other programs **should not** use the VTY interface to interact with the Osmocom software, as parsing the textual representation is cumbersome, inefficient, and will break every time the formatting is changed by the Osmocom developers.

Instead, the *Control Interface* was introduced as a programmatic interface that can be used to interact with the respective program.

Control Interface Protocol

The control interface protocol is a mixture of binary framing with text based payload.

The protocol for the control interface is wrapped inside the IPA multiplex header with the stream identifier set to IPAC_PROTO_OSMO (0xEE).

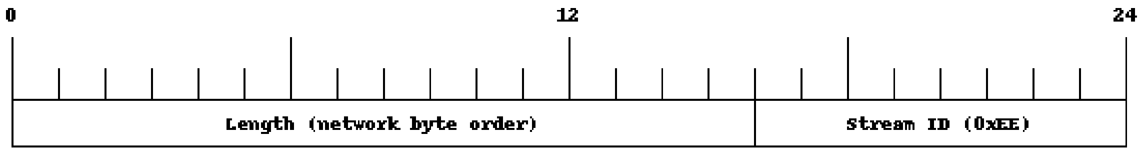

Figure 2: IPA header for control protocol

Inside the IPA header is a single byte of extension header with protocol ID 0x00 which indicates the control interface.

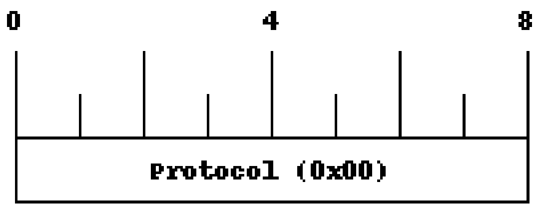

Figure 3: IPA extension header for control protocol

After the concatenation of the two above headers, the plain-text payload message starts. The format of that plain text is illustrated for each operation in the respective message sequence chart in the chapters below.

The fields specified below follow the following meaning:

<id>
: A numeric identifier, uniquely identifying this particular operation. Value `0` is not allowed unless it's a TRAP message. It will be echoed back in any response to a particular request.

<var>
The name of the variable / field affected by the GET / SET / TRAP operation. Which variables/fields are available is dependent on the specific application under control.

<val>
The value of the variable / field

<reason>
A text formatted, human-readable reason why the operation resulted in an error.

GET operation

The GET operation is performed by an external application to get a certain value from inside the Osmocom application.

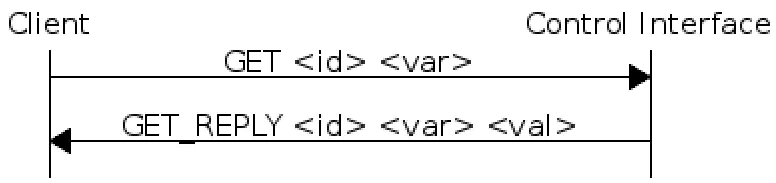

Figure 4: Control Interface GET operation (successful outcome)

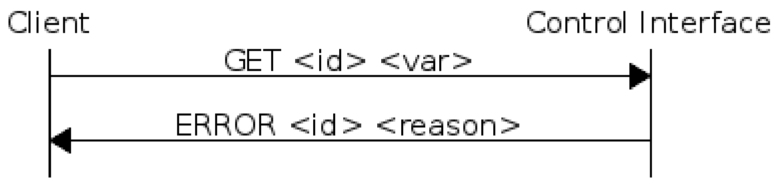

Figure 5: Control Interface GET operation (unsuccessful outcome)

SET operation

The SET operation is performed by an external application to set a value inside the Osmocom application.

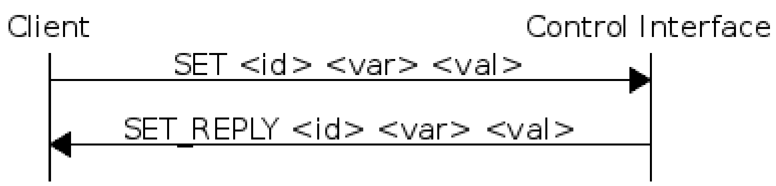

Figure 6: Control Interface SET operation (successful outcome)

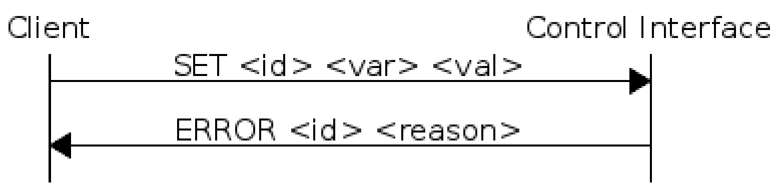

Figure 7: Control Interface SET operation (unsuccessful outcome)

TRAP operation

The program can at any time issue a trap. The term is used in the spirit of SNMP.

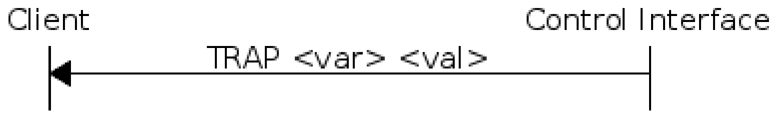

Figure 8: Control Interface TRAP operation

Common variables

There are several variables which are common to all the programs using control interface. They are described in the following table.

Table 1: Variables available over control interface

Name	Access	Value	Comment
counter.*	RO		Get counter value.
rate_ctr.*	RO		Get list of rate counter groups.
rate_ctr.IN.GN.GI.name	RO		Get value for interval IN of rate counter name which belong to group named GN with index GI.

Those read-only variables allow to get value of arbitrary counter using its name.

For example "`rate_ctr.per_hour.bsc.0.handover:timeout`" is the number of handover timeouts per hour.

Of course for that to work the program in question have to register corresponding counter names and groups using libosmocore functions.

In the example above, "`bsc`" is the rate counter group name and "`0`" is its index. It is possible to obtain all the rate counters in a given group by requesting "`rate_ctr.per_sec.bsc.*`" variable.

The list of available groups can be obtained by requesting "`rate_ctr.*`" variable.

The rate counter group name have to be prefixed with interval specification which can be any of "**per_sec**", "**per_min**", "**per_hour**", "**per_day**" or "**abs**" for absolute value.

The old-style counters available via "`counter.*`" variables are superseded by "`rate_ctr.abs`" so its use is discouraged. There might still be some applications not yet converted to rate_ctr.

Control Interface python examples

In the `osmo-python-tests` repository, there is an example python script called `scripts/osmo_ctrl.py` which implements the Osmocom control interface protocol.

You can use this tool either stand-alone to perform control interface operations against an Osmocom program, or you can use it as a reference for developing your own python software talking to the control interface.

Another implementation is in `scripts/osmo_rate_ctr2csv.py` which will retrieve performance counters for a given Osmocom program and output it in csv format. This can be used to periodically (using systemd timer for example) retrieve data to build KPI and evaluate how it changes over time.

Internally it uses "`rate_ctr.*`" variable described in [?] to get the list of counter groups and than request all the counters in each group. Applications interested in individual metrics can request it directly using `rate_ctr2csv.py` as an example.

Getting rate counters

Example: Use rate_ctr2csv.py to get rate counters from OsmoBSC

```
$ ./scripts/osmo_rate_ctr2csv.py --header
Connecting to localhost:4249...
Getting rate counter groups info...
"group","counter","absolute","second","minute","hour","day"
"e1inp.0","hdlc:abort","0","0","0","0","0"
"e1inp.0","hdlc:bad_fcs","0","0","0","0","0"
"e1inp.0","hdlc:overrun","0","0","0","0","0"
"e1inp.0","alarm","0","0","0","0","0"
"e1inp.0","removed","0","0","0","0","0"
"bsc.0","chreq:total","0","0","0","0","0"
"bsc.0","chreq:no_channel","0","0","0","0","0"
...
"msc.0","call:active","0","0","0","0","0"
"msc.0","call:complete","0","0","0","0","0"
"msc.0","call:incomplete","0","0","0","0","0"
Completed: 44 counters from 3 groups received.
```

Setting a value

Example: Use osmo_ctrl.py to set the short network name of OsmoBSC

```
$ ./osmo_ctrl.py -d localhost -s short-name 32C3
Got message: SET_REPLY 1 short-name 32C3
```

Getting a value

Example: Use osmo_ctrl.py to get the mnc of OsmoBSC

```
$ ./osmo_ctrl.py -d localhost -g mnc
Got message: GET_REPLY 1 mnc 262
```

Listening for traps

You can use `osmo_ctrl.py` to listen for traps the following way:

Example: Using osmo_ctrl.py to listen for traps:

```
$ ./osmo_ctrl.py -d localhost -m
❶
```

❶ the command will not return and wait for any TRAP messages to arrive

Control interface

The actual protocol is described in Section 6, the variables common to all programs using it are described in Section 6.2. Here we describe variables specific to OsmoTRX.

Table 2: Variables available over control interface

Name	Access	Trap	Value	Comment

The Osmocom VTY Interface

All human interaction with Osmocom software is typically performed via an interactive command-line interface called the *VTY*.

> **Note**
> Integration of your programs and scripts should **not** be done via the telnet VTY interface, which is intended for human interaction only: the VTY responses may arbitrarily change in ways obvious to humans, while your scripts' parsing will likely break often. For external software to interact with Osmocom programs (besides using the dedicated protocols), it is strongly recommended to use the Control interface instead of the VTY, and to actively request / implement the Control interface commands as required for your use case.

The interactive telnet VTY is used to

- explore the current status of the system, including its configuration parameters, but also to view run-time state and statistics,
- review the currently active (running) configuration,
- perform interactive changes to the configuration (for those items that do not require a program restart),
- store the current running configuration to the config file,
- enable or disable logging; to the VTY itself or to other targets.

The Virtual Tele Type (VTY) has the concept of *nodes* and *commands*. Each command has a name and arguments. The name may contain a space to group several similar commands into a specific group. The arguments can be a single word, a string, numbers, ranges or a list of options. The available commands depend on the current node. there are various keyboard shortcuts to ease finding commands and the possible argument values.

Configuration file parsing during program start is actually performed the VTY's CONFIG node, which is also available in the telnet VTY. Apart from that, the telnet VTY features various interactive commands to query and instruct a running Osmocom program. A main difference is that during config file parsing, consistent indenting of parent vs. child nodes is required, while the interactive VTY ignores indenting and relies on the *exit* command to return to a parent node.

> **Note**
> In the *CONFIG* node, it is not well documented which commands take immediate effect without requiring a program restart. To save your current config with changes you may have made, you may use the `write file` command to **overwrite** your config file with the current configuration, after which you should be able to restart the program with all changes taking effect.

This chapter explains most of the common nodes and commands. A more detailed list is available in various programs' VTY reference manuals, e.g. see [vty-ref-osmomsc].

There are common patterns for the parameters, these include IPv4 addresses, number ranges, a word, a line of text and choice. The following will explain the commonly used syntactical patterns:

Table 3: VTY Parameter Patterns

Pattern	Example	Explanation
`A.B.C.D`	`127.0.0.1`	An IPv4 address
`TEXT`	`example01`	A single string without any spaces, tabs
`.TEXT`	`Some information`	A line of text
`(OptionA|OptionB|OptionC)`	`OptionA`	A choice between a list of available options
`<0-10>`	`5`	A number from a range

Accessing the telnet VTY

The VTY of a given Osmocom program is implemented as a telnet server, listening to a specific TCP port.

Please see Appendix A to check for the default TCP port number of the VTY interface of the specific Osmocom software you would like to connect to.

As telnet is insecure and offers neither strong authentication nor encryption, the VTY by default only binds to localhost (127.0.0.1) and will thus not be reachable by other hosts on the network.

Warning
By default, any user with access to the machine running the Osmocom software will be able to connect to the VTY. We assume that such systems are single-user systems, and anyone with local access to the system also is authorized to access the VTY. If you require stronger security, you may consider using the packet filter of your operating system to restrict access to the Osmocom VTY ports further.

VTY Nodes

The VTY by default has the following minimal nodes:

VIEW
When connecting to a telnet VTY, you will be on the *VIEW* node. As its name implies, it can only be used to view the system status, but it does not provide commands to alter the system state or configuration. As long as you are in the non-privileged *VIEW* node, your prompt will end in a > character.

ENABLE
The *ENABLE* node is entered by the `enable` command, from the *VIEW* node. Changing into the *ENABLE* node will unlock all kinds of commands that allow you to alter the system state or perform any other change to it. The *ENABLE* node and its children are signified by a # character at the end of your prompt.
You can change back from the *ENABLE* node to the *VIEW* node by using the `disable` command.

CONFIG
The *CONFIG* node is entered by the `configure terminal` command from the *ENABLE* node. The config node is used to change the run-time configuration parameters of the system. The prompt will indicate that you are in the config node by a `(config)#` prompt suffix.
You can always leave the *CONFIG* node or any of its children by using the `end` command.
This node is also automatically entered at the time the configuration file is read. All configuration file lines are processed as if they were entered from the VTY *CONFIG* node at start-up.

Other
Depending on the specific Osmocom program you are running, there will be few or more other nodes, typically below the *CONFIG* node. For example, the OsmoBSC has nodes for each BTS, and within the BTS node one for each TRX, and within the TRX node one for each Timeslot.

Interactive help

The VTY features an interactive help system, designed to help you to efficiently navigate is commands.

Note
The VTY is present on most Osmocom GSM/UMTS/GPRS software, thus this chapter is present in all the relevant manuals. The detailed examples below assume you are executing them on the OsmoMSC VTY. They will work in similar fashion on the other VTY interfaces, while the node structure will differ in each program.

The question-mark (?) command

If you type a single ? at the prompt, the VTY will display possible completions at the exact location of your currently entered command.

If you type ? at an otherwise empty command (without having entered even only a partial command), you will get a list of the first word of all possible commands available at this node:

Example: Typing ? at start of OsmoMSC prompt

```
OsmoMSC> ❶
  show         Show running system information
  list         Print command list
  exit         Exit current mode and down to previous mode
  help         Description of the interactive help system
  enable       Turn on privileged mode command
  terminal     Set terminal line parameters
  who          Display who is on vty
  logging      Configure logging
  no           Negate a command or set its defaults
  sms          SMS related commands
  subscriber   Operations on a Subscriber
```

❶ Type ? here at the prompt, the ? itself will not be printed.

If you have already entered a partial command, ? will help you to review possible options of how to continue the command. Let's say you remember that show is used to investigate the system status, but you don't remember the exact name of the object. Hitting ? after typing show will help out:

Example: Typing ? after a partial command

```
OsmoMSC> show ❶
  version            Displays program version
  online-help        Online help
  history            Display the session command history
  cs7                ITU-T Signaling System 7
  logging            Show current logging configuration
  alarms             Show current logging configuration
  talloc-context     Show talloc memory hierarchy
  stats              Show statistical values
  asciidoc           Asciidoc generation
  rate-counters      Show all rate counters
  fsm                Show information about finite state machines
  fsm-instances      Show information about finite state machine instances
  sgs-connections    Show SGs interface connections / MMEs
  subscriber         Operations on a Subscriber
  bsc                BSC
  connection         Subscriber Connections
  transaction        Transactions
  statistics         Display network statistics
  sms-queue          Display SMSqueue statistics
  smpp            SMPP Interface
```

❶ Type ? after the show command, the ? itself will not be printed.

You may pick the bsc object and type ? again:

Example: Typing ? after show bsc

```
OsmoMSC> show bsc
  <cr>
```

By presenting <cr> as the only option, the VTY tells you that your command is complete without any remaining arguments being available, and that you should hit enter, a.k.a. "carriage return".

TAB completion

The VTY supports tab (tabulator) completion. Simply type any partial command and press `<tab>`, and it will either show you a list of possible expansions, or completes the command if there's only one choice.

Example: Use of <tab> pressed after typing only s as command

```
OsmoMSC> s❶
show       sms        subscriber
```

❶ Type `<tab>` here.

At this point, you may choose `show`, and then press `<tab>` again:

Example: Use of <tab> pressed after typing show command

```
OsmoMSC> show ❶
version    online-help history    cs7        logging    alarms
talloc-context stats      asciidoc   rate-counters fsm        fsm-instances
sgs-connections subscriber bsc        connection transaction statistics
sms-queue smpp
```

❶ Type `<tab>` here.

The `list` command

The `list` command will give you a full list of all commands and their arguments available at the current node:

Example: Typing list at start of OsmoMSC VIEW node prompt

```
OsmoMSC> list
  show version
  show online-help
  list
  exit
  help
  enable
  terminal length <0-512>
  terminal no length
  who
  show history
  show cs7 instance <0-15> users
  show cs7 (sua|m3ua|ipa) [<0-65534>]
  show cs7 instance <0-15> asp
  show cs7 instance <0-15> as (active|all|m3ua|sua)
  show cs7 instance <0-15> sccp addressbook
  show cs7 instance <0-15> sccp users
  show cs7 instance <0-15> sccp ssn <0-65535>
  show cs7 instance <0-15> sccp connections
  show cs7 instance <0-15> sccp timers
  logging enable
  logging disable
  logging filter all (0|1)
  logging color (0|1)
  logging timestamp (0|1)
  logging print extended-timestamp (0|1)
  logging print category (0|1)
  logging print category-hex (0|1)
  logging print level (0|1)
  logging print file (0|1|basename) [last]
```

```
  logging set-log-mask MASK
  logging level (rll|cc|mm|rr|mncc|pag|msc|mgcp|ho|db|ref|ctrl|smpp|ranap|vlr|iucs|bssap| ↩
      sgs|lglobal|llapd|linp|lmux|lmi|lmib|lsms|lctrl|lgtp|lstats|lgsup|loap|lss7|lsccp|lsua ↩
      |lm3ua|lmgcp|ljibuf|lrspro) (debug|info|notice|error|fatal)
  logging level set-all (debug|info|notice|error|fatal)
  logging level force-all (debug|info|notice|error|fatal)
  no logging level force-all
  show logging vty
  show alarms
  show talloc-context (application|all) (full|brief|DEPTH)
  show talloc-context (application|all) (full|brief|DEPTH) tree ADDRESS
  show talloc-context (application|all) (full|brief|DEPTH) filter REGEXP
  show stats
  show stats level (global|peer|subscriber)
  show asciidoc counters
  show rate-counters
  show fsm NAME
  show fsm all
  show fsm-instances NAME
  show fsm-instances all
  show sgs-connections
  show subscriber (msisdn|extension|imsi|tmsi|id) ID
  show subscriber cache
  show bsc
  show connection
  show transaction
  sms send pending
  sms delete expired
  subscriber create imsi ID
  subscriber (msisdn|extension|imsi|tmsi|id) ID sms sender (msisdn|extension|imsi|tmsi|id)  ↩
      SENDER_ID send .LINE
  subscriber (msisdn|extension|imsi|tmsi|id) ID silent-sms sender (msisdn|extension|imsi| ↩
      tmsi|id) SENDER_ID send .LINE
  subscriber (msisdn|extension|imsi|tmsi|id) ID silent-call start (any|tch/f|tch/any|sdcch)
  subscriber (msisdn|extension|imsi|tmsi|id) ID silent-call stop
  subscriber (msisdn|extension|imsi|tmsi|id) ID ussd-notify (0|1|2) .TEXT
  subscriber (msisdn|extension|imsi|tmsi|id) ID ms-test close-loop (a|b|c|d|e|f|i)
  subscriber (msisdn|extension|imsi|tmsi|id) ID ms-test open-loop
  subscriber (msisdn|extension|imsi|tmsi|id) ID paging
  show statistics
  show sms-queue
  logging filter imsi IMSI
  show smpp esme
```

Tip

Remember, the list of available commands will change significantly depending on the Osmocom program you are accessing, its software version and the current node you're at. Compare the above example of the OsmoMSC *VIEW* node with the list of the OsmoMSC *NETWORK* config node:

Example: Typing list at start of OsmoMSC NETWORK config node prompt

```
OsmoMSC(config-net)# list
  help
  list
  write terminal
  write file
  write memory
  write
  show running-config
```

```
exit
end
network country code <1-999>
mobile network code <0-999>
short name NAME
long name NAME
encryption a5 <0-3> [<0-3>] [<0-3>] [<0-3>]
authentication (optional|required)
rrlp mode (none|ms-based|ms-preferred|ass-preferred)
mm info (0|1)
timezone <-19-19> (0|15|30|45)
timezone <-19-19> (0|15|30|45) <0-2>
no timezone
periodic location update <6-1530>
no periodic location update
```

libosmocore Logging System

In any reasonably complex software it is important to understand how to enable and configure logging in order to get a better insight into what is happening, and to be able to follow the course of action. We therefore ask the reader to bear with us while we explain how the logging subsystem works and how it is configured.

Most Osmocom Software (like `osmo-bts`, `osmo-bsc`, `osmo-nitb`, `osmo-sgsn` and many others) uses the same common logging system.

This chapter describes the architecture and configuration of this common logging system.

The logging system is composed of

- log targets (where to log),
- log categories (who is creating the log line),
- log levels (controlling the verbosity of logging), and
- log filters (filtering or suppressing certain messages).

All logging is done in human-readable ASCII-text. The logging system is configured by means of VTY commands that can either be entered interactively, or read from a configuration file at process start time.

Log categories

Each sub-system of the program in question typically logs its messages as a different category, allowing fine-grained control over which log messages you will or will not see. For example, in OsmoBSC, there are categories for the protocol layers `rsl`, `rr`, `mm`, `cc` and many others. To get a a list of categories interactively on the vty, type: `logging level ?`

Log levels

For each of the log categories (see Section 9.1), you can set an independent log level, controlling the level of verbosity. Log levels include:

fatal
Fatal messages, causing abort and/or re-start of a process. This *shouldn't happen*.

error
An actual error has occurred, its cause should be further investigated by the administrator.

notice
A noticeable event has occurred, which is not considered to be an error.

info
Some information about normal/regular system activity is provided.

debug
Verbose information about internal processing of the system, used for debugging purpose. This will log the most.

The log levels are inclusive, e.g. if you select *info*, then this really means that all events with a level of at least *info* will be logged, i.e. including events of *notice*, *error* and *fatal*.

So for example, in OsmoBSC, to set the log level of the Mobility Management category to info, you can use the following command: `log level mm info`.

There is also a special command to set all categories as a one-off to a desired log level. For example, to silence all messages but those logged as notice and above issue the command: `log level set-all notice`

Afterwards you can adjust specific categories as usual.

A similar command is `log level force-all <level>` which causes all categories to behave as if set to log level <level> until the command is reverted with `no log level force-all` after which the individually-configured log levels will again take effect. The difference between `set-all` and `force-all` is that `set-all` actually changes the individual category settings while `force-all` is a (temporary) override of those settings and does not change them.

Log printing options

The logging system has various options to change the information displayed in the log message.

log color 1
With this option each log message will log with the color of its category. The color is hard-coded and can not be changed. As with other options a *0* disables this functionality.

log timestamp 1
Includes the current time in the log message. When logging to syslog this option should not be needed, but may come in handy when debugging an issue while logging to file.

log print extended-timestamp 1
In order to debug time-critical issues this option will print a timestamp with millisecond granularity.

log print category 1
Prefix each log message with the category name.

log print category-hex 1
Prefix each log message with the category number in hex (*<000b>*).

log print level 1
Prefix each log message with the name of the log level.

log print file 1
Prefix each log message with the source file and line number. Append the keyword `last` to append the file information instead of prefixing it.

Log filters

The default behavior is to filter out everything, i.e. not to log anything. The reason is quite simple: On a busy production setup, logging all events for a given subsystem may very quickly be flooding your console before you have a chance to set a more restrictive filter.

To request no filtering, i.e. see all messages, you may use: `log filter all 1`

In addition to generic filtering, applications can implement special log filters using the same framework to filter on particular context.

For example in OsmoBSC, to only see messages relating to a particular subscriber identified by his IMSI, you may use: `log filter imsi 262020123456789`

Log targets

Each of the log targets represent certain destination for log messages. It can be configured independently by selecting levels (see Section 9.2) for categories (see Section 9.1) as well as filtering (see Section 9.4) and other options like `logging timestamp` for example.

Logging to the VTY

Logging messages to the interactive command-line interface (VTY) is most useful for occasional investigation by the system administrator.

Logging to the VTY is disabled by default, and needs to be enabled explicitly for each such session. This means that multiple concurrent VTY sessions each have their own logging configuration. Once you close a VTY session, the log target will be destroyed and your log settings be lost. If you re-connect to the VTY, you have to again activate and configure logging, if you wish.

To create a logging target bound to a VTY, you have to use the following command: `logging enable` This doesn't really activate the generation of any output messages yet, it merely creates and attaches a log target to the VTY session. The newly-created target still doesn't have any filter installed, i.e. *all log messages will be suppressed by default*

Next, you can configure the log levels for desired categories in your VTY session. See Section 9.1 for more details on categories and Section 9.2 for the log level details.

For example, to set the log level of the Call Control category to debug, you can use: `log level cc debug`

Finally, after having configured the levels, you still need to set the filter as it's described in Section 9.4.

Tip
If many messages are being logged to a VTY session, it may be hard to impossible to still use the same session for any commands. We therefore recommend to open a second VTY session in parallel, and use one only for logging, while the other is used for interacting with the system. Another option would be to use different log target.

To review the current vty logging configuration, you can use: `show logging vty`

Logging to the ring buffer

To avoid having separate VTY session just for logging output while still having immediate access to them, one can use `alarms` target. It lets you store the log messages inside the ring buffer of a given size which is available with `show alarms` command.

It's configured as follows:

```
OsmoBSC> enable
OsmoBSC# configure terminal
OsmoBSC(config)# log alarms 98
OsmoBSC(config-log)#
```

In the example above 98 is the desired size of the ring buffer (number of messages). Once it's filled, the incoming log messages will push out the oldest messages available in the buffer.

Logging via gsmtap

When debugging complex issues it's handy to be able to reconstruct exact chain of events. This is enabled by using GSMTAP log output where frames sent/received over the air are intersperced with the log lines. It also simplifies the bug handling as users don't have to provide separate .pcap and .log files anymore - everything will be inside self-contained packet dump.

It's configured as follows:

```
OsmoBSC> enable
OsmoBSC# configure terminal
OsmoBSC(config)# log gsmtap 192.168.2.3
OsmoBSC(config-log)#
```

The hostname/ip argument is optional: if omitted the default 127.0.0.1 will be used. The log strings inside GSMTAP are already supported by Wireshark. Capturing for `port 4729` on appropriate interface will reveal log messages including source file name and line number as well as application. This makes it easy to consolidate logs from several different network components alongside the air frames. You can also use Wireshark to quickly filter logs for a given subsystem, severity, file name etc.

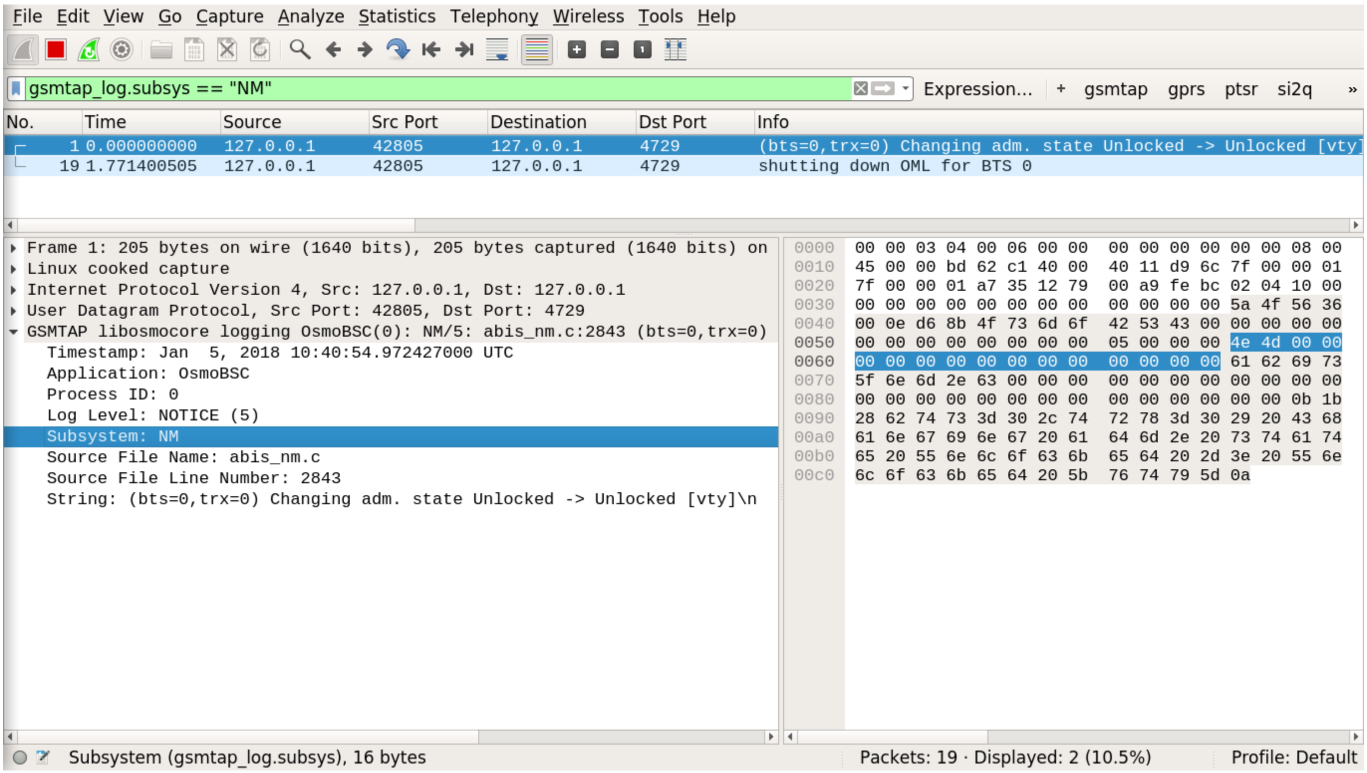

Figure 9: Wireshark with logs delivered over GSMTAP

Note: the logs are also duplicated to stderr when GSMTAP logging is configured because stderr is the default log target which is initialized automatically. To descrease stderr logging to absolute minimum, you can configure it as follows:

```
OsmoBSC> enable
OsmoBSC# configure terminal
OsmoBSC(config)# log stderr
OsmoBSC(config-log)# logging level force-all fatal
```

Logging to a file

As opposed to Logging to the VTY, logging to files is persistent and stored in the configuration file. As such, it is configured in sub-nodes below the configuration node. There can be any number of log files active, each of them having different settings regarding levels / subsystems.

To configure a new log file, enter the following sequence of commands:

```
OsmoBSC> enable
OsmoBSC# configure terminal
OsmoBSC(config)# log file /path/to/my/file
OsmoBSC(config-log)#
```

This leaves you at the config-log prompt, from where you can set the detailed configuration for this log file. The available commands at this point are identical to configuring logging on the VTY, they include `logging filter`, `logging level` as well as `logging color` and `logging timestamp`.

Tip
Don't forget to use the `copy running-config startup-config` (or its short-hand `write file`) command to make your logging configuration persistent across application re-start.

Note
libosmocore provides file close-and-reopen support by SIGHUP, as used by popular log file rotating solutions such as https://github.com/logrotate/logrotate found in most GNU/Linux distributions.

Logging to syslog

syslog is a standard for computer data logging maintained by the IETF. Unix-like operating systems like GNU/Linux provide several syslog compatible log daemons that receive log messages generated by application programs.

libosmocore based applications can log messages to syslog by using the syslog log target. You can configure syslog logging by issuing the following commands on the VTY:

```
OsmoBSC> enable
OsmoBSC# configure terminal
OsmoBSC(config)# log syslog daemon
OsmoBSC(config-log)#
```

This leaves you at the config-log prompt, from where you can set the detailed configuration for this log file. The available commands at this point are identical to configuring logging on the VTY, they include `logging filter`, `logging level` as well as `logging color` and `logging timestamp`.

Note
Syslog daemons will normally automatically prefix every message with a time-stamp, so you should disable the libosmocore time-stamping by issuing the `logging timestamp 0` command.

Logging to stderr

If you're not running the respective application as a daemon in the background, you can also use the stderr log target in order to log to the standard error file descriptor of the process.

In order to configure logging to stderr, you can use the following commands:

```
OsmoBSC> enable
OsmoBSC# configure terminal
OsmoBSC(config)# log stderr
OsmoBSC(config-log)#
```

Counters

These counters and their description are based on OsmoTRX 1.0.0.95-9527 (OsmoTRX).

Rate Counters

Table 4: trx:chan - osmo-trx statistics

Name	Reference	Description
device:rx_overruns	[?]	Number of Rx overruns in FIFO queue
device:tx_underruns	[?]	Number of Tx underruns in FIFO queue
device:rx_drop_events	[?]	Number of times Rx samples were dropped by HW
device:rx_drop_samples	[?]	Number of Rx samples dropped by HW
device:tx_drop_events	[?]	Number of times Tx samples were dropped by HW
device:tx_drop_samples	[?]	Number of Tx samples dropped by HW

Osmo Stat Items

Osmo Counters

Rate Counter Configurable Error Thresholds

Some rate counters such as overruns, underruns and dropped packets indicate events that can really harm correct operation of the BTS served by OsmoTRX, specially if they happen frequently. OsmoTRX is in most cases (depending on maturity of device driver) prepared to dodge the temporary failure and keep running and providing service.

Still, it is sometimes important for this kind of events to not go unnoticed by the operator, since they may indicate issues regarding the set up that may require operator intervention to fix it.

For instance, frequent dropped packets could indicate SDR HW/FW/power errors, or a faulty connection against the host running OsmoTRX.

They can also indicate issues on the host running OsmoTRX itself: For instance, OsmoTRX may not be running under a high enough priority (hence other processes eventually battling for resources with it), or that simply the HW running OsmoTRX is not powerful enough to accomplish all work in a timely fashion all the time.

As a result, OsmoTRX can be configured to exit the process upon certain conditions being met, in order to let osmoBTS notice something is wrong and thus announcing issues through alarms to the network, where the operator can then investigate the issue by looking at OsmoTRX logs.

These conditions are configured by means of introducing rate counter thresholds in the VTY. The OsmoTRX user can provide those threshold commands either in the VTY cfg file read by OsmoTRX process during startup, or by adding/removing them dynamically through the VTY interactive console.

Each threshold cmd states an event (a rate counter type), a value and an time interval (a second, a minute, an hour or a day). A threshold will be reached (and OsmoTRX stopped) if its value grows bigger than the configured threshold value over the configured time interval. This is the syntax used to manage rate counter thresholds:

```
(no) ctr-error-threshold <EVENT> <VALUE> <INTERVAL>
```

If several rate counter thresholds are set, then all of them are checked over time and the first one reached will stop OsmoTRX.

Example: rate counter threshold configuration (VTY .cfg file)

```
trx
 ctr-error-threshold rx_drop_events 2 per-minute ❶
 ctr-error-threshold rx_drop_samples 800 per-second ❷
```

❶ Stop OsmoTRX if dropped event (any amount of samples) during Rx was detected 2 times or more during a minute.

❷ Stop OsmoTRX if 800 or more samples were detected during Rx to be dropped by the HW during a second.

Example: rate counter threshold configuration (VTY interactive)

```
OsmoTRX(config-trx)# ctr-error-threshold tx_underruns 3 per-hour ❶
OsmoTRX(config-trx)# no ctr-error-threshold tx_underruns 3 per-hour ❷
```

❶ Stop OsmoTRX if 3 or more underruns were detected during Tx over the last hour

❷ Remove previously set rate counter threshold

Configuring OsmTRX

OsmoTRX will read the configuration at startup time and configure the transceiver accordingly after validating the configuration.

OsmoTRX can handle several TRX channels, but at least one must be configured in order to be able to start it successfully. Channels must be present in the configuration file in incremental order, starting from 0 and be consecutive.

Example configuration files for different devices and setups can be found in `doc/examples/` in *osmo-trx* git repository.

Documented example

Example: Single carrier configuration

```
trx
 bind-ip 127.0.0.1 ❶
 remote-ip 127.0.0.1 ❷
 base-port 5700 ❸
 egprs disable ❹
 tx-sps 4 ❺
 rx-sps 4 ❻
 chan 0 ❼
  tx-path BAND1 ❽
  rx-path LNAW ❾
```

❶ Configure the local IP address at the TRX used for the connection against `osmo-bts-trx`.

❷ Specify the IP address of `osmo-bts-trx` to connect to.

❸ Specify the reference base UDP port to use for communication.

❹ Don't enable EDGE support.

❺ Use 4 TX samples per symbol. This is device specific.

❻ Use 4 RX samples per symbol. This is device specific.

❼ Configure the first channel. As no other channels are specified, `osmo-trx` assumes it is using only one channel.

❽ Configure the device to use `BAND1` Tx antenna path from all the available ones (device specific).

❾ Configure the device to use `LNAW` Rx antenna path from all the available ones (device specific).

Multi-ARFCN mode

The Multi-ARFCN feature allows to have a multi-carrier approach multiplexed on a single physical RF channel, which can introduce several benefits, such as lower cost and higher capacity support.

Multi-ARFCN support is available since osmo-trx release `0.2.0`, and it was added specifically in commit `76764278169d252980853251daeb9f1ba0c246e1`.

This feature is useful for instance if you want to run more than 1 TRX with an Ettus B200 device, or more than 2 TRXs with an Ettus B210 device, since they support only 1 and 2 physical RF channels respectively. No device from other providers or even other devices than B200 and B210 from Ettus are known to support this feature.

With multi-ARFCN enabled, ARFCN spacing is fixed at 800 kHz or 4 GSM channels. So if TRX-0 is set to ARFCN 51, TRX-1 *must* be set to 55, and so on. Up to three ARFCN's is supported for multi-TRX.

From BTS and BSC point of view, supporting multiple TRXs through multi-ARFCN feature in OsmoTRX doesn't make any difference from a regular multi-TRX setup, leaving apart of course the mentioned ARFCN limitations explained above and as a consequence physical installation and operational differences.

Example: osmo-bts-trx.cfg using 2 TRX against an osmo-trx driven device

```
phy 0
 osmotrx ip local 127.0.0.1
 osmotrx ip remote 127.0.0.1
 instance 0
 instance 1
bts 0
 ...
 band GSM-1800
 trx 0
  phy 0 instance 0
 trx 1
  phy 0 instance 1
```

Example: osmo-trx.cfg using Multi-ARFCN mode to run 2 TRX

```
trx
 ...
 multi-arfcn enable
 chan 0
 chan 1
```

OsmoTRX hardware architecture support

OsmoTRX comes out-of-the-box with several algorithms and operations optimized for certain instruction-set architectures, as well as non-optimized fall-back algorithms in case required instruction sets are not supported by the compiler at compile time or by the executing machine at run-time. Support for these optimized algorithms can be enabled and disabled by means of configure flags. Accelerated operations include pulse shape filtering, resampling, sequence correlation, and many other signal processing operations.

On Intel processors, OsmoTRX makes heavy use of the Streaming SIMD Extensions (SSE) instruction set. SSE3 is the minimum requirement for accelerated use. SSE3 is present in the majority of Intel processors since later versions of the Pentium 4 architecture and is also present on low power Atom processors. Support is automatically detected at build time. SSE4.1 instruction set is supported too. This feature is enabled by default unless explicitly disabled by passing the configure flag *--with-sse=no*. When enabled, the compiler will build an extra version of each of the supported algorithms using each of the supported mentioned instruction sets. Then, at run-time, OsmoTRX will auto-detect capabilities of the executing machine and enable an optimized algorithm using the most suitable available (previously compiled) instruction set.

On ARM processors, NEON and NEON FMA are supported. Different to the x86, there is no auto-detection in this case, nor difference between compile and runtime. NEON support is disabled by default and can be enabled by passing the flag *--with-neon=yes* to the configure script; the used compiler must support NEON instruction set and the resulting binary will only run fine on an ARM board supporting NEON extensions. Running OsmoTRX built with flag *--with-neon* on a board without NEON instruction set support, will most probably end up in the process being killed with a *SIGILL* Illegal Instruction signal by the operating system. NEON FMA (Fused Multiply-Add) is an extension to the NEON instruction set, and its use in OsmoTRX can be enabled by passing the *--with_neon_vfpv4* flag, which will also implicitly enable NEON support (*--with_neon*).

OsmoTRX hardware device support

OsmoTRX consists of a *common* part that applies to all TRX devices as well as *hardware-specific* parts for each TRX device. The hardware-specific parts are usually provided by vendor-specific or device-specific libraries that are then handled by some OsmoTRX glue code presenting a unified interface towards the rest of the code by means of a *RadioDevice* class.

The common part includes the core TRX architecture as well as code for implementing the external interfaces such as the TRX Manager UDP socket, control, and VTY interfaces.

The hardware-specific parts include support for driving one particular implementation of a radio modem. Such a physical layer implementation can come in many forms. Sometimes it runs on a general purpose CPU, sometimes on a dedicated ARM core, a dedicated DSP, a combination of DSP and FPGA.

Joining the common part with each of the available backends results in a different binary with different suffix for each backend. For instance, when OsmoTRX is built with UHD backend, an *osmo-trx-uhd* binary is generated; when OsmoTRX is built with LimeSuite backend, an *osmo-trx-lms* binary is generated. Build of different backend can be enabled and disabled by means of configure flags, which can be found in each subsection relative to each backend below.

Ettus USRP1

The binary *osmo-trx-usrp1* is used to drive this device, see Section 16.3.

USRP1 in-band USB protocol

This section specifies the format of USB packets used for in-band data transmission and signaling on the USRP1. All packets are 512-byte long, and are transferred using USB "bulk" transfers.

IN packets are sent towards the host. OUT packets are sent away from the host.

The layout is 32-bits wide. All data is transmitted in little-endian format across the USB.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|O|U|D|S|E|   RSSI    |  Chan    | mbz |  Tag  |    Payload Len   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Timestamp                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                                 |
+                                                                 +
|                             Payload                             |
.                                                                 .
.                                                                 .
.                                                                 .
|                                                                 |
+             +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ...    |                                                   .
+-+-+-+-+-+-+-+                                                   .
.                                                                 .
.                            Padding                              .
.                                                                 .
|                                                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```


mbz: Must be Zero
These bits must be zero in both IN and OUT packets.

O: Overrun Flag
Set in an IN packet if an overrun condition was detected. Must be zero in OUT packets. Overrun occurs when the FPGA has data to transmit to the host and there is no buffer space available. This generally indicates a problem on the host. Either it is not keeping up, or it has configured the FPGA to transmit data at a higher rate than the transport (USB) can support.

U: Underrun Flag
Set in an IN packet if an underrun condition was detected. Must be zero in OUT packets. Underrun occurs when the FPGA runs out of samples, and it's not between bursts. See the "End of Burst flag" below.

D: Dropped Packet Flag
Set in an IN packet if the FPGA discarded an OUT packet because its timestamp had already passed.

S: Start of Burst Flag
Set in an OUT packet if the data is the first segment of what is logically a continuous burst of data. Must be zero in IN packets.

E: End of Burst Flag
Set in an OUT packet if the data is the last segment of what is logically a continuous burst of data. Must be zero in IN packets. Underruns are not reported when the FPGA runs out of samples between bursts.

RSSI: 6-bit Received Strength Signal Indicator
Must be zero in OUT packets. In IN packets, indicates RSSI as reported by front end. FIXME The format and interpretation are to be determined.

Chan: 5-bit logical channel number
Channel number 0x1f is reserved for control information. See "Control Channel" below. Other channels are "data channels". Each data channel is logically independent of the others. A data channel payload field contains a sequence of homogeneous samples. The format of the samples is determined by the configuration associated with the given channel. It is often the case that the payload field contains 32-bit complex samples, each containing 16-bit real and imaginary components.

Tag
4-bit tag for matching IN packets with OUT packets.

Payload Len
9-bit field that specifies the length of the payload field in bytes. Must be in the range 0 to 504 inclusive.

Timestamp: 32-bit timestamp
On IN packets, the timestamp indicates the time at which the first sample of the packet was produced by the A/D converter(s) for that channel. On OUT packets, the timestamp specifies the time at which the first sample in the packet should go out the D/A converter(s) for that channel. If a packet reaches the head of the transmit queue, and the current time is later than the timestamp, an error is assumed to have occurred and the packet is discarded. As a special case, the timestamp 0xffffffff is interpreted as "Now". The time base is a free running 32-bit counter that is incremented by the A/D sample-clock.

Payload
Variable length field Length is specified by the Payload Len field.

Padding
This field is 504 - Payload Len bytes long, and its content is unspecified. This field pads the packet out to a constant 512 bytes.

"Data Channel" payload format

If `Chan !=0x1f`, the packet is a "data packet" and the payload is a sequence of homogeneous samples. The format of the samples is determined by the configuration associated with the given channel. It is often the case that the payload field contains 32-bit complex samples, each containing 16-bit real and imaginary components.

"Control Channel" payload format

If `Chan ==0x1f`, the packet is a "control packet". The control channel payload consists of a sequence of 0 or more sub-packets.

Each sub-packet starts on a 32-bit boundary, and consists of an 8-bit Opcode field, an 8-bit Length field, Length bytes of arguments, and 0, 1, 2 or 3 bytes of padding to align the tail of the sub-packet to a 32-bit boundary.

Control channel packets shall be processed at the head of the queue, and shall observe the timestamp semantics described above.

General sub-packet format

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-//-+-+-+-+-+-+-+-+
|     Opcode    |    Length     |        <length bytes> ...     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-//-+-+-+-+-+-+-+-+
```

Specific sub-packet formats

RID: 6-bit Request-ID
Copied from request sub-packet into corresponding reply sub-packet. RID allows the host to match requests and replies.

Reg Number
10-bit Register Number.

Ping Fixed Length

- Opcode: OP_PING_FIXED

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |       2       |    RID    |      Ping Value   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Ping Fixed Length Reply

- Opcode: OP_PING_FIXED_REPLY

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |       2       |    RID    |      Ping Value   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Write Register

- Opcode: OP_WRITE_REG

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |       6       |    mbz    |      Reg Number   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Register Value                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Write Register Masked
Only the register bits that correspond to 1's in the mask are written with the new value. `REG[Num] =(REG[Num] & ~Mask) | (Value & Mask)`

- Opcode: OP_WRITE_REG_MASKED

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |       10      |     mbz     |      Reg Number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Register Value                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Mask Value                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Read Register

- Opcode: OP_READ_REG

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |        2      |     RID     |      Reg Number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Read Register Reply

- Opcode: OP_READ_REG_REPLY

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |        6      |     RID     |      Reg Number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Register Value                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

I2C Write

- Opcode: OP_I2C_WRITE
- I2C Addr: 7-bit I2C address
- Data: The bytes to write to the I2C bus
- Length: Length of Data + 2

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |    Length     |         mbz         |   I2C Addr  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Data ...                                                      .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

I2C Read

- Opcode: OP_I2C_READ
- I2C Addr: 7-bit I2C address
- Nbytes: Number of bytes to read from I2C bus

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |        3      |     RID     | mbz |   I2C Addr  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Nbytes    |              unspecified padding                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

I2C Read Reply

- Opcode: OP_I2C_READ_REPLY

- I2C Addr: 7-bit I2C address
- Data: Length - 2 bytes of data read from I2C bus.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |     Length    |    RID    | mbz |   I2C Addr  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Data ...                                                   .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

SPI Write

- Opcode: OP_SPI_WRITE
- Enables: Which SPI enables to assert (mask)
- Format: Specifies format of SPI data and Opt Header Bytes
- Opt Header Bytes: 2-byte field containing optional Tx bytes; see Format
- Data: The bytes to write to the SPI bus
- Length: Length of Data + 6

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |    Length     |              mbz              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Enables    |    Format     |        Opt Header Bytes       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Data ...                                                   .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

SPI Read

- Opcode: OP_SPI_READ
- Enables: Which SPI enables to assert (mask)
- Format: Specifies format of SPI data and Opt Header Bytes
- Opt Header Bytes: 2-byte field containing optional Tx bytes; see Format
- Nbytes: Number of bytes to read from SPI bus.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |       7       |    RID    |        mbz        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Enables    |    Format     |        Opt Header Bytes       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Nbytes    |              unspecified padding              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

SPI Read Reply

- Opcode: OP_SPI_READ_REPLY
- Data: Length - 2 bytes of data read from SPI bus.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |     Length    |    RID    |        mbz        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Data ...                                                   .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Delay

- Opcode: OP_DELAY
- Ticks: 16-bit unsigned delay count
- Delay Ticks clock ticks before executing next operation.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Opcode    |       2       |              Ticks              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Ettus B200

The binary *osmo-trx-uhd* is used to drive this device, see Section 16.1.

Comes only with 1 RF channel. It can still be used in a multi-TRX setup by using the Section 13.2 feature. By using this feature, one can drive up to 3 TRX (with the restrictions explained there).

Ettus B210

The binary *osmo-trx-uhd* is used to drive this device, see Section 16.1.

Comes with 2 RF channels, which can be used to set up a multi-TRX BTS. However, due to a shared local oscillator for both RF channels, ARFCN separation can be up about 25 MHz.

This device also supports the Section 13.2 feature. By using this feature, one can drive up to 3 TRX (with the restrictions explained there). Please note that the above configurations cannot be combined, which means maximum number of TRX one can achieve is 2 by using separate physical RF channels, or 3 by using multi-ARFCN method. You cannot support, for example, 6 ARFCN operation on B210 using 3 TRX on side A and another 3 TRX on side B.

LimeSDR-USB

The binary *osmo-trx-lms* is used to drive this device, see Section 16.2.

This device comes with 2 RF channels, so it should theoretically be possible to run a multi-TRX setup with it, but there are yet no records that this kind of setup was tested with this device.

This device has 3 different Rx paths with different antenna connectors in the PCB, each with a different frequency and bandwidth range. One should make sure the physical antenna is connected to the correct connector matching the Rx path you want to use. If one wants to be able to use the device in both 900 and 1800 MHz GSM bands and easily switch between them, then Rx Path `LNAW` should be used ,since it is the only one covering both bands, and the antenna physically plugged accordingly. Following example shows how to then configure *osmo-trx-lms* to use that Rx path to read samples.

Example: Configure osmo-trx-lms to use LNAW as Rx path and BAND1 as Tx Path

```
trx
 ...
 chan 0
  tx-path BAND1
  rx-path LNAW
```

LimeSDR-mini

The binary *osmo-trx-lms* is used to drive this device, see Section 16.2.

As a smaller brother of the , this device comes only with 1 RF channel. As a result, it can only hold 1 TRX as of today.

OsmoTRX backend support

`osmo-trx-uhd` for UHD based Transceivers

This OsmoTRX model uses *libuhd* (UHD, USRP Hardware Driver) to drive the device, that is configuring it and reading/writing samples from/to it.

So far, this backend has been mostly used to drive devices such as the Ettus B200 family and Fairwaves UmTRX family, and used to be the default backend used for legacy @osmo-trx@ binary when per-backend binaries didn't exist yet.

Any device providing generic support for UHD should theoretically be able to be run through this backend without much effort, but practical experience showed that some devices don't play well with it, such as the LimeSDR family of devices, which showed far better results when using its native interface.

Related code can be found in the *Transceiver52M/device/uhd/* directory in *osmo-trx.git*.

`osmo-trx-lms` for LimeSuite based Transceivers

This OsmoTRX model uses LimeSuite API and library to drive the device, that is configuring it and reading/writing samples from/to it.

This backend was developed in order to be used together with LimeSDR-USB and LimeSDR-mini devices, due to to the poor results obtained with the UHD backend, and to simplify the stack.

Related code can be found in the *Transceiver52M/device/lms/* directory in *osmo-trx.git*.

`osmo-trx-usrp1` for libusrp based Transceivers

This OsmoTRX model uses the legacy libusrp driver provided in GNU Radio 3.4.2.

As this code was dropped from GNU Radio at some point and was found very difficult to build, some work was done to create a standalone libusrp which can be nowadays found as a separate git repository together with other osmocom git repositories, in https://git.osmocom.org/libusrp/

Related code can be found in the *Transceiver52M/device/usrp1/* directory in *osmo-trx.git*.

The USRPDevice module is basically a driver that reads/writes packets to a USRP with two RFX900 daughterboards, board A is the Tx chain and board B is the Rx chain.

The `radioInterface` module is basically an interface between the transceiver and the USRP. It operates the basestation clock based upon the sample count of received USRP samples. Packets from the USRP are queued and segmented into GSM bursts that are passed up to the transceiver; bursts from the transceiver are passed down to the USRP.

The transceiver basically operates "layer 0" of the GSM stack, performing the modulation, detection, and demodulation of GSM bursts. It communicates with the GSM stack via three UDP sockets, one socket for data, one for control messages, and one socket to pass clocking information. The transceiver contains a priority queue to sort to-be-transmitted bursts, and a filler table to fill in timeslots that do not have bursts in the priority queue. The transceiver tries to stay ahead of the basestation clock, adapting its latency when underruns are reported by the radioInterface/USRP. Received bursts (from the radioInterface) pass through a simple energy detector, a RACH or midamble correlator, and a DFE-based demodulator.

> **Note**
> There's a `SWLOOPBACK` #define statement, where the USRP is replaced with a memory buffer. In this mode, data written to the USRP is actually stored in a buffer, and read commands to the USRP simply pull data from this buffer. This was very useful in early testing, and still may be useful in testing basic Transceiver and radioInterface functionality.

Code Architecture

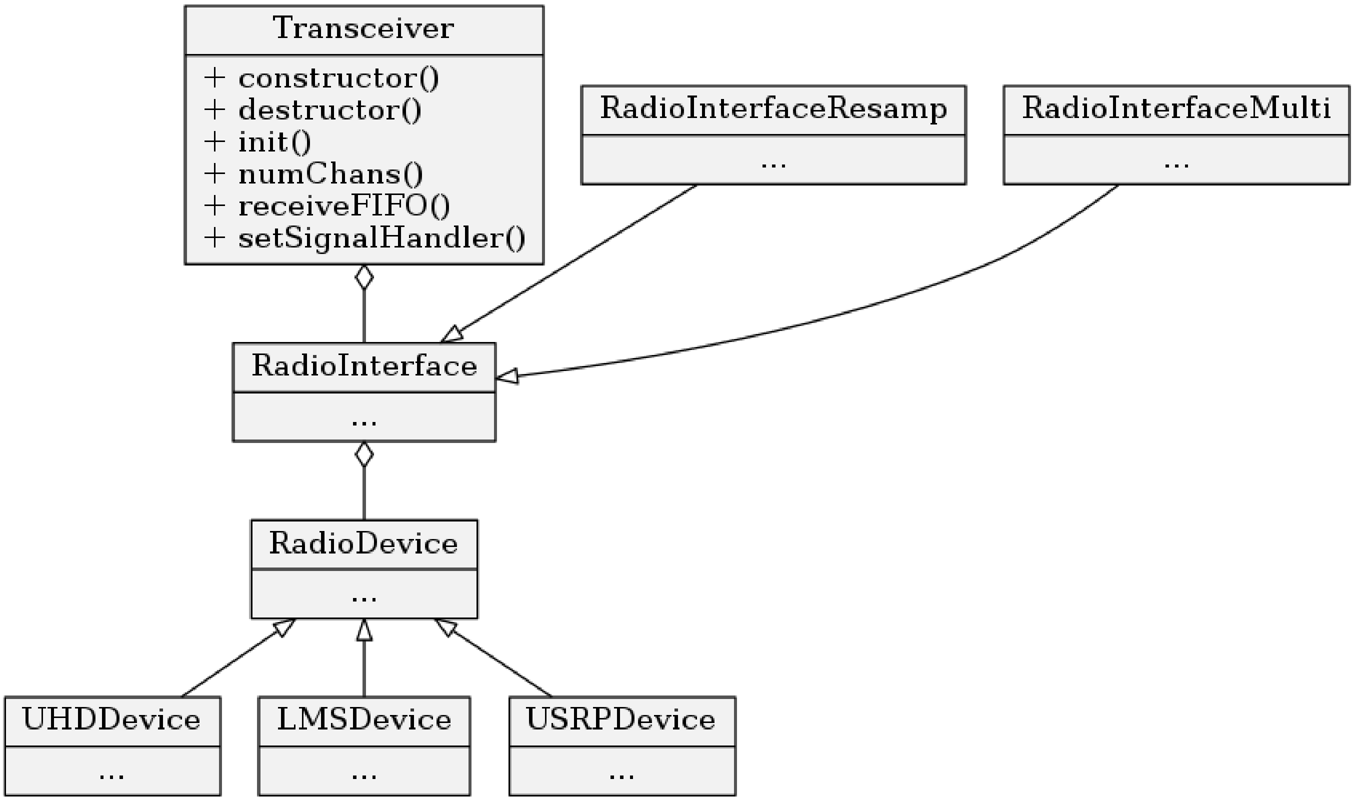

Figure 10: General overview of main OsmoTRX components

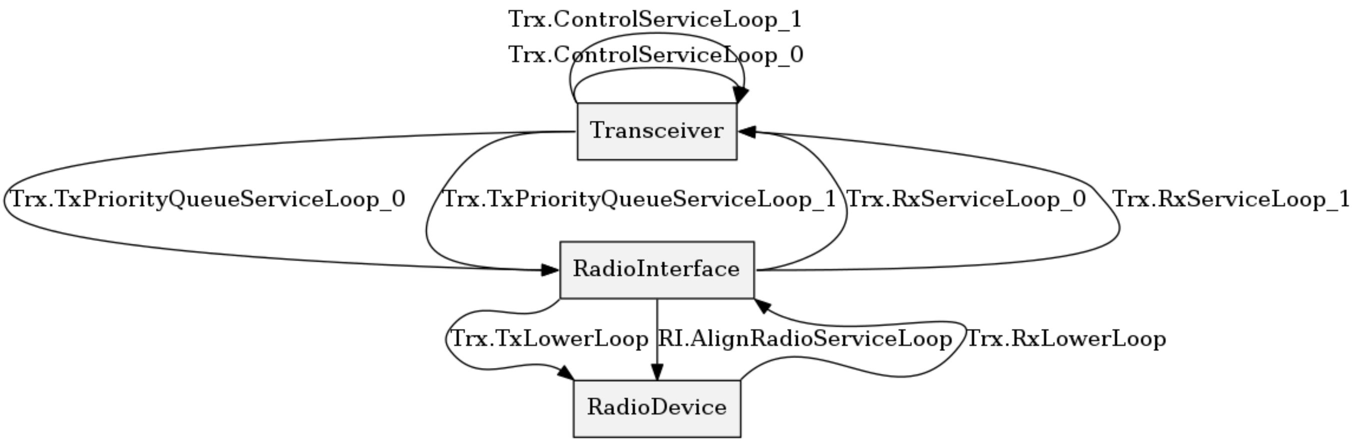

Figure 11: Example of thread architecture with OsmoTRX configured to use 2 logical RF channels (Trx=Transceiver, RI=RadioIface)

Transceiver

The Transceiver is the main component managing the other components running in the OsmoTRX process. There's a unique instance per process.

This class is quite complex from code point of view, as it starts lots of different threads and hence the interaction with this class from the outside is quite limited. Only interaction possible is to:

- `Transceiver()`: Create an instance through its constructor, at this time most configuration is handed to it.
- `init()`: Start running all the threads.
- `receiveFIFO()`: Attach a `radioInterface` channel FIFO in order to use it.
- `setSignalHandler()`: Used to set up a callback to receive certain events asynchronously from the Transceiver. No assumptions can be made about from which thread is the callback being called, which means multi-thread locking precautions may be required in certain cases, similar to usual signal handler processing. One important event received through this path is for instance when the Transceiver detected a fatal error which requires it to stop. Since it cannot stop itself (see destructor below), stopping procedure must be delegated to the user who created the instance.
- `~Transceiver()`: The destructor, which stops all running threads created at `init()` time. Destroying the object is the only way to stop the `Transceiver` completely, and must be called from a thread not managed by the `Transceiver`, otherwise it will deadlock. Usually it is stopped from the main thread, the one that called the constructor during startup.

During `init()` time, `Transceiver` will create a noticeable amount of threads, which may vary depending on the amount of RF channels requested.

Static amount of Threads (1 per `Transceiver` instance):

- `RxLowerLoop`: This thread is responsible for reading bursts from the `RadioInterface`, storing them into its FIFO and sending Clock Indications (Section 18.1) to *osmo-bts_trx*.
- `TxLowerLoop`: Manages pushing bursts from buffers in the FIFO into the `RadioInterface` at expected correct time based on the Transceiver clock.

Dynamic amount of Threads (1 per RF logical channel on the `Transceiver` instance):

- `ControlServiceLoop`: Handles commands from the Per-ARFCN Control Interface socket (Section 18.2). Each thread is responsible for managing one socket related to one ARFCN or which is the same, to one RF logical channel. These are the only threads expected to use the private `start()` and `stop()` methods of the `Transceiver()` class, since those methods don't stop any of the `ControlServiceLoop` threads as they must keep running to handle new commands (for instance, to re-start processing samples with the *POWERON* command).
- `RxServiceLoop`: Each thread of this type pulls bursts from the `RadioInterface` FIFO for one specific logical RF channel and handles it according to the slot and burst correlation type, finally sending proper data over the TRX Manager UDP socket (Section 18).
- `TxPriorityQueueServiceLoop`: Blocks reading from one ARFCN specific TRX Manager UDP socket (Section 18), and fills the `RadioInterface` with it setting clock related information.

RadioInterface

The `RadioInterface` sits between the `Transceiver` and the `RadioDevice`, and provides extra features to the pipe like channelizers, resamplers, Tx/Rx synchronization on some devices, etc.

If the `RadioDevice` it drives requires it (only *USRP1* so far), the `RadioIntercace` will start and manage a thread internally called `AlignRadioServiceLoop` which will align current RX and TX timestamps.

Different features are offered through different `RadioInterface` subclasses which are selected based on configuration and device detected at runtime. Using these features may impact on the amount of CPU required to run the entire pipe.

RadioInterfaceResamp

This subclass of `RadioInterface` is automatically selected when some known specific UHD are to be used, since they require resampling to work properly. Some of this devices are for instance Ettus B100, USRP2 and X3XX models.

RadioInterfaceMulti

This subclass of `RadioInterface` is used when Section 13.2 is requested.

RadioDevice

The `RadioDevice` class is responsible for driving the actual Hardware device. It is actually only an interface, and it is implemented in each backend which in turn becomes a specific OsmoTRX binary, see Section 16.

TRX Manager UDP socket interface

This is the protocol used between `osmo-trx` (the transceiver) and `osmo-bts-trx` (the BTS or core).

Each TRX Manager UDP socket interface represents a single transceiver (ARFCN). Each of these channels is a pair of UDP sockets, one for control (`TRXC`) and one for data (`TRXD`). Additionally, there's a separate global socket managing the Master Clock Interface, shared among all channels.

Given a base port `B` (5700), and a set of channels `0..N`, the ports related to a channel `0 <=X <=N` are:

- The Master clock interface is located on port `P=B`.
- The `TRXC` interface for channel `X` is located on port `P=B+2X+1`
- The `TRXD` interface for channel `X` is located on port `P=B+2X+2`.

The corresponding interface for every socket is at `P+100` on the BTS side.

Indications on the Master Clock Interface

The master clock interface is output only (uplink, from the radio to the BTS). Messages are "indications".

CLOCK gives the current value of the transceiver clock to be used by the BTS. This message is usually sent around once per second (217 GSM frames), but can be sent at any time. The clock value is NOT the current transceiver time. It is a time setting that the BTS should use to give better packet arrival times. The initial clock value is taken randomly, and then increased over time as the transceiver submits downlink packets to the radio.

```
IND CLOCK <totalFrames>
```

TRXC protocol

The per-ARFCN control interface uses a command-response protocol. Each command has a corresponding response. Commands are sent in downlink direction (BTS → TRX), and responses are sent in uplink direction (TRX → BTS). Commands and responses are NULL-terminated ASCII strings.

Every command is structured this way:

```
CMD <cmdtype> [params]
```

The `<cmdtype>` is the actual command. Parameters are optional depending on the commands type.

Every response is of the form:

```
RSP <cmdtype> <status> [result]
```

The `<status>` is 0 for success and a non-zero error code for failure. Successful responses may include results, depending on the command type.

Power Control

POWEROFF shuts off transmitter power and stops the demodulator.

```
CMD POWEROFF
RSP POWEROFF <status>
```

POWERON starts the transmitter and starts the demodulator. Initial power level is very low. This command fails if the transmitter and receiver are not yet tuned. This command fails if the transmit or receive frequency creates a conflict with another ARFCN that is already running. If the transceiver is already on, it answers successfully to this command.

```
CMD POWERON
RSP POWERON <status>
```

SETPOWER sets output power in dB wrt full scale. This command fails if the transmitter and receiver are not running.

```
CMD SETPOWER <dB>
RSP SETPOWER <status> <dB>
```

ADJPOWER adjusts power by the given dB step. Response returns resulting power level wrt full scale. This command fails if the transmitter and receiver are not running.

```
CMD ADJPOWER <dBStep>
RSP ADJPOWER <status> <dBLevel>
```

Tuning Control

RXTUNE tunes the receiver to a given frequency in kHz. This command fails if the receiver is already running. (To re-tune you stop the radio, re-tune, and restart.) This command fails if the transmit or receive frequency creates a conflict with another ARFCN that is already running.

```
CMD RXTUNE <kHz>
RSP RXTUNE <status> <kHz>
```

TXTUNE tunes the transmitter to a given frequency in kHz. This command fails if the transmitter is already running. (To re-tune you stop the radio, re-tune, and restart.) This command fails if the transmit or receive frequency creates a conflict with another ARFCN that is already running.

```
CMD TXTUNE <kHz>
RSP TXTUNE <status> <kHz>
```

Timeslot Control

SETSLOT sets the format of a given uplink timeslot in the ARFCN. The <timeslot> indicates the timeslot of interest. The <chantype> indicates the type of channel that occupies the timeslot. A chantype of zero indicates the timeslot is off.

```
CMD SETSLOT <timeslot> <chantype>
RSP SETSLOT <status> <timeslot> <chantype>
```

Here’s the list of channel combinations and related values (<chantype>):

Table 5: List of channel combinations and related values (<chantype>)

value	Channel Combination
0	Channel is transmitted, but unused
1	TCH/FS

Table 5: (continued)

value	Channel Combination
2	TCH/HS, idle every other slot
3	TCH/HS
4	Downlink: FCCH + SCH + CCCH + BCCH, Uplink: RACH
5	Downlink: FCCH + SCH + CCCH + BCCH + SDCCH/4 + SACCH/4, Uplink: RACH+SDCCH/4
6	Downlink: CCCH+BCCH, Uplink: RACH
7	SDCCH/8 + SACCH/8
8	TCH/F + FACCH/F + SACCH/M
9	TCH/F + SACCH/M
10	TCH/FD + SACCH/MD
11	PBCCH+PCCCH+PDTCH+PACCH+PTCCH
12	PCCCH+PDTCH+PACCH+PTCCH
13	PDTCH+PACCH+PTCCH

TRXD header version negotiation

Messages on DATA interface may have different header formats, defined by a version number, which can be negotiated on the control interface. By default, the Transceiver will use the legacy header version (0).

The header format negotiation can be initiated by the BTS using *SETFORMAT* command. If the requested version is not supported by the transceiver, status code of the response message should indicate a preferred (basically, the latest) version. The format of this message is the following:

```
CMD SETFORMAT <ver_req>
RSP SETFORMAT <ver_resp> <ver_req>
```

where:

- <ver_req> is the requested version (suggested by the BTS),
- <ver_rsp> is either the applied version if matches <ver_req>, or a preferred version if <ver_req> is not supported.

If the transceiver indicates <ver_rsp> different than <ver_req>, the BTS is supposed to re-initiate the version negotiation using the suggested <ver_rsp>. For example:

```
  BTS -> TRX: CMD SETFORMAT 2
  BTS <- TRX: RSP SETFORMAT 1 2

  BTS -> TRX: CMD SETFORMAT 1
  BTS <- TRX: RSP SETFORMAT 1 1
```

If no suitable <ver_rsp> is found, or the <ver_req> is incorrect, the status code in the response shall be `-1`.

As soon as <ver_rsp> matches <ver_req> in the response, the process of negotiation is complete. Changing the header version is supposed to be done before `POWERON`, but can be also done afterwards.

TRXD protocol

Messages on the data interface carry one radio burst per UDP message.

Uplink Data Burst

Uplink data burst message structure differs from version 0 to 1. Basically, version 1 contains an extended header with regards to version 0, and the final padding existence is completely dropped.

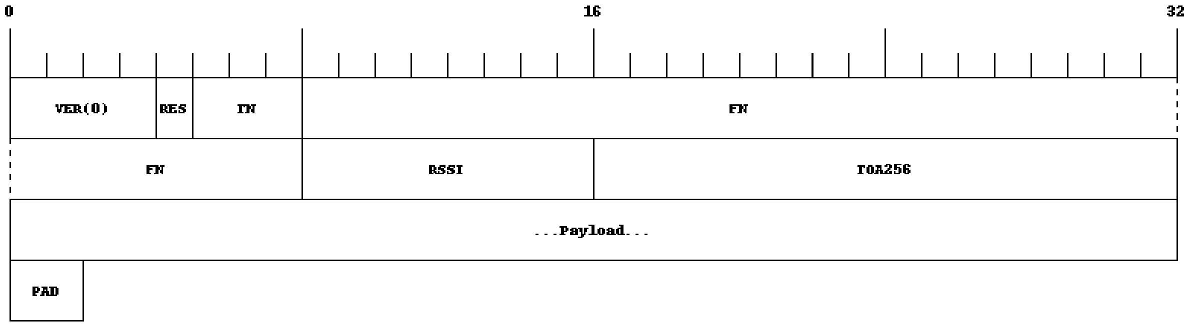

Figure 12: TRXDv0 Uplink data burst message structure

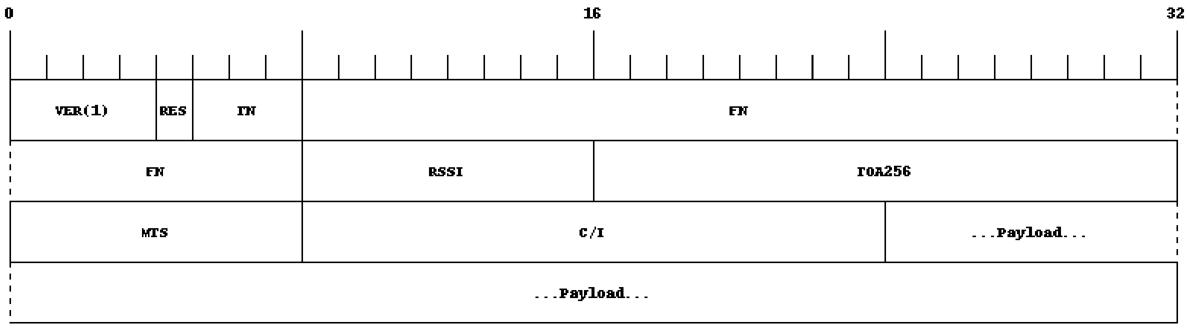

Figure 13: TRXDv1 Uplink data burst message structure

VER: 4 bits
TRXD header version, v0 and v1 are specified so far.

TN: 3 bits
Timeslot number.

RES: 1 bit
Reserved, shall be 0. It can be used in the future to extend the TDMA TN range to (0..15), in case anybody would need to transfer UMTS bursts.

FN: 32 bits (4 bytes)
GSM frame number, big endian.

RSSI: 8 bits (1 byte)
Received Signal Strength Indication in -dBm, encoded without the negative sign.

TOA256: 16 bits (2 bytes)
Timing of Arrival in units of 1/256 of symbol, big endian.

MTS: 8 bits (1 byte)
Contains the Modulation and Training Sequence information. See Section 18.3.1.1 for more information on the encoding.

C/I: 16 bits (2 bytes)
Contains the Carrier-to-Interference ratio in centiBels, big endian. The C/I value is computed from the training sequence of each burst, where the "ideal" training sequence is compared to the actual training sequence and the result expressed in centiBels.

Payload: 148 bytes for GSM, 444 bytes for EDGE
Contains the uplink burst. Unlike the downlink bursts, the uplink bursts are designated using the soft-bits notation, so the receiver can indicate its assurance from 0 to -127 that a given bit is 1, and from 0 to +127 that a given bit is 0. The Viterbi algorithm allows to approximate the original sequence of hard-bits (1 or 0) using these values. Each soft-bit (-127..127) of the burst is encoded as an unsigned value in range (0..255) respectively using the constant shift. This way:

- 0 → definite "0"
- 255 → definite "1".

PAD: 2 bits (optional)
Padding at the end, historical reasons (OpenBTS inheritance). Bits can take any value, but 0 is preferred. Only expected on TRXDv0 headers.

Coding of MTS: Modulation and Training Sequence info

3GPP TS 45.002 version 15.1.0 defines several modulation types, and a few sets of training sequences for each type. The most common are GMSK and 8-PSK (which is used in EDGE).

MTS field structure

```
+-----------------+---------------------------------------+
| 7 6 5 4 3 2 1 0 | bit numbers (value range)             |
+-----------------+---------------------------------------+
| X . . . . . . . | IDLE / nope frame indication (0 or 1) |
+-----------------+---------------------------------------+
| . X X X X . . . | Modulation, TS set number (see below) |
+-----------------+---------------------------------------+
| . . . . . X X X | Training Sequence Code (0..7)         |
+-----------------+---------------------------------------+
```

IDLE / nope frame indication
The bit number 7 (MSB) is set to high when either nothing has been detected, or during IDLE frames, so noise levels can be delivered, and avoid clock gaps on the BTS side. Other bits are ignored, and should be set to low (`0`) in this case.

Modulation and TS set number
GMSK has 4 sets of training sequences (see tables 5.2.3a-d), while 8-PSK (see tables 5.2.3f-g) and the others have 2 sets. Access and Synchronization bursts also have several synchronization sequences.

Modulation and TS set number

```
+-----------------+---------------------------------------+
| 7 6 5 4 3 2 1 0 | bit numbers (value range)             |
+-----------------+---------------------------------------+
| . 0 0 X X . . . | GMSK, 4 TS sets (0..3)                |
+-----------------+---------------------------------------+
| . 0 1 0 X . . . | 8-PSK, 2 TS sets (0..1)               |
+-----------------+---------------------------------------+
| . 0 1 1 X . . . | AQPSK, 2 TS sets (0..1)               |
+-----------------+---------------------------------------+
```

```
| . 1 0 0 X . . . | 16QAM, 2 TS sets (0..1)             |
+-----------------+-------------------------------------+
| . 1 0 1 X . . . | 32QAM, 2 TS sets (0..1)             |
+-----------------+-------------------------------------+
| . 1 1 X X . . . | RESERVED (0)                        |
+-----------------+-------------------------------------+
```

Training Sequence Code
The Training Sequence Code used to decode an Access or a Synchronization burst. This field hence doesn't apply for Normal bursts.

Downlink Data Burst

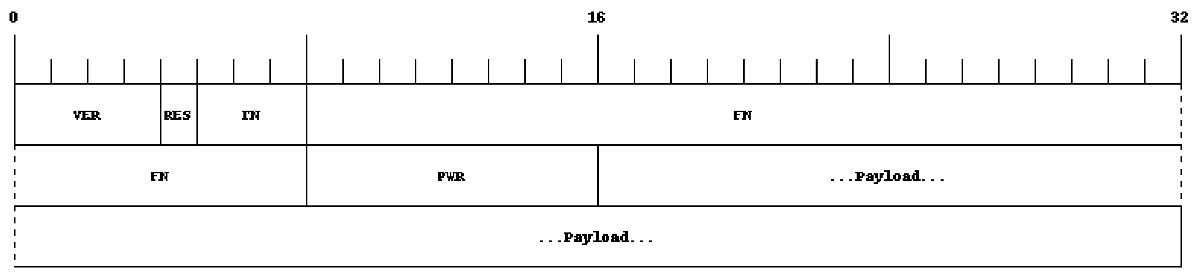

Figure 14: TRXD Downlink data burst message structure

VER: 4 bits
TRXD header version, v0 and v1 are specified so far.

TN: 3 bits
Timeslot number.

RES: 1 bit
Reserved, shall be 0. It can be used in the future to extend the TDMA TN range to (0..15), in case anybody would need to transfer UMTS bursts.

FN: 32 bits (4 bytes)
GSM frame number, big endian.

PWR: 8 bits (1 byte)
Contains the relative (to the full-scale amplitude) transmit power level in dB. The absolute value is set on the control interface.

Payload: 148 bytes for GSM, 444 bytes for EDGE
Contains the downlink burst. Each hard-bit (1 or 0) of the burst is represented using one byte (0x01 or 0x00 respectively).

Glossary

2FF
2nd Generation Form Factor; the so-called plug-in SIM form factor

3FF
: 3rd Generation Form Factor; the so-called microSIM form factor

3GPP
: 3rd Generation Partnership Project

4FF
: 4th Generation Form Factor; the so-called nanoSIM form factor

A Interface
: Interface between BTS and BSC, traditionally over E1 (*3GPP TS 48.008* [3gpp-ts-48-008])

A3/A8
: Algorithm 3 and 8; Authentication and key generation algorithm in GSM and GPRS, typically COMP128v1/v2/v3 or MILENAGE are typically used

A5
: Algorithm 5; Air-interface encryption of GSM; currently only A5/0 (no encryption), A5/1 and A5/3 are in use

Abis Interface
: Interface between BTS and BSC, traditionally over E1 (*3GPP TS 48.058* [3gpp-ts-48-058] and *3GPP TS 52.021* [3gpp-ts-52-021])

ACC
: Access Control Class; every BTS broadcasts a bit-mask of permitted ACC, and only subscribers with a SIM of matching ACC are permitted to use that BTS

AGCH
: Access Grant Channel on Um interface; used to assign a dedicated channel in response to RACH request

AGPL
: GNU Affero General Public License, a copyleft-style Free Software License

ARFCN
: Absolute Radio Frequency Channel Number; specifies a tuple of uplink and downlink frequencies

AUC
: Authentication Center; central database of authentication key material for each subscriber

BCCH
: Broadcast Control Channel on Um interface; used to broadcast information about Cell and its neighbors

BCC
: Base Station Color Code; short identifier of BTS, lower part of BSIC

BTS
: Base Transceiver Station

BSC
: Base Station Controller

BSIC
: Base Station Identity Code; 16bit identifier of BTS within location area

BSSGP
: Base Station Subsystem Gateway Protocol (*3GPP TS 48.018* [3gpp-ts-48-018])

BVCI
: BSSGP Virtual Circuit Identifier

CBCH
: Cell Broadcast Channel; used to transmit Cell Broadcast SMS (SMS-CB)

CC
Call Control; Part of the GSM Layer 3 Protocol

CCCH
Common Control Channel on Um interface; consists of RACH (uplink), BCCH, PCH, AGCH (all downlink)

Cell
A cell in a cellular network, served by a BTS

CEPT
Conférence européenne des administrations des postes et des télécommunications; European Conference of Postal and Telecommunications Administrations.

CGI
Cell Global Identifier comprised of MCC, MNC, LAC and BSIC

CSFB
Circiut-Switched Fall Back; Mechanism for switching from LTE/EUTRAN to UTRAN/GERAN when circuit-switched services such as voice telephony are required.

dB
deci-Bel; relative logarithmic unit

dBm
deci-Bel (milliwatt); unit of measurement for signal strength of radio signals

DHCP
Dynamic Host Configuration Protocol (*IETF RFC 2131* [ietf-rfc2131]

downlink
Direction of messages / signals from the network core towards the mobile phone

DSP
Digital Signal Processor

dvnixload
Tool to program UBL and the Bootloader on a sysmoBTS

EDGE
Enhanced Data rates for GPRS Evolution; Higher-speed improvement of GPRS; introduces 8PSK

EGPRS
Enhanced GPRS; the part of EDGE relating to GPRS services

EIR
Equipment Identity Register; core network element that stores and manages IMEI numbers

ESME
External SMS Entity; an external application interfacing with a SMSC over SMPP

ETSI
European Telecommunications Standardization Institute

FPGA
Field Programmable Gate Array; programmable digital logic hardware

Gb
Interface between PCU and SGSN in GPRS/EDGE network; uses NS, BSSGP, LLC

GERAN
GPRS/EDGE Radio Access Network

GFDL
GNU Free Documentation License; a copyleft-style Documentation License

GGSN
GPRS Gateway Support Node; gateway between GPRS and external (IP) network

GMSK
Gaussian Minimum Shift Keying; modulation used for GSM and GPRS

GPL
GNU General Public License, a copyleft-style Free Software License

Gp
Gp interface between SGSN and GGSN; uses GTP protocol

GPRS
General Packet Radio Service; the packet switched 2G technology

GPS
Global Positioning System; provides a highly accurate clock reference besides the global position

GSM
Global System for Mobile Communications. ETSI/3GPP Standard of a 2G digital cellular network

GSMTAP
GSM tap; pseudo standard for encapsulating GSM protocol layers over UDP/IP for analysis

GSUP
Generic ubscriber Update Protocol. Osmocom-specific alternative to TCAP/MAP

GT
Global Title; an address in SCCP

GTP
GPRS Tunnel Protocol; used between SGSN and GGSN

HLR
Home Location Register; central subscriber database of a GSM network

HNB-GW
Home NodeB Gateway. Entity between femtocells (Home NodeB) and CN in 3G/UMTS.

HPLMN
Home PLMN; the network that has issued the subscriber SIM and has his record in HLR

IE
Information Element

IMEI
International Mobile Equipment Identity; unique 14-digit decimal number to globally identify a mobile device, optionally with a 15th checksum digit

IMEISV
IMEI software version; unique 14-digit decimal number to globally identify a mobile device (same as IMEI) plus two software version digits (total digits: 16)

IMSI
International Mobile Subscriber Identity; 15-digit unique identifier for the subscriber/SIM; starts with MCC/MNC of issuing operator

IP
Internet Protocol (*IETF RFC 791* [?])

IPA
ip.access GSM over IP protocol; used to multiplex a single TCP connection

Iu
: Interface in 3G/UMTS between RAN and CN

IuCS
: Iu interface for circuit-switched domain. Used in 3G/UMTS between RAN and MSC

IuPS
: Iu interface for packet-switched domain. Used in 3G/UMTS between RAN and SGSN

LAC
: Location Area Code; 16bit identifier of Location Area within network

LAPD
: Link Access Protocol, D-Channel (*ITU-T Q.921* [itu-t-q921])

LAPDm
: Link Access Protocol Mobile (*3GPP TS 44.006* [3gpp-ts-44-006])

LLC
: Logical Link Control; GPRS protocol between MS and SGSN (*3GPP TS 44.064* [3gpp-ts-44-064])

Location Area
: Location Area; a geographic area containing multiple BTS

LU
: Location Updating; can be of type IMSI-Attach or Periodic. Procedure that indicates a subscriber's physical presence in a given radio cell.

M2PA
: MTP2 Peer-to-Peer Adaptation; a SIGTRAN Variant (*RFC 4165* [ietf-rfc4165])

M2UA
: MTP2 User Adaptation; a SIGTRAN Variant (*RFC 3331* [ietf-rfc3331])

M3UA
: MTP3 User Adaptation; a SIGTRAN Variant (*RFC 4666* [ietf-rfc4666])

MCC
: Mobile Country Code; unique identifier of a country, e.g. 262 for Germany

MFF
: Machine-to-Machine Form Factor; a SIM chip package that is soldered permanently onto M2M device circuit boards.

MGW
: Media Gateway

MM
: Mobility Management; part of the GSM Layer 3 Protocol

MNC
: Mobile Network Code; identifies network within a country; assigned by national regulator

MNCC
: Mobile Network Call Control; Unix domain socket based Interface between MSC and external call control entity like osmo-sip-connector

MNO
: Mobile Network Operator; operator with physical radio network under his MCC/MNC

MO
: Mobile Originated. Direction from Mobile (MS/UE) to Network

MS
: Mobile Station; a mobile phone / GSM Modem

MSC
Mobile Switching Center; network element in the circuit-switched core network

MSISDN
Mobile Subscriber ISDN Number; telephone number of the subscriber

MT
Mobile Terminated. Direction from Network to Mobile (MS/UE)

MTP
Message Transfer Part; SS7 signaling protocol (*ITU-T Q.701* [itu-t-q701])

MVNO
Mobile Virtual Network Operator; Operator without physical radio network

NCC
Network Color Code; assigned by national regulator

NITB
Network In The Box; combines functionality traditionally provided by BSC, MSC, VLR, HLR, SMSC functions; see OsmoNITB

NSEI
NS Entity Identifier

NVCI
NS Virtual Circuit Identifier

NWL
Network Listen; ability of some BTS to receive downlink from other BTSs

NS
Network Service; protocol on Gb interface (*3GPP TS 48.016* [3gpp-ts-48-016])

OCXO
Oven Controlled Crystal Oscillator; very high precision oscillator, superior to a VCTCXO

OML
Operation & Maintenance Link (ETSI/*3GPP TS 52.021* [3gpp-ts-52-021])

OpenBSC
Open Source implementation of GSM network elements, specifically OsmoBSC, OsmoNITB, OsmoSGSN

OpenGGSN
Open Source implementation of a GPRS Packet Control Unit

OpenVPN
Open-Source Virtual Private Network; software employed to establish encrypted private networks over untrusted public networks

Osmocom
Open Source MObile COMmunications; collaborative community for implementing communications protocols and systems, including GSM, GPRS, TETRA, DECT, GMR and others

OsmoBSC
Open Source implementation of a GSM Base Station Controller

OsmoNITB
Open Source implementation of a GSM Network In The Box, combines functionality traditionally provided by BSC, MSC, VLR, HLR, AUC, SMSC

OsmoSGSN
Open Source implementation of a Serving GPRS Support Node

OsmoPCU
Open Source implementation of a GPRS Packet Control Unit

OTA
Over-The-Air; Capability of operators to remotely reconfigure/reprogram ISM/USIM cards

PC
Point Code; an address in MTP

PCH
Paging Channel on downlink Um interface; used by network to page an MS

PCU
Packet Control Unit; used to manage Layer 2 of the GPRS radio interface

PDCH
Packet Data Channel on Um interface; used for GPRS/EDGE signalling + user data

PIN
Personal Identification Number; a number by which the user authenticates to a SIM/USIM or other smart card

PLMN
Public Land Mobile Network; specification language for a single GSM network

PUK
PIN Unblocking Code; used to unblock a blocked PIN (after too many wrong PIN attempts)

RAC
Routing Area Code; 16bit identifier for a Routing Area within a Location Area

RACH
Random Access Channel on uplink Um interface; used by MS to request establishment of a dedicated channel

RAM
Remote Application Management; Ability to remotely manage (install, remove) Java Applications on SIM/USIM Card

RF
Radio Frequency

RFM
Remote File Management; Ability to remotely manage (write, read) files on a SIM/USIM card

Roaming
Procedure in which a subscriber of one network is using the radio network of another network, often in different countries; in some countries national roaming exists

Routing Area
Routing Area; GPRS specific sub-division of Location Area

RR
Radio Resources; Part of the GSM Layer 3 Protocol

RSL
Radio Signalling Link (*3GPP TS 48.058* [3gpp-ts-48-058])

RTP
Real-Time Transport Protocol (*IETF RFC 3550* [ietf-rfc3550]); Used to transport audio/video streams over UDP/IP

SACCH
Slow Associate Control Channel on Um interface; bundled to a TCH or SDCCH, used for signalling in parallel to active dedicated channel

SCCP
Signaling Connection Control Part; SS7 signaling protocol (*ITU-T Q.711* [itu-t-q711])

SDCCH
Slow Dedicated Control Channel on Um interface; used for signalling and SMS transport in GSM

SDK
Software Development Kit

SGs
Interface between MSC (GSM/UMTS) and MME (LTE/EPC) to facilitate CSFB and SMS.

SGSN
Serving GPRS Support Node; Core network element for packet-switched services in GSM and UMTS.

SIGTRAN
Signaling Transport over IP (*IETF RFC 2719* [ietf-rfc2719])

SIM
Subscriber Identity Module; small chip card storing subscriber identity

Site
A site is a location where one or more BTSs are installed, typically three BTSs for three sectors

SMPP
Short Message Peer-to-Peer; TCP based protocol to interface external entities with an SMSC

SMSC
Short Message Service Center; store-and-forward relay for short messages

SS7
Signaling System No. 7; Classic digital telephony signaling system

SS
Supplementary Services; query and set various service parameters between subscriber and core network (e.g. USSD, 3rd-party calls, hold/retrieve, advice-of-charge, call deflection)

SSH
Secure Shell; *IETF RFC 4250* [ietf-rfc4251] to 4254

SSN
Sub-System Number; identifies a given SCCP Service such as MSC, HLR

STP
Signaling Transfer Point; A Router in SS7 Networks

SUA
SCCP User Adaptation; a SIGTRAN Variant (*RFC 3868* [ietf-rfc3868])

syslog
System logging service of UNIX-like operating systems

System Information
A set of downlink messages on the BCCH and SACCH of the Um interface describing properties of the cell and network

TCH
Traffic Channel; used for circuit-switched user traffic (mostly voice) in GSM

TCP
Transmission Control Protocol; (*IETF RFC 793* [ietf-rfc793])

TFTP
Trivial File Transfer Protocol; (*IETF RFC 1350* [ietf-rfc1350])

TRX
Transceiver; element of a BTS serving a single carrier

TS
Technical Specification

u-Boot
Boot loader used in various embedded systems

UBI
An MTD wear leveling system to deal with NAND flash in Linux

UBL
Initial bootloader loaded by the TI Davinci SoC

UDP
User Datagram Protocol (*IETF RFC 768* [ietf-rfc768])

UICC
Universal Integrated Chip Card; A smart card according to *ETSI TR 102 216* [etsi-tr102216]

Um interface
U mobile; Radio interface between MS and BTS

uplink
Direction of messages: Signals from the mobile phone towards the network

USIM
Universal Subscriber Identity Module; application running on a UICC to provide subscriber identity for UMTS and GSM networks

USSD
Unstructured Supplementary Service Data; textual dialog between subscriber and core network, e.g. **100 → Your extension is 1234*

VCTCXO
Voltage Controlled, Temperature Compensated Crystal Oscillator; a precision oscillator, superior to a classic crystal oscillator, but inferior to an OCXO

VLR
Visitor Location Register; volatile storage of attached subscribers in the MSC

VPLMN
Visited PLMN; the network in which the subscriber is currently registered; may differ from HPLMN when on roaming

VTY
Virtual TeletYpe; a textual command-line interface for configuration and introspection, e.g. the OsmoBSC configuration file as well as its telnet link on port 4242

Osmocom TCP/UDP Port Numbers

The Osmocom GSM system utilizes a variety of TCP/IP based protocols. The table below provides a reference as to which port numbers are used by which protocol / interface.

Table 6: TCP/UDP port numbers

L4 Protocol	Port Number	Purpose	Software
UDP	2427	MGCP GW	osmo-bsc_mgcp, osmo-mgw
TCP	2775	SMPP (SMS interface for external programs)	osmo-nitb
TCP	3002	A-bis/IP OML	osmo-bts, osmo-bsc, osmo-nitb
TCP	3003	A-bis/IP RSL	osmo-bts, osmo-bsc, osmo-nitb

Table 6: (continued)

L4 Protocol	Port Number	Purpose	Software
TCP	4236	Control Interface	osmo-trx
TCP	4237	telnet (VTY)	osmo-trx
TCP	4238	Control Interface	osmo-bts
TCP	4239	telnet (VTY)	osmo-stp
TCP	4240	telnet (VTY)	osmo-pcu
TCP	4241	telnet (VTY)	osmo-bts
TCP	4242	telnet (VTY)	osmo-nitb, osmo-bsc, cellmgr-ng
TCP	4243	telnet (VTY)	osmo-bsc_mgcp, osmo-mgw
TCP	4244	telnet (VTY)	osmo-bsc_nat
TCP	4245	telnet (VTY)	osmo-sgsn
TCP	4246	telnet (VTY)	osmo-gbproxy
TCP	4247	telnet (VTY)	OsmocomBB
TCP	4249	Control Interface	osmo-nitb, osmo-bsc
TCP	4250	Control Interface	osmo-bsc_nat
TCP	4251	Control Interface	osmo-sgsn
TCP	4252	telnet (VTY)	sysmobts-mgr
TCP	4253	telnet (VTY)	osmo-gtphub
TCP	4254	telnet (VTY)	osmo-msc
TCP	4255	Control Interface	osmo-msc
TCP	4256	telnet (VTY)	osmo-sip-connector
TCP	4257	Control Interface	osmo-ggsn, ggsn (OpenGGSN)
TCP	4258	telnet (VTY)	osmo-hlr
TCP	4259	Control Interface	osmo-hlr
TCP	4260	telnet (VTY)	osmo-ggsn
TCP	4261	telnet (VTY)	osmo-hnbgw
TCP	4262	Control Interface	osmo-hnbgw
TCP	4263	Control Interface	osmo-gbproxy
TCP	4264	telnet (VTY)	osmo-cbc
TCP	4265	Control Interface	osmo-cbc
TCP	4266	D-GSM MS Lookup: mDNS serve	osmo-hlr
TCP	4267	Control Interface	osmo-mgw
UDP	4729	GSMTAP	Almost every osmocom project
TCP	5000	A/IP	osmo-bsc, osmo-bsc_nat
UDP	23000	GPRS-NS over IP default port	osmo-pcu, osmo-sgsn, osmo-gbproxy

Bibliography / References

References

[1] [osmobts-abis-spec] Neels Hofmeyr & Harald Welte. OsmoBTS Abis Protocol Specification. http://ftp.osmocom.org/docs/latest/osmobts-abis.pdf

[2] [userman-osmobts] Osmocom Project: OsmoBTS User Manual. http://ftp.osmocom.org/docs/latest/osmobts-usermanual.pdf

[3] [vty-ref-osmobts] Osmocom Project: OsmoBTS VTY Reference Manual. http://ftp.osmocom.org/docs/latest/-osmobts-vty-reference.pdf

[4] [userman-osmobsc] Osmocom Project: OsmoBSC User Manual. http://ftp.osmocom.org/docs/latest/osmobsc-usermanual.pdf

[5] [vty-ref-osmobsc] Osmocom Project: OsmoBSC VTY Reference Manual. http://ftp.osmocom.org/docs/latest/-osmobsc-vty-reference.pdf

[6] [userman-osmomsc] Osmocom Project: OsmoMSC User Manual. http://ftp.osmocom.org/docs/latest/-osmomsc-usermanual.pdf

[7] [vty-ref-osmomsc] Osmocom Project: OsmoMSC VTY Reference Manual. http://ftp.osmocom.org/docs/-latest/osmomsc-vty-reference.pdf

[8] [userman-osmohlr] Osmocom Project: OsmoHLR User Manual. http://ftp.osmocom.org/docs/latest/osmohlr-usermanual.pdf

[9] [vty-ref-osmohlr] Osmocom Project: OsmoHLR VTY Reference Manual. http://ftp.osmocom.org/docs/latest/-osmohlr-vty-reference.pdf

[10] [userman-osmopcu] Osmocom Project: OsmoPCU User Manual. http://ftp.osmocom.org/docs/latest/osmopcu-usermanual.pdf

[11] [vty-ref-osmopcu] Osmocom Project: OsmoPCU VTY Reference Manual. http://ftp.osmocom.org/docs/latest/-osmopcu-vty-reference.pdf

[12] [userman-osmonitb] Osmocom Project: OsmoNITB User Manual. http://ftp.osmocom.org/docs/latest/-osmonitb-usermanual.pdf

[13] [vty-ref-osmonitb] Osmocom Project: OsmoNITB VTY Reference Manual. http://ftp.osmocom.org/docs/-latest/osmonitb-vty-reference.pdf

[14] [userman-osmosgsn] Osmocom Project: OsmoSGSN User Manual. http://ftp.osmocom.org/docs/latest/-osmosgsn-usermanual.pdf

[15] [vty-ref-osmosgsn] Osmocom Project: OsmoSGSN VTY Reference Manual. http://ftp.osmocom.org/docs/-latest/osmonitb-vty-reference.pdf

[16] [userman-osmoggsn] Osmocom Project: OpenGGSN User Manual. http://ftp.osmocom.org/docs/latest/-osmoggsn-usermanual.pdf

[17] [vty-ref-osmoggsn] Osmocom Project: OsmoGGSN VTY Reference Manual. http://ftp.osmocom.org/docs/-latest/osmoggsn-vty-reference.pdf

[18] [3gpp-ts-23-048] 3GPP TS 23.048: Security mechanisms for the (U)SIM application toolkit; Stage 2 http://www.3gpp.org/DynaReport/23048.htm

[19] [3gpp-ts-24-007] 3GPP TS 24.007: Mobile radio interface signalling layer 3; General Aspects http://www.3gpp.org/DynaReport/24007.htm

[20] [3gpp-ts-24-008] 3GPP TS 24.008: Mobile radio interface Layer 3 specification; Core network protocols; Stage 3. http://www.3gpp.org/dynareport/24008.htm

[21] [3gpp-ts-31-101] 3GPP TS 31.101: UICC-terminal interface; Physical and logical characteristics http://www.3gpp.org/DynaReport/31101.htm

[22] [3gpp-ts-31-102] 3GPP TS 31.102: Characteristics of the Universal Subscriber Identity Module (USIM) application http://www.3gpp.org/DynaReport/31102.htm

[23] [3gpp-ts-31-111] 3GPP TS 31.111: Universal Subscriber Identity Module (USIM) Application Toolkit (USAT) http://www.3gpp.org/DynaReport/31111.htm

[24] [3gpp-ts-31-115] 3GPP TS 31.115: Secured packet structure for (Universal) Subscriber Identity Module (U)SIM Toolkit applications http://www.3gpp.org/DynaReport/31115.htm

[25] [3gpp-ts-31-116] 3GPP TS 31.116: Remote APDU Structure for (U)SIM Toolkit applications http://www.3gpp.org/DynaReport/31116.htm

[26] [3gpp-ts-35-205] 3GPP TS 35.205: 3G Security; Specification of the MILENAGE algorithm set: General

[27] [3gpp-ts-35-206] 3GPP TS 35.206: 3G Security; Specification of the MILENAGE algorithm set: Algorithm specification http://www.3gpp.org/DynaReport/35206.htm

[28] [3gpp-ts-44-006] 3GPP TS 44.006: Mobile Station - Base Station System (MS - BSS) interface; Data Link (DL) layer specification http://www.3gpp.org/DynaReport/44006.htm

[29] [3gpp-ts-44-018] 3GPP TS 44.018: Mobile radio interface layer 3 specification; Radio Resource Control (RRC) protocol http://www.3gpp.org/DynaReport/44018.htm

[30] [3gpp-ts-44-064] 3GPP TS 44.064: Mobile Station - Serving GPRS Support Node (MS-SGSN); Logical Link Control (LLC) Layer Specification http://www.3gpp.org/DynaReport/44064.htm

[31] [3gpp-ts-48-008] 3GPP TS 48.008: Mobile Switching Centre - Base Station system (MSC-BSS) interface; Layer 3 specification http://www.3gpp.org/DynaReport/48008.htm

[32] [3gpp-ts-48-016] 3GPP TS 48.016: General Packet Radio Service (GPRS); Base Station System (BSS) - Serving GPRS Support Node (SGSN) interface; Network service http://www.3gpp.org/DynaReport/48016.htm

[33] [3gpp-ts-48-018] 3GPP TS 48.018: General Packet Radio Service (GPRS); Base Station System (BSS) - Serving GPRS Support Node (SGSN); BSS GPRS protocol (BSSGP) http://www.3gpp.org/DynaReport/48018.htm

[34] [3gpp-ts-48-056] 3GPP TS 48.056: Base Station Controller - Base Transceiver Station (BSC - BTS) interface; Layer 2 specification http://www.3gpp.org/DynaReport/48056.htm

[35] [3gpp-ts-48-058] 3GPP TS 48.058: Base Station Controller - Base Transceiver Station (BSC - BTS) Interface; Layer 3 specification http://www.3gpp.org/DynaReport/48058.htm

[36] [3gpp-ts-51-011] 3GPP TS 51.011: Specification of the Subscriber Identity Module - Mobile Equipment (SIM-ME) interface

[37] [3gpp-ts-51-014] 3GPP TS 51.014: Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface http://www.3gpp.org/DynaReport/51014.htm

[38] [3gpp-ts-52-021] 3GPP TS 52.021: Network Management (NM) procedures and messages on the A-bis interface http://www.3gpp.org/DynaReport/52021.htm

[39] [etsi-tr102216] ETSI TR 102 216: Smart cards http://www.etsi.org/deliver/etsi_tr/102200_102299/102216/03.00.00_60/tr_102216v030000p.pdf

[40] [etsi-ts102221] ETSI TS 102 221: Smart Cards; UICC-Terminal interface; Physical and logical characteristics http://www.etsi.org/deliver/etsi_ts/102200_102299/102221/13.01.00_60/ts_102221v130100p.pdf

[41] [etsi-ts101220] ETSI TS 101 220: Smart Cards; ETSI numbering system for telecommunication application providers http://www.etsi.org/deliver/etsi_ts/101200_101299/101220/12.00.00_60/ts_101220v120000p.pdf

[42] [ietf-rfc768] IETF RFC 768: Internet Protocol https://tools.ietf.org/html/rfc791

[43] [ietf-rfc793] IETF RFC 793: Transmission Control Protocol https://tools.ietf.org/html/rfc793

[44] [ietf-rfc1035] IETF RFC 1035: Domain Names - Implementation and Specification https://tools.ietf.org/html/rfc1035

[45] [ietf-rfc1350] IETF RFC 1350: Trivial File Transfer Protool https://tools.ietf.org/html/rfc1350

[46] [ietf-rfc2131] IETF RFC 2131: Dynamic Host Configuration Protocol https://tools.ietf.org/html/rfc2131

[47] [ietf-rfc2719] IETF RFC 2719: Signal Transport over IP https://tools.ietf.org/html/rfc2719

[48] [ietf-rfc3331] IETF RFC 3331: Message Transfer Part 2 User Adaptation Layer https://tools.ietf.org/html/rfc3331

[49] [ietf-rfc3550] IETF RFC 3550: RTP: A Transport protocol for Real-Time Applications https://tools.ietf.org/html/rfc3550

[50] [ietf-rfc3596] IETF RFC 3596: DNS Extensions to Support IP Version 6 https://tools.ietf.org/html/rfc3596

[51] [ietf-rfc3868] IETF RFC 3868: SCCP User Adaptation Layer https://tools.ietf.org/html/rfc3868

[52] [ietf-rfc4165] IETF RFC 4165: Message Transfer Part 2 Peer-to-Peeer Adaptation Layer https://tools.ietf.org/-html/rfc4165

[53] [ietf-rfc4251] IETF RFC 4251: The Secure Shell (SSH) Protocol Architecture https://tools.ietf.org/html/-rfc4251

[54] [ietf-rfc4666] IETF RFC 4666: Message Transfer Part 3 User Adaptation Layer https://tools.ietf.org/html/-rfc4666

[55] [ietf-rfc5771] IETF RFC 5771: IANA Guidelines for IPv4 Multicast Address Assignments https://tools.ietf.org/html/rfc5771

[56] [itu-t-q701] ITU-T Q.701: Functional Description of the Message Transfer Part (MTP) https://www.itu.int/rec/-T-REC-Q.701/en/

[57] [itu-t-q711] ITU-T Q.711: Functional Description of the Signalling Connection Control Part https://www.itu.int/rec/T-REC-Q.711/en/

[58] [itu-t-q713] ITU-T Q.713: Signalling connection control part formats and codes https://www.itu.int/rec/T-REC-Q.713/en/

[59] [itu-t-q714] ITU-T Q.714: Signalling connection control part procedures https://www.itu.int/rec/T-REC-Q.714/en/

[60] [itu-t-q921] ITU-T Q.921: ISDN user-network interface - Data link layer specification https://www.itu.int/rec/-T-REC-Q.921/en

[61] [smpp-34] SMPP Develoepers Forum. Short Message Peer-to-Peer Protocol Specification v3.4 http://docs.nimta.com/SMPP_v3_4_Issue1_2.pdf

[62] [gnu-agplv3] Free Software Foundation. GNU Affero General Public License. http://www.gnu.org/licenses/-agpl-3.0.en.html

[63] [freeswitch_pbx] FreeSWITCH SIP PBX https://freeswitch.org

GNU Free Documentation License

Version 1.3, 3 November 2008

PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The “Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Section whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not Transparent is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, Title Page means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section Section C.4.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

a. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

b. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

c. State on the Title Page the name of the publisher of the Modified Version, as the publisher.

d. Preserve all the copyright notices of the Document.

e. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

f. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

g. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

h. Include an unaltered copy of this License.

i. Preserve the section Entitled “History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

j. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

k. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

l. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

m. Delete any section Entitled “Endorsements”. Such a section may not be included in the [?].

n. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Sections.

o. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Section and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled “History” in the various original documents, forming one section Entitled “History”; likewise combine any sections Entitled “Acknowledgements”, and any sections Entitled “Dedications”. You must delete all sections Entitled “Endorsements”.

COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an “aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled “Acknowledgements”, “Dedications”, or “History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (c)  YEAR  YOUR NAME.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.