

sysmocom

sysmocom - s.f.m.c. GmbH



osmocom

OsmoBTS User Manual

by Harald Welte

Copyright © 2016 sysmocom - s.f.m.c. GmbH

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 the Invariant Sections being just 'Foreword', 'Acknowledgements' and 'Preface', with no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

| HISTORY | | | |
|---------|--------------|---|------|
| NUMBER | DATE | DESCRIPTION | NAME |
| 1 | January 2016 | Initial version, reflecting OsmoBTS master branch as on FIXME (commit FIXME). | HW |

Contents

| | |
|---|----------|
| 1 Foreword | 1 |
| 1.1 Acknowledgements | 1 |
| 1.2 Endorsements | 2 |
| 2 Preface | 2 |
| 2.1 FOSS lives by contribution! | 2 |
| 2.2 Osmocom and sysmocom | 2 |
| 2.3 Corrections | 3 |
| 2.4 Legal disclaimers | 3 |
| 2.4.1 Spectrum License | 3 |
| 2.4.2 Software License | 3 |
| 2.4.3 Trademarks | 3 |
| 2.4.4 Liability | 3 |
| 2.4.5 Documentation License | 4 |
| 3 Introduction | 4 |
| 3.1 Required Skills | 4 |
| 3.2 Getting assistance | 4 |
| 4 Overview | 4 |
| 4.1 About this manual | 4 |
| 4.2 About OsmoBTS | 5 |
| 4.3 Credits | 5 |
| 4.4 OsmoBTS in the Osmocom GSM network architecture | 5 |
| 5 Abis/IP Interface | 6 |
| 5.1 A-bis Operation & Maintenance Link | 6 |
| 5.2 A-bis Radio Signalling Link | 6 |
| 5.3 Locate Abis/IP based BTS | 6 |
| 5.3.1 abisip-find | 6 |
| 5.4 Deploying a new nanoBTS | 7 |
| 5.4.1 ipaccess-config | 7 |
| 6 OsmoBTS Interfaces | 7 |
| 6.1 OsmoBTS Abis/IP Interface | 8 |
| 6.2 bts_model specific PHY interface | 8 |
| 6.3 OsmoBTS VTY Interface | 8 |
| 6.4 OsmoBTS Control Interface | 8 |
| 6.4.1 trx.N.thermal-attenuation | 8 |
| 6.5 OsmoBTS GSMTAP Interface | 9 |
| 6.6 OsmoBTS PCU Socket Interface | 9 |

| | | |
|-----------|---|-----------|
| 7 | Control interface | 10 |
| 7.1 | thermal-attenuation | 10 |
| 8 | Osmocom Counters | 10 |
| 8.1 | Osmo Counters (deprecated) | 10 |
| 8.2 | Rate Counters | 10 |
| 8.3 | Stat Item | 11 |
| 8.4 | Statistic Levels | 11 |
| 8.4.1 | Global | 11 |
| 8.4.2 | Peer | 11 |
| 8.4.3 | Subscriber | 11 |
| 8.5 | Stats Reporter | 11 |
| 8.5.1 | Configuring a stats reporter | 11 |
| 9 | Counters | 12 |
| 9.1 | Rate Counters | 12 |
| 10 | Osmo Stat Items | 14 |
| 11 | Osmo Counters | 14 |
| 12 | The Osmocom VTY Interface | 14 |
| 12.1 | Accessing the telnet VTY | 15 |
| 12.2 | VTY Nodes | 15 |
| 12.3 | Interactive help | 16 |
| 12.3.1 | The question-mark (?) command | 16 |
| 12.3.2 | TAB completion | 17 |
| 12.3.3 | The list command | 17 |
| 12.3.4 | The attribute system | 19 |
| 13 | libosmocore Logging System | 20 |
| 13.1 | Log categories | 21 |
| 13.2 | Log levels | 21 |
| 13.3 | Log printing options | 21 |
| 13.4 | Log filters | 22 |
| 13.5 | Log targets | 22 |
| 13.5.1 | Logging to the VTY | 22 |
| 13.5.2 | Logging to the ring buffer | 23 |
| 13.5.3 | Logging via gsmmap | 23 |
| 13.5.4 | Logging to a file | 24 |
| 13.5.5 | Logging to syslog | 25 |
| 13.5.6 | Logging to stderr | 25 |

| | |
|--|-----------|
| 14 BTS Configuration | 25 |
| 14.1 Command Line Options | 25 |
| 14.1.1 SYNOPSIS | 26 |
| 14.1.2 OPTIONS | 26 |
| 14.2 Configuration using the VTY | 26 |
| 14.2.1 Required BTS/TRX configuration | 27 |
| 14.2.2 Configuring GSMTAP tracing | 27 |
| 14.2.3 Configuring power ramping | 28 |
| 14.2.4 Running multiple instances | 28 |
| 15 Support for Dynamic Timeslots (TCH/F, TCH/H, PDCH) | 29 |
| 16 OsmoBTS hardware support | 29 |
| 17 osmo-bts-sysmo for sysmocom sysmoBTS | 29 |
| 17.1 osmo-bts-sysmo specific command line arguments | 30 |
| 17.2 osmo-bts-sysmo specific VTY commands | 30 |
| 17.2.1 at the <i>SHOW</i> node | 30 |
| 17.2.1.1 show trx 0 clock-source | 30 |
| 17.2.1.2 show trx 0 dsp-trace-flags | 30 |
| 17.2.1.3 trx 0 dsp-trace-flag | 30 |
| 17.2.2 at the <i>ENABLE</i> node | 30 |
| 17.2.2.1 trx 0 tx-power <-110-100> | 30 |
| 17.2.2.2 trx 0 rf-clock-info reset | 30 |
| 17.2.2.3 trx 0 rf-clock-info correct | 30 |
| 17.2.3 at the <i>PHY instance</i> node | 30 |
| 17.2.4 clock-calibration eeprom | 30 |
| 17.2.5 clock-calibration default | 31 |
| 17.2.6 clock-calibration <-4095-4095> | 31 |
| 17.2.7 clock-source (tcxo ocxo ext gps) | 31 |
| 17.2.8 trx-calibration-path PATH | 31 |
| 17.3 osmo-bts-sysmo specific control interface commands | 31 |
| 17.3.1 trx.0.clock-info | 31 |
| 17.3.2 trx.0.clock-correction | 32 |
| 18 osmo-bts-trx for OsmoTRX | 32 |
| 18.1 osmo-bts-trx specific VTY commands | 32 |
| 18.1.1 at the <i>SHOW</i> node | 32 |
| 18.1.1.1 show transceivers | 32 |
| 18.1.2 at the <i>PHY</i> configuration node | 32 |

| | | |
|-----------|---|-----------|
| 18.1.2.1 | osmotrx ip HOST | 32 |
| 18.1.2.2 | osmotrx ip (local remote) A.B.C.D | 32 |
| 18.1.2.3 | osmotrx base-port (local remote) <0-65535> | 33 |
| 18.1.2.4 | osmotrx fn-advance <0-30> | 33 |
| 18.1.2.5 | osmotrx rts-advance <0-30> | 33 |
| 18.1.2.6 | osmotrx rx-gain <0-50> | 33 |
| 18.1.2.7 | osmotrx tx-attenuation <0-50> | 33 |
| 18.1.2.8 | osmotrx tx-attenuation oml | 33 |
| 18.1.3 | at the <i>PHY Instance</i> configuration node | 34 |
| 18.1.3.1 | slotmask (1 0) (1 0) (1 0) (1 0) (1 0) (1 0) (1 0) (1 0) | 34 |
| 18.1.3.2 | osmotrx maxdly <0-31> | 34 |
| 19 | osmo-bts-octphy for Octasic OCTPHY-2G | 34 |
| 20 | osmo-bts-litecell1115 for Nutaq/Nuran LiteCell 1.5 | 34 |
| 20.1 | osmo-bts-trx specific VTY commands | 34 |
| 20.1.1 | at the <i>SHOW</i> node | 34 |
| 20.1.1.1 | show phy <0-255> system-information | 34 |
| 20.1.1.2 | show phy <0-255> rf-port-stats <0-1> | 35 |
| 20.1.1.3 | show phy <0-255> clk-sync-stats | 35 |
| 20.1.2 | at the <i>PHY</i> configuration node | 35 |
| 20.1.2.1 | octphy hw-addr HWADDR | 35 |
| 20.1.2.2 | octphy net-device NAME | 35 |
| 20.1.2.3 | octphy rf-port-index <0-255> | 35 |
| 20.1.2.4 | octphy rx-gain <0-73> | 35 |
| 20.1.2.5 | octphy tx-attenuation <0-359> | 35 |
| 21 | osmo-bts-virtual for Virtual Um Interface | 35 |
| 21.1 | osmo-bts-virtual specific VTY commands | 36 |
| 21.1.1 | at the <i>PHY</i> config node | 36 |
| 21.1.1.1 | virtual-um net-device NETDEV | 36 |
| 21.1.1.2 | virtual-um ms-udp-port <0-65535> | 36 |
| 21.1.1.3 | virtual-um ms-multicast-group GROUP | 36 |
| 21.1.1.4 | virtual-um bts-udp-port <0-65535> | 36 |
| 21.1.1.5 | virtual-um bts-multicast-group GROUP | 36 |

| | |
|---|-----------|
| 22 OsmoBTS software architecture | 36 |
| 22.1 OsmoBTS PHY interface abstraction | 36 |
| 22.1.1 PHY link | 36 |
| 22.1.2 PHY instance | 37 |
| 22.1.3 Mapping PHY instances to TRXs | 37 |
| 22.2 Internal control flow | 37 |
| 22.2.1 start-up / sequencing during OsmoBTS start | 37 |
| 22.2.2 At time of OML establishment | 38 |
| 22.2.3 At time of RSL connection loss | 38 |
| 23 VTY Process and Thread management | 38 |
| 23.1 Scheduling Policy | 38 |
| 23.2 CPU-Affinity Mask | 39 |
| 24 TRX Manager UDP socket interface | 40 |
| 24.1 Indications on the Master Clock Interface | 40 |
| 24.2 TRXC protocol | 40 |
| 24.2.1 Power Control | 41 |
| 24.2.2 Tuning Control | 41 |
| 24.2.3 Timeslot Control | 42 |
| 24.2.4 TRXD header version negotiation | 42 |
| 24.3 TRXD protocol | 43 |
| 24.3.1 Uplink Data Burst | 43 |
| 24.3.1.1 Coding of MTS: Modulation and Training Sequence info | 44 |
| 24.3.2 Downlink Data Burst | 45 |
| 25 Osmocom Control Interface | 46 |
| 25.1 Control Interface Protocol | 46 |
| 25.1.1 GET operation | 47 |
| 25.1.2 SET operation | 47 |
| 25.1.3 TRAP operation | 48 |
| 25.2 Common variables | 48 |
| 25.3 Control Interface python examples | 48 |
| 25.3.1 Getting rate counters | 49 |
| 25.3.2 Setting a value | 49 |
| 25.3.3 Getting a value | 49 |
| 25.3.4 Listening for traps | 49 |
| 26 Glossary | 49 |
| A Osmocom TCP/UDP Port Numbers | 57 |

| | |
|---|-----------|
| B Bibliography / References | 59 |
| B.0.0.0.1 References | 59 |
| C GNU Free Documentation License | 62 |
| C.1 PREAMBLE | 62 |
| C.2 APPLICABILITY AND DEFINITIONS | 62 |
| C.3 VERBATIM COPYING | 63 |
| C.4 COPYING IN QUANTITY | 63 |
| C.5 MODIFICATIONS | 63 |
| C.6 COMBINING DOCUMENTS | 65 |
| C.7 COLLECTIONS OF DOCUMENTS | 65 |
| C.8 AGGREGATION WITH INDEPENDENT WORKS | 65 |
| C.9 TRANSLATION | 65 |
| C.10 TERMINATION | 65 |
| C.11 FUTURE REVISIONS OF THIS LICENSE | 66 |
| C.12 RELICENSING | 66 |
| C.13 ADDENDUM: How to use this License for your documents | 66 |

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 this manual

This manual should help you getting started with the OsmoBTS software. It will cover aspects of configuring and running OsmoBTS as well as some introduction about its internal architecture and external interfaces.

About OsmoBTS

OsmoBTS is an implementation of a GSM BTS (Base Transceiver Station). A BTS serves as the interface between the Um radio interface towards phones and the wired Abis interface towards the BSC (Base Station Controller). It also implements the network side of the Layer 2 of the Um radio interface: The LAPDm protocol.

OsmoBTS is licensed as Free and Open Source Software (FOSS) under *GNU AGPLv3* [gnu-agplv3]. It is developed as one GSM network infrastructure component part of the overall Osmocom project.

As perhaps the first implementation of a GSM BTS ever in the industry, OsmoBTS is implemented in a vendor-independent way and supports a large variety of transceiver hardware and physical layer implementations from many vendors.

Credits

OsmoBTS was originally developed in 2011 by Andreas Eversberg and Harald Welte. It has since been maintained by Harald Welte and Holger Freyther at sysmocom.

OsmoBTS in the Osmocom GSM network architecture

OsmoBTS can be used in combination with the various other GSM network elements developed under the umbrella of the Osmocom project.

Typical configurations either use OsmoBTS with OsmoBSC, or with OsmoNITB, as can be seen in the following figures.

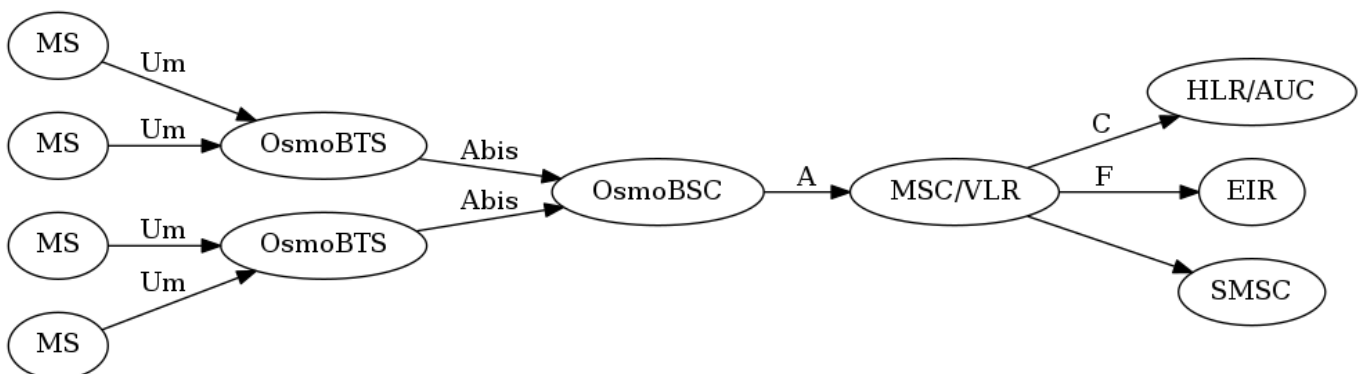


Figure 1: Classic GSM architecture using OsmoBTS with OsmoBTS components

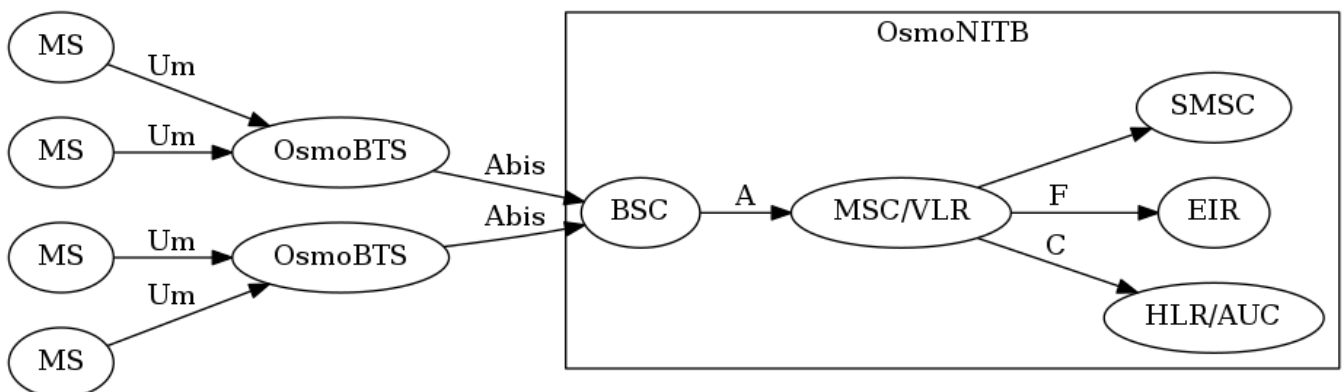


Figure 2: GSM architecture using OsmoBTS + OsmoNITB

If intended by the user, it is of course also possible to implement an OsmoBTS-compatible Abis-over-IP interface in any third party BSC. The Abis/IP interface and its protocol are documented in the *OsmoBTS Abis Protocol Specification* [\[osmobts-abis-spec\]](#). However, be advised that such a configuration is currently not officially supported by Osmocom.

Abis/IP Interface

A-bis Operation & Maintenance Link

The GSM Operation & Maintenance Link (OML) is specified in 3GPP TS 12.21 and is used between a GSM Base-Transceiver-Station (BTS) and a GSM Base-Station-Controller (BSC). The default TCP port for OML is 3002. The connection will be opened from the BTS to the BSC.

Abis OML is only specified over E1 interfaces. The Abis/IP implementation of OsmoBTS and OsmoBSC extend and/or deviate from the TS 12.21 specification in several ways. Please see the *OsmoBTS Abis Protocol Specification* [\[osmobts-abis-spec\]](#) for more information.

A-bis Radio Signalling Link

The GSM Radio Signalling Link (RSL) is specified in 3GPP TS 08.58 and is used between a GSM Base-Transceiver-Station and a GSM Base-Station-Controller (BSC). The default TCP port for RSL is 3003. The connection will be opened from the BTS to BSC after it has been instructed by the BSC.

Abis RSL is only specified over E1 interfaces. The Abis/IP implementation of OsmoBTS and OsmoBSC extend and/or deviate from the TS 08.58 specification in several ways. Please see the *OsmoBTS Abis Protocol Specification* [\[osmobts-abis-spec\]](#) for more information.

Locate Abis/IP based BTS

We can use a tool called abisip-find to be able to find BTS which is connected in the network. This tool is located in the OsmoBSC project repository under: `./src/ipaccess`

abisip-find

abisip-find is a small command line tool which is used to search and find BTS devices in your network (e.g. sysmoBTS, nanoBTS).

It uses broadcast packets of the UDP variant of the Abis-IP protocol on port 3006, and thus will find any BTS that can be reached by the all-network broadcast address 255.255.255.255

When program is started it will print one line for each BTS it can find.

Example: using abisip-find to find BTS in your network

```
$ ./abisip-find
abisip-find (C) 2009 by Harald Welte
This is FREE SOFTWARE with ABSOLUTELY NO WARRANTY

you might need to specify the outgoing
network interface, e.g. ``abisip-find eth0``
Trying to find ip.access BTS by broadcast UDP...

MAC_Address='24:62:78:01:02:03' IP_Address='192.168.0.171' Serial_Number='123'
Unit_ID='sysmoBTS 1002'

MAC_Address='24:62:78:04:05:06' IP_Address='192.168.0.182' Serial_Number='456'
Unit_ID='sysmoBTS 1002'
```

```
MAC Address='00:01:02:03:04:05' IP Address='192.168.100.123' Unit ID='65535/0/0'
Location_1='' Location 2='BTS_NBT131G' Equipment Version='165a029_55'
Software Version='168a302_v142b13d0' Unit Name='nbts-00-02-95-00-4E-B3'
Serial Number='00123456'
```

```
^C
```

You may have to start the program as a root:

```
$ sudo ./abisip-find eth0
```

Deploying a new nanoBTS

A tool called ipaccess-config can be used to configure a new ip.access nanoBTS.

ipaccess-config

This program is very helpful tool which is used to configure Unit ID and Primary OML IP. You can find this tool in the OsmoBSC repository under: */src/ipaccess*

Example: using ipaccess-config to configure Unit ID and Primary OML IP of nanoBTS

```
$ ./ipaccess-config -u 1801/0/0❶ 10.9.1.195❷ -o 10.9.1.154❸

ipaccess-config (C) 2009-2010 by Harald Welte and others
This is FREE SOFTWARE with ABSOLUTELY NO WARRANTY

Trying to connect to ip.access BTS ...
abis_nm.c:316 OC=SITE-MANAGER(00) INST=(ff,ff,ff) STATE CHG:
OP_STATE=Disabled AVAIL=Not installed(07)
abis_nm.c:316 OC=BTS(01) INST=(00,ff,ff) STATE CHG:
OP_STATE=Disabled AVAIL=Not installed(07) ADM=Locked
abis_nm.c:316 OC=BASEBAND-TRANSCEIVER(04) INST=(00,00,ff) STATE CHG:
OP_STATE=Disabled AVAIL=Not installed(07) ADM=Locked
OML link established using TRX 0
setting Unit ID to '1801/0/0'
setting primary OML link IP to '10.9.1.154'
abis_nm.c:316 OC=CHANNEL(03) INST=(00,00,00) STATE CHG:
OP_STATE=Disabled AVAIL=Not installed(07) ADM=Locked
...
abis_nm.c:2433 OC=BASEBAND-TRANSCEIVER(04) INST=(00,00,ff) IPACCESS(0xf0):
SET NVATTR ACK
Set the NV Attributes.
```

- ❶ Unit ID
- ❷ IP address of the NITB
- ❸ IP address of the nanoBTS

OsmoBTS Interfaces

OsmoBTS offers a set of interfaces to interact with external entities:

- A-bis/IP interface to talk to the BSC

- bts_model specific PHY interface
- VTY interface
- Osmocom control interface
- GSMTAP interface
- PCU interface

OsmoBTS Abis/IP Interface

OsmoBTS implements the GSM A-bis interface as described in the relevant 3GPP specifications:

- A-bis RSL according to 3GPP TS 08.58
- A-bis OML according to 3GPP TS 12.21

As the 3GPP specifies A-bis only over E1 interfaces and not over IP, significant enhancements and modifications to the 3GPP specifications are employed. Nevertheless, the implementation tries to stay as close as possible to the 3GPP specifications.

Please see the *OsmoBTS Abis Protocol Specification* [\[osmobts-abis-spec\]](#) for more information on this subject.

bts_model specific PHY interface

This interface is specific to the bts_model that OsmoBTS was compiled for. It can take any form as required by the respective hardware.

Please see the PHY documentation of your respective BTS hardware for more details.

OsmoBTS VTY Interface

See Section [12](#) for further information.

OsmoBTS Control Interface

The general structure of the Osmocom control interface is described in Section [25](#).

The number of control interface commands/attributes is currently quite limited and largely depends on the bts_model used.

trx.N.thermal-attenuation

The idea of this parameter is to attenuate the system output power as part of thermal management. In some cases the PA might be passing a critical level, so an external control process can use this attribute to reduce the system output power.

Please note that all values in the context of transmit power calculation are integers in milli-dB (1/10000 bel), so the below example is setting the attenuation at 3 dB:

```
bsc_control.py -d localhost -p 4238 -s trx.0.thermal-attenuation 3000
Got message: SET_REPLY 1 trx.0.thermal-attenuation 3000
```

```
bsc_control.py -d localhost -p 4238 -g trx.0.thermal-attenuation
Got message: GET_REPLY 1 trx.0.thermal-attenuation 3000
```


OsmoBTS GSMTAP Interface

GSMTAP is a standard created by Osmocom to UDP-encapsulate GSM protocol messages normally communicated over non-IP interfaces for the primary purpose of protocol analysis in the wireshark dissector.

The initial purpose was to encapsulate GSM Um frames including some meta-data like ARFCN and GSM frame number into something that can be parsed and dispatched within the wireshark dissector.

This interface has since been extended to many other GSM/GPRS/UMTS interfaces and protocols, and even to TETRA and GMR.

In OsmoBTS, it is possible to export both uplink and downlink Um messages via GSMTAP. There is a set of VTY configuration options to specify for which logical channels of the Um interface GSMTAP messages shall be emitted, and to which destination IP address they shall be sent.

Using GSMTAP it is possible to place a virtual tap at the air interface between BTS and MS, without going through the trouble of setting up an actual radio receiver at the same frequencies. Also, GSMTAP export is performed before the Um air-interface encryption (A5) is performed, so all frames are always in plain text.

Please refer to Section [14.2.2](#) for more information on how to configure and use this interface.

OsmoBTS PCU Socket Interface

In order to assist the provisioning of GPRS services over the same radio interface as circuit-switched GSM, OsmoBTS exposes a Unix domain socket based interface towards OsmoPCU.

OsmoPCU is the Osmocom implementation of the GPRS Packet Control Unit (PCU), which is co-located with the BTS in the Osmocom implementation. Contrary to that, many classic E1-based implementations of the GSM RAN co-locate the PCU with the BSC. However, the GSM specifications keep the location up to the implementor.

The GPRS network architecture is shown in Figure 3.

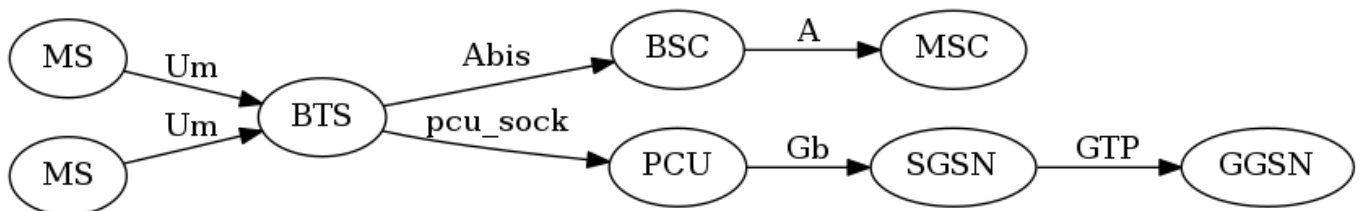


Figure 3: GPRS network architecture

The PCU socket interface serves the following purposes:

- to pass PCU relevant configuration from BTS to PCU
- to forward paging requests from BTS to PCU
- to forward RACH Requests from BTS to PCU

Depending on your `bts_model`, the PCU may also be passing actual PH-DATA.request / PH-DATA.indication / PH-RTS.indication primitives for the PDCH. This is considered sub-optimal, and some BTS models offer a direct interface by which the PCU can exchange those primitives directly with the PHY.

The default PCU socket interface name is `/tmp/pcu_sock`, but this can be overridden by the `pcu-socket` VTY command in the BTS configuration VTY node.

Control interface

The actual protocol is described in Section 25, the variables common to all programs using it are described in Section 25.2. Here we describe variables specific to OsmoBTS. The commands starting with prefix "net.btsN." are specific to a certain BTS so N have to be replaced with BTS number when issuing command. Similarly the TRX-specific commands are additionally prefixed with TRX number e. g. "net.bts1.trx2.thermal-attenuation".

Table 1: Variables available over control interface

| Name | Access | Trap | Value | Comment |
|-----------------------------------|--------|------|---------|------------------------------|
| net.btsN.trxM.thermal-attenuation | RW | No | integer | See Section 7.1 for details. |

thermal-attenuation

Allowed SET value for thermal attenuation is between 0 to 40 dB. Note: the value is SET in dB units but GET will return value in mdB units used internally.

Osmocom Counters

The following gives an overview of all the types of counters available:

Osmo Counters (deprecated)

Osmo counters are the oldest type of counters added to Osmocom projects. They are not grouped.

- Printed as part of VTY show stats
- Increment, Decrement
- Accessible through the control interface: counter.<counter_name>

Rate Counters

Rate counters count rates of events.

- Printed as part of VTY show stats
- Intervals: per second, minute, hour, day or absolute value
- Increment only
- Accessible through the control interface
- Rate counters are grouped and different instances per group can exist

The control interface command to get a counter (group) is:

```
rate_ctr.per_{sec,min,hour,day,abs}.<group_name>.<idx>.[counter_name]
```

It is possible to get all counters in a group by omitting the counter name

Stat Item

Stat items are a grouped replacement for osmo counters.

- Printed as part of VTY show stats
- Replacement for osmo counters
- Not yet available through the control interface
- Grouped and indexed like rate counters
- Items have a unit
- Keeps a list of the last values measured, so could return an average, min, max, std. deviation. So far this is not implemented in any of the reporting options.

Statistic Levels

There are three levels on which a statistic can be aggregated in Osmocom projects: globally, per-peer and per-subscriber.

Global

These are global statistics.

Peer

These statistics relate to a peer the program connects to such as the NSVC in an SGSN.

This level also includes reporting global statistics.

Subscriber

These statistics are related to an individual mobile subscriber. An example would be bytes transferred in an SGSN PDP context.

This level also includes global and peer-based statistics.

Stats Reporter

The stats reporter periodically collects osmo counter, rate counter and stat item values and sends them to a backend. Currently implemented are outputting to the configured log targets and a statsd connector.

Configuring a stats reporter

Periodically printing the statistics to the log can be done in the following way:

Example 8.1 Log statistics

```
OsmoBSC> enable
OsmoBSC# configure terminal
OsmoBSC(config)# stats interval 60 ❶
OsmoBSC(config)# stats reporter log ❷
OsmoBSC(config-stats)# level global ❸
OsmoBSC(config-stats)# enable ❹
```

- ❶ The interval determines how often the statistics are reported.
- ❷ Write the statistic information to any configured log target.
- ❸ Report only global statistics (can be global, peer, or subscriber).
- ❹ Enable the reporter, disable will disable it again.

The counter values can also be sent to any aggregation/visualization tool that understands the statsd format, for example a statsd server with graphite or prometheus using the statsd_exporter together with grafana.

The statsd format is specified in https://github.com/b/statsd_spec

Example 8.2 Report statistics to statsd

```
OsmoBSC> enable
OsmoBSC# configure terminal
OsmoBSC(config)# stats interval 10
OsmoBSC(config)# stats reporter statsd ❶
OsmoBSC(config-stats)# prefix BSC1 ❷
OsmoBSC(config-stats)# level subscriber ❸
OsmoBSC(config-stats)# remote-ip 1.2.3.4 ❹
OsmoBSC(config-stats)# remote-port 8125 ❺
OsmoBSC(config-stats)# enable
```

- ❶ Configure the statsd reporter.
- ❷ Prefix the reported statistics. This is useful to distinguish statistics from multiple instances of the same service.
- ❸ Report only global statistics or include peer or subscriber statistics as well.
- ❹ IP address of the statsd server.
- ❺ UDP port of the statsd server. Statsd by default listens to port 8125.

Setting up a statsd server and configuring the visualization is beyond the scope of this document.

Counters

These counters and their description based on OsmoBTS 0.8.1.346-33ed (OsmoBTS).

Rate Counters

Table 2: elinp - E1 Input subsystem

| Name | Reference | Description |
|--------------|-----------|--------------|
| hdlc:abort | [?] | HDLC abort |
| hdlc:bad_fcs | [?] | HDLC Bad FCS |
| hdlc:overrun | [?] | HDLC Overrun |
| alarm | [?] | Alarm |
| removed | [?] | Line removed |

Table 3: cbch - cell broadcast channel

| Name | Reference | Description |
|-------------------|-----------|---|
| cbch:rcvd_queued | [?] | Received + queued CBCH messages (Abis) |
| cbch:rcvd_dropped | [?] | Received + dropped CBCH messages (Abis) |
| cbch:sent_single | [?] | Sent single CBCH messages (Um) |
| cbch:sent_default | [?] | Sent default CBCH messages (Um) |
| cbch:sent_null | [?] | Sent NULL CBCH messages (Um) |

Table 4: cbch - cell broadcast channel

| Name | Reference | Description |
|-------------------|-----------|---|
| cbch:rcvd_queued | [?] | Received + queued CBCH messages (Abis) |
| cbch:rcvd_dropped | [?] | Received + dropped CBCH messages (Abis) |
| cbch:sent_single | [?] | Sent single CBCH messages (Um) |
| cbch:sent_default | [?] | Sent default CBCH messages (Um) |
| cbch:sent_null | [?] | Sent NULL CBCH messages (Um) |

Table 5: bts - base transceiver station

| Name | Reference | Description |
|---------------|-----------|-----------------------------------|
| paging:rcvd | [?] | Received paging requests (Abis) |
| paging:drop | [?] | Dropped paging requests (Abis) |
| paging:sent | [?] | Sent paging requests (Um) |
| rach:rcvd | [?] | Received RACH requests (Um) |
| rach:drop | [?] | Dropped RACH requests (Um) |
| rach:handover | [?] | Received RACH requests (Handover) |
| rach:cs | [?] | Received RACH requests (CS/Abis) |
| rach:ps | [?] | Received RACH requests (PS/PCU) |
| agch:rcvd | [?] | Received AGCH requests (Abis) |
| agch:sent | [?] | Sent AGCH requests (Abis) |
| agch:delete | [?] | Sent AGCH DELETE IND (Abis) |

Osmo Stat Items

Osmo Counters

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 6: VTY Parameter Patterns

| Pattern | Example | Explanation |
|-----------|----------------|--------------------------|
| A.B.C.D | 127.0.0.1 | An IPv4 address |
| A.B.C.D/M | 192.168.1.0/24 | An IPv4 address and mask |
| X:X::X:X | ::1 | An IPv6 address |

Table 6: (continued)

| Pattern | Example | Explanation |
|---------------------------|------------------|--|
| X:X::X:X/M | ::1/128 | An IPv6 address and mask |
| 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|sdch)
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
```

The attribute system

The VTY allows to edit the configuration at runtime. For many VTY commands the configuration change is immediately valid but for some commands a change becomes valid on a certain event only. In some cases it is even necessary to restart the whole process.

To give the user an overview, which configuration change applies when, the VTY implements a system of attribute flags, which can be displayed using the `show` command with the parameter `vty-attributes`

Example: Typing show vty-attributes at the VTY prompt

```
OsmoBSC> show vty-attributes
Global attributes:
  ! This command applies immediately
  @ This command applies on VTY node exit
Library specific attributes:
  A This command applies on ASP restart
  I This command applies on IPA link establishment
  L This command applies on E1 line update
Application specific attributes:
  o This command applies on A-bis OML link (re)establishment
  r This command applies on A-bis RSL link (re)establishment
  l This command applies for newly created lchans
```

The attributes are symbolized through a single ASCII letter (flag) and do exist in three levels. This is more or less due to the technical aspects of the VTY implementation. For the user, the level of an attribute has only informative purpose.

The global attributes, which can be found under the same attribute letter in every osmocom application, exist on the top level. The Library specific attributes below are used in various osmocom libraries. Like with the global attributes the attribute flag

letter stays the same throughout every osmocom application here as well. On the third level one can find the application specific attributes. Those are unique to each osmocom application and the attribute letters may have different meanings in different osmocom applications. To make the user more aware of this, lowercase letters were used as attribute flags.

The `list` command with the parameter `with-flags` displays a list of available commands on the current VTY node, along with attribute columns on the left side. Those columns contain the attribute flag letters to indicate to the user how the command behaves in terms of how and when the configuration change takes effect.

Example: Typing `list with-flags` at the VTY prompt

```
OsmoBSC(config-net-bts)# list with-flags
. ... help
. ... list [with-flags]
. ... show vty-attributes
. ... show vty-attributes (application|library|global)
. ... write terminal
. ... write file [PATH]
. ... write memory
. ... write
. ... show running-config
. ... exit
. ... end
. o.. type (unknown|bs11|nanobts|rbs2000|nokia_site|sysmobts)
. ... description .TEXT
. ... no description
. o.. band BAND
. .r. cell_identity <0-65535>
. .r. dtx uplink [force]
. .r. dtx downlink
. .r. no dtx uplink
. .r. no dtx downlink
. .r. location_area_code <0-65535>
. o.. base_station_id_code <0-63>
. o.. ipa unit-id <0-65534> <0-255>
. o.. ipa rsl-ip A.B.C.D
. o.. nokia_site skip-reset (0|1)
! ... nokia_site no-local-rel-conf (0|1)
! ... nokia_site bts-reset-timer <15-100>
```

There are multiple columns because a single command may be associated with multiple attributes at the same time. To improve readability each flag letter gets a dedicated column. Empty spaces in the column are marked with a dot (".")

In some cases the listing will contain commands that are associated with no flags at all. Those commands either play an exceptional role (interactive commands outside "configure terminal", vty node navigation commands, commands to show / write the config file) or will require a full restart of the overall process to take effect.

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 list of categories interactively on the vty, type: `logging level ?`

Log levels

For each of the log categories (see Section 13.1), 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 13.2) for categories (see Section 13.1) as well as filtering (see Section 13.4) 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 13.1 for more details on categories and Section 13.2 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 13.4.

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 gsmmap

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 interspersed 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 gsmmap 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 4: 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 decrease 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)#
```

BTS Configuration

The role of the BTS is to handle the GSM radio interface. When the BTS application is starting, the A-bis OML connection is established towards the BSC. Almost all BTS configuration (such as ARFCN, channel configuration, transmit power, etc.) will be sent from the BSC to the BTS via OML messages. After OML start-up has completed, the BSC will instruct the BTS to establish the RSL connections.

Given that most configuration is downloaded from the BSC into the BTS at start-up time, only some very basic settings have to be made in the OsmoBTS software.

Command Line Options

The OsmoBTS executables (`osmo-bts-sysmo`, `osmo-bts-trx`, `osmo-bts-octphy`, `osmo-bts-litecell115`, ...) share the following generic command line options:

SYNOPSIS

osmo-bts-sysmo [-hl-V] [-d *DBGMASK*] [-D] [-c *CONFIGFILE*] [-s] [-T] [-e *LOGLEVEL*] [-r *PRIO*] [-i *GSMTAP-IP*] [-t <1-255>]

OPTIONS

-h, --help

Print a short help message about the supported options

-V, --version

Print the compile-time version number of the OsmoBTS program

-d, --debug *DBGMASK,DBGLEVELS*

Set the log subsystems and levels for logging to stderr. This has mostly been superseded by VTY-based logging configuration, see Section 13 for further information.

-D, --daemonize

Fork the process as a daemon into background.

-c, --config-file *CONFIGFILE*

Specify the file and path name of the configuration file to be used. If none is specified, use `osmo-bts.cfg` in the current working directory.

-s, --disable-color

Disable colors for logging to stderr. This has mostly been deprecated by VTY based logging configuration, see Section 13 for further information.

-T, --timestamp

Enable time-stamping of log messages to stderr. This has mostly been deprecated by VTY based logging configuration, see Section 13 for further information.

-e, --log-level *LOGLEVEL*

Set the global log level for logging to stderr. This has mostly been deprecated by VTY based logging configuration, see Section 13 for further information.

-r, --realtime *PRIO*

Enable use of the Linux kernel realtime priority scheduler with the specified priority. It is recommended you use this option on low-performance embedded systems or systems that encounter high non-GSM/GPRS load.

-i, --gsmtap-ip *GSMTAP-IP*

Specify the destination IP address for GSMTAP messages.

-t, --trx-num <1-255>

Specify the number of TRX supported by this BTS.

There may be additional, hardware specific command line options by the different `bts_model` implementations.

Configuration using the VTY

Most configuration as well as run-time monitoring and system introspection is implemented using a command-line based interface called *VTY*. A full reference syntax of all existing VTY command is available as a separate document.

See Section 12 for further information on the VTY.

Required BTS/TRX configuration

There are some settings that have to be configured locally in the sysmoBTS, as they cannot be set remotely from the BSC. Those settings are stored in the OsmoBTS configuration file, which commonly is stored in `/etc/osmocom/osmo-bts.cfg`.

Example Minimal configuration file

```
!
! OsmoBTS (0.0.1.100-0455) configuration saved from vty
!!
!
phy 0 ❶
    instance 0 ❷
bts 0 ❸
    band DCS1800
    ipa unit-id 1801 0 ❹
    oml remote-ip 192.168.100.11 ❺
    trx 0 ❻
        phy 0 instance 0 ❼
```

- ❶ You must configure at least one PHY link by means of the PHY node
- ❷ You must configure at least one PHY instance in the PHY link
- ❸ There is always exactly one BTS (`bts 0`) configured in OsmoBTS
- ❹ The `ipa unit-id` is what is used to identify this BTS to the BSC
- ❺ The OML Remote IP is the IP address of the BSC, to which the BTS shall connect to.
- ❻ There must be at least one `trx (trx 0)` in each BTS
- ❼ Every TRX must be mapped to a specific PHY instance this way

For a full reference of all available VTY configuration parameters, please refer to the OsmoBTS VTY Reference document.

Configuring GSMTAP tracing

In addition to being able to obtain pcap protocol traces of the A-bis communication and the text-based logging from the OsmoBTS software, there is also the capability of tracing all communication on the radio interface. To do so, OsmoBTS can encapsulate MAC blocks (23byte messages at the L2-L1 interface) into *GSMTAP* and send them via UDP/IP. At that point, they can be captured with utilities like **tcpdump** or **tshark** for further analysis by the **wireshark** protocol analyzer.

In order to activate this feature, you first need to make sure to start OsmoBTS using the `-i` or `--gsmtap-ip` command line option, specifying the destination IP address for the GSMTAP messages. In most cases, using 127.0.0.1 for passing the messages over the loopback (`lo`) device will be sufficient.

OsmoBTS can selectively trace such messages by their L1 SAPI, for both Rx and Tx. For a complete list of L1 SAPI values, please refer to the *OsmoBTS VTY reference manual* [\[vty-ref-osmobts\]](#).

For example, to enable GSMTAP tracing for messages on all SDCCH channels, you can use the `gsmtap-sapi sdcch` command at the CONFIG TRX node of the OsmoBTS VTY.

Example: Enabling GSMTAP for SDCCH

```
OsmoBTS> enable
OsmoBTS# configure terminal
OsmoBTS(config)# bts 0
OsmoBTS(bts)# trx 0
OsmoBTS(trx)# gsmtap-sapi sdcch
OsmoBTS(trx)# write ❶
```

- ❶ the `write` command will make the configuration persistent in the configuration file. This is not required if you wish to enable GSMTAP only in the current session of OsmoBTS.

De-activation can be performed similarly by using the `no gsmtap-sapi sdcch` command at the `trx` node of the OsmoBTS VTY.

From the moment they are enabled via VTY, GSMTAP messages will be generated and sent in UDP encapsulation to the IANA-registered UDP port for GSMTAP (4729) at the IP address specified in the command line argument.

Configuring power ramping

OsmoBTS can ramp up the power of its `trx` over time. This helps reduce cell congestion in busy environments.

Some models of OsmoBTS (such as `osmo-bts-trx`) also support ramping down the transmit power over time until finally ceasing broadcast, for instance due to a `trx` becoming administratively locked or due to the whole BTS being gracefully shut down. This allows for mobile stations camping on the cell to gradually move to other cells in the area once the signal drop is detected.

In this example, the `trx` starts with 5dBm output power which increases by 1dB every two seconds until it reaches nominal power. Power ramping can use the `power-ramp` commands at the CONFIG TRX node of the OsmoBTS VTY.

Example: Configure power ramping on `trx 0`

```
OsmoBTS> enable
OsmoBTS# configure terminal
OsmoBTS(config)# bts 0
OsmoBTS(bts)# trx 0
OsmoBTS(trx)# power-ramp max-initial 5 dBm
OsmoBTS(trx)# power-ramp step-size 1 dB
OsmoBTS(trx)# power-ramp step-interval 2
OsmoBTS(trx)# write ❶
```

- ❶ the `write` command will make the configuration persistent in the configuration file.

De-activating power-ramping can be performed by setting the `max-initial` value to the nominal power. The default `max-initial` value is 23 dBm.

Running multiple instances

It is possible to run multiple instances of `osmo-bts` on one and the same machine, if the phy-interface is flexible enough to distinguish between different phy hardware interfaces.

Since usually a BTS instance runs in conjunction with a dedicated PCU instance, the socket path between PCU and BTS has to be distinguished between the running instances. It is possible to change the default socket path via VTY config:

Example: Personalize PCU socket path

```
bts 0
pcu-socket /tmp/pcu_bts_2
```

It is also necessary to separate the VTY and CTRL interfaces of the different instances. The VTY, as well as the CTRL interface can be bound to a free IP address from the loopback range:

Example: Binding VTY and CTRL interface to a specific IP address

```
line vty
bind 127.0.0.2
ctrl
bind 127.0.0.2
```

Support for Dynamic Timeslots (TCH/F, TCH/H, PDCH)

OsmoBTS supports dynamic switchover of timeslots between different physical channel configurations, initiated by the BSC via (non-standard) Abis messages — see the *OsmoBTS Abis Protocol Specification* [\[osmobts-abis-spec\]](#).

The Abis message handling for dynamic timeslots is independent of the BTS model. However, dynamic switchover will only work for BTS models that implement the internal API to reconnect a timeslot (*bts_model_ts_disconnect()* and *bts_model_ts_connect()*), see also Section 16).

Currently, these OsmoBTS models support dynamic timeslots:

- *osmo-bts-sysmo*
- *osmo-bts-litecell15*
- *osmo-bts-trx*

Dynamic timeslots are driven by the BSC and need to be configured there. When using OsmoBSC or OsmoNITB, see the BTS configuration chapter on dynamic timeslots in [\[userman-osmobsc\]](#) or [\[userman-osmonitb\]](#), respectively.

OsmoBTS hardware support

OsmoBTS consists of a *common* part that applies to all BTS models as well as *hardware-specific* parts for each BTS model. The hardware specific parts are generally referred to as the *bts_model* code.

The common part includes the core BTS architecture as well as code for implementing the external interfaces such as Abis, control, PCU socket and GSMTAP.

The *bts_model* parts include support for driving one particular implementation of a GSM physical layer (PHY). 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.

Every PHY implementation offers some kind of primitives by which the PHY can be controlled, and by which the PHY exchanges data with the higher layers of the protocol stack in the OsmoBTS code.

The PHY-specific primitives are encapsulated in the *bts_model* code, and offered as a PHY-independent *LISAP* interface towards the common part of OsmoBTS.

In addition, each *bts_model* implements a set of functions that the common part calls. Those functions are pre-fixed by *bts_model_*.

Each *bts_model* may offer

- model-specific VTY commands for both configuration and run-time interaction
- model-specific command line arguments
- model-specific control interface commands

osmo-bts-sysmo for sysmocom sysmoBTS

The sysmocom sysmoBTS is a range of GSM BTSs based around an embedded system implementing the PHY in a combination of DSP+FPGA. The PHY is configured by a set of primitives described by header files. Those primitives are exchanged over a set of message queues exposed on the Linux-running ARM core via device nodes in `/dev/msgq/`. Internally, the message queues map to shared memory between the Linux-running ARM core and the DSP running the PHY implementation.

The OsmoBTS *bts_model* code for the sysmoBTS can be found in the `src/osmo-bts-sysmo` sub-directory of the OsmoBTS code base.

osmo-bts-sysmo has been the primary target platform for OsmoBTS for many years and is thus the most feature-complete and mature platform supported by OsmoBTS at this point.

The sysmoBTS PHY supports a direct PHY interface to OsmoPCU, reducing the latency and amount of primitives that OsmoBTS would otherwise need to pass through from the PHY message queues to the PCU socket and vice-versa.

osmo-bts-sysmo specific command line arguments

--dsp-trace *DSPMASK*

Set the DSP trace flags (a single hexadecimal 32bit value). This has been deprecated by VTY based commands, see Section [17.2.1.2](#) for further information.

--pcu-direct

Indicate that an external PCU (e.g. OsmoPCU) will directly open the DSP message queues to the PHY / PH-SAP, and only MPH primitives are passed via OsmoBTS.

osmo-bts-sysmo specific VTY commands

For a auto-generated complete syntax reference of the VTY commands, please see the associated *OsmoBTS VTY reference manual* [\[vty-ref-osmobts\]](#). The section below only lists the most important commands.

at the *SHOW* node

show trx 0 clock-source

Display the currently active clock source configuration for the TRX

show trx 0 dsp-trace-flags

Show the currently active DSP trace flags for the TRX

trx 0 dsp-trace-flag

Use this command to enable/disable/configure the DSP tracing flags that define what debug messages will appear on `/dev/rtrfifo/dsp_trace`.

at the *ENABLE* node

trx 0 tx-power <-110-100>

Change the current TRX transmit power to the given value in dBm.

trx 0 rf-clock-info reset

Part of the clock calibration procedure: Reset the clock correction value.

trx 0 rf-clock-info correct

Part of the clock calibration procedure: Apply the current measured correction value between the reference clock and the local clock.

at the *PHY instance* node

clock-calibration eeprom

Obtain clock calibration value from EEPROM.

clock-calibration default

Use hardware default clock calibration value.

clock-calibration <-4095-4095>

Use specified clock calibration value (not EEPROM/default).

clock-source (tcxo|ocxo|ext|gps)

Specify the clock source for the PHY:

tcxo

Use the TCXO. This is the default on sysmoBTS 2050.

ocxo

Use the OCXO (only valid on units equipped with OCXO). This is the default on all sysmoBTS 1002/1020/1100 and SOB-BTS.

ext

Use the external clock input.

gps

Use the clock derived from GPS. You shouldn't use this clock directly, but rather use the TCXO and regularly re-calibrate against GPS.

trx-calibration-path PATH

Use calibration files from the given *PATH*, rather than calibration values from the EEPROM.

osmo-bts-sysmo specific control interface commands**trx.0.clock-info**

Obtain information on the current clock status:

```
bsc_control.py -d localhost -p 4238 -g trx.0.clock-info  
Got message: GET_REPLY 1 trx.0.clock-info -100,ocxo,0,0,gps
```

which is to be interpreted as:

- current clock correction value is -100 ppb
- current clock source is OCXO
- deviation between clock source and calibration source is 0 ppb
- resolution of clock error measurement is 0 ppt (0 means no result yet)
- current calibration source is GPS

When this attribute is set, any value passed on is discarded, but the clock calibration process is re-started.

trx.0.clock-correction

This attribute can get and set the current clock correction value:

```
bsc_control.py -d localhost -p 4238 -g trx.0.clock-correction
Got message: GET_REPLY 1 trx.0.clock-correction -100
```

```
bsc_control.py -d localhost -p 4238 -s trx.0.clock-correction -- -99
Got message: SET_REPLY 1 trx.0.clock-correction success
```

osmo-bts-trx for OsmoTRX

OsmoTRX is a 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.

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.

osmo-bts-trx specific VTY commands

For a auto-generated complete syntax reference of the VTY commands, please see the associated *OsmoBTS VTY reference manual* [[vty-ref-osmobts](#)]. The section below only lists the most important commands.

at the *SHOW* node

show transceivers

Display information about configured/connected OsmoTRX transceivers in human-readable format to current VTY session.

at the *PHY* configuration node

osmotrx ip HOST

Set the IP address for the OsmoTRX interface for both the local (OsmoBTS) and remote (OsmoTRX) side of the UDP flows. This option has been deprecated by the more detailed option `osmotrx ip (local|remote) A.B.C.D`.

osmotrx ip (local|remote) A.B.C.D

Set the IP address for the OsmoTRX interface for either the local (OsmoBTS) or remote (OsmoTRX) side of the UDP flows.

osmotrx base-port (local|remote) <0-65535>

Configure the base UDP port for the OsmoTRX interface for either the local (OsmoBTS) or remote (OsmoTRX) side of the UDP flows.

osmotrx fn-advance <0-30>

Set the number of frames to be transmitted to transceiver in advance of current GSM frame number.

GSM is a TDMA (time division multiple access) system on the radio interface. OsmoTRX is the "clock master" of that in the Osmocom implementation. It informs OsmoBTS of the current GSM frame number. However, as there is non-zero delays (UDP packet transmission delay, operating system scheduler delay on both OsmoTRX and OsmoBTS side, ...), OsmoBTS must compensate for that delay by "advancing" the clock a certain amount of time.

In other words, if OsmoTRX informs us that the current frame number is N, we advance it by `fn-advance` and transmit burst data for $N + \text{fn-advance}$ towards OsmoTRX.

The `fn-advance` should be kept as low as possible to avoid additional delays to the user voice plane as well as to improve the performance of the control plane (LAPDm) as well as GPRS.

However, `fn-advance` must be kept sufficiently high to ensure no underruns on the OsmoTRX side.

The detailed value will depend on your underlying computer systems, operating system and related tuning parameters. Running OsmoTRX on a remote host will inevitably require a higher `fn-advance` than running it on the same machine, where the UDP packets are just passed over the loopback device.

The default value for `fn-advance` is 2 (corresponding to 9.2 milliseconds).

osmotrx rts-advance <0-30>

Set the number of frames to be requested from L1SAP in advance of current frame number and `fn-advance`.

The value specified as `rts-advance` is added to the current GSM frame number as reported by OsmoTRX **and** the `osmotrx fn-advance` in order to generate the PH-RTS.ind (ready to send indications) across the L1SAP interface inside osmo-bts. This will trigger the Layer 2 (LAPDm for the control plane, RTP for the voice plane, and OsmoPCU for GPRS) to generate a MAC block and input it into the osmo-bts-trx TDMA scheduler.

If OsmoTRX reported N as the current frame number, the actual frame number reported on L1SAP to higher layers will be computed as follows:

$$N + \text{fn-advance} + \text{rts-advance}$$

The default value of `rts-advance` is 3 (corresponding to 14 milliseconds). Do not change this unless you have a good reason!

osmotrx rx-gain <0-50>

Set the receiver gain (configured in the hardware) in dB.

osmotrx tx-attenuation <0-50>

Set the transmitter attenuation (configured in the hardware) in dB.

osmotrx tx-attenuation oml

Use the Value in the A-bis OML Attribute `MAX_POWER_REDUCTION` as transmitter attenuation.

at the **PHY Instance** configuration node

slotmask (1|0) (1|0) (1|0) (1|0) (1|0) (1|0) (1|0) (1|0)

Configure which timeslots should be active on this TRX. Normally all timeslots are enabled, unless you are running on a cpu-constrained deeply embedded system.

osmotrx maxdly <0-31>

Set the maximum delay for received symbols (in number of GSM symbols).

osmo-bts-octphy for Octasic OCTPHY-2G

The Octasic OCTPHY-2G is a GSM PHY implementation inside an Octasic proprietary 24-core DSP called OCTDSP.

This DSP has a built-in Gigabit Ethernet interface, over which it exchanges PHY-layer primitives in raw Ethernet frames with the upper layers running on another CPU attached to the same Ethernet. Those primitives are described in a set of C-language header files.

OsmoBTS implements the raw Ethernet frame based primitives as well as the associated transport protocol (OKTPKT/OCTVC1) in the `osmo-bts-octphy` `bts_model` code.

You can run the `osmo-bts-octphy` on any system connected to the same Ethernet as the OCTDSP running the OCTPHY. This can be either an embedded ARM or x86 SoM part of the OCTBTS hardware, or it can be any other Linux system attached via an Ethernet switch.

Each OCTDSP running OCTSDR-2G offers a set of primitives part of a OKTPKT session, which is mapped to an OsmoBTS PHY link. Depending on the OCTSDR-2G software version, you may create multiple software TRX by creating multiple OsmoBTS PHY instances inside that PHY link.

Multiple DSPs may exist in one circuit board, then each of the DSPs is interfaced by one OsmoBTS PHY link, and each of them may have one or more OsmoBTS PHY instances creating a Multi-TRX configuration.

osmo-bts-litecell115 for Nutaq/Nuran LiteCell 1.5

The Nutaq/Nuran LiteCell 1.5 implements a dual-transceiver GSM BTS based on a mixed ARM/DSP/FPGA architecture. The PHY layer is implemented on DSP/FPGA and similar to that of the sysmoBTS: It exchanges primitives described in a set of C-language header files over message queues between the ARM and the DSP.

This interface is implemented in the `osmo-bts-litecell115` `bts_model` of OsmoBTS. You would run `osmo-bts-litecell115` on the ARM/Linux processor of the Litecell 1.5.

The two transceivers of the Litecell 1.5 each have their own set of DSP message queues. Each set of message queues is wrapped into one OsmoBTS PHY link, offering one OsmoBTS PHY instance.

The Litecell 1.5 PHY supports a direct PHY interface to OsmoPCU, reducing the latency and amount of primitives that OsmoBTS would otherwise need to pass through from the PHY message queues to the PCU socket and vice-versa.

osmo-bts-trx specific VTY commands

For a auto-generated complete syntax reference of the VTY commands, please see the associated *OsmoBTS VTY reference manual* [[vty-ref-osmobts](#)]. The section below only lists the most important commands.

at the **SHOW** node

show phy <0-255> **system-information**

Show information about the hardware platform, DSP and OCTPHY-2G software version.

```
show phy <0-255> rf-port-stats <0-1>
```

Show information about the RF port interfaces.

```
show phy <0-255> clk-sync-stats
```

Show information about the clock synchronization manager.

at the *PHY* configuration node

```
octphy hw-addr HWADDR
```

Specify the Ethernet hardware address (mac address) of the DSP running the OCTPHY-2G software for this PHY link.

```
octphy net-device NAME
```

Specify the Ethernet network device (like `eth0`) through which the DSP can be reached from OsmoBTS.

```
octphy rf-port-index <0-255>
```

Specify which RF port should be used for this PHY link.

```
octphy rx-gain <0-73>
```

Configure the receiver gain in dB.

```
octphy tx-attenuation <0-359>
```

Configure the transmitter attenuation in quarter-dB

osmo-bts-virtual for Virtual Um Interface

This is a special BTS model used for research, simulation and testing. Rather than communicating over a wireless RF interface, the GSM Um messages are encapsulated over GSMTAP/UDP/IP.

The Virtual Um interface (i.e. virtual radio layer) between OsmoBTS and OsmocomBB allows us to run a complete GSM network with 1-N BTSs and 1-M MSs without any actual radio hardware, which is of course excellent for all kinds of testing scenarios.

The Virtual Um layer is based on sending L2 frames (blocks) encapsulated via GSMTAP UDP multicast packets. There are two separate multicast groups, one for uplink and one for downlink. The multicast nature simulates the shared medium and enables any simulated phone to receive the signal from multiple BTSs via the downlink multicast group.

In OsmoBTS, this is implemented via the `osmo-bts-virtual` BTS model.

Setting up OsmoBTS in its `osmo-bts-virtual` flavor isn't really much different from setting it up with real hardware. The amount of required configuration at the BTS configuration file is (as always) very minimal, as in the GSM network architecture provides almost all relevant configuration to the BTS from the BSC.

An example configuration file is provided as part of the `osmo-bts` source code: `doc/examples/virtual/osmobts-virtual.cfg`

For more information see http://osmocom.org/projects/cellular-infrastructure/wiki/Virtual_Um

osmo-bts-virtual specific VTY commands

For a auto-generated complete syntax reference of the VTY commands, please see the associated *OsmoBTS VTY reference manual* [\[vty-ref-osmobts\]](#). The section below only lists the most important commands.

at the *PHY* config node

virtual-um net-device NETDEV

Configure the network device used for sending/receiving the virtual Um interface messages (e.g. `eth0`).

virtual-um ms-udp-port <0-65535>

Configure the UDP port used for sending virtual Um downlink messages towards the MS (default: GSMTAP 4729).

virtual-um ms-multicast-group GROUP

Configure the IP multicast group used for sending virtual Um downlink messages towards the MS (default: 239.193.23.1)

virtual-um bts-udp-port <0-65535>

Configure the UDP port used for receiving virtual Um uplink messages from the MS (default: GSMTAP 4729).

virtual-um bts-multicast-group GROUP

Configure the IP multicast group used for receiving virtual Um uplink messages from the MS (default: 239.193.23.2)

OsmoBTS software architecture

OsmoBTS PHY interface abstraction

The OsmoBTS PHY interface serves as an internal abstraction layer between given PHY hardware (as provided by the `bts_model`) and the actual logical transceivers (TRXs) of a BTS inside the OsmoBTS code base.

PHY link

A PHY link is a physical connection / link towards a given PHY. This might be, for example,

- a set of file descriptors to device nodes in the `/dev/` directory (`sysmobts`, `litecell15`)
- a packet socket for sending raw Ethernet frames to an OCTPHY
- a set of UDP sockets for interacting with OsmoTRX

Each PHY interface has a set of attribute/parameters and a list of 1 to n PHY instances.

PHY links are numbered 0..n globally inside OsmoBTS.

Each PHY link is configured via the VTY using its individual top-level vty node. Given the different `bts-model` / `phy` specific properties, the VTY configuration options (if any) of the PHY instance differ between BTS models.

The PHY links and instances must be configured above the BTS/TRX nodes in the configuration file. If the file is saved via the VTY, the code automatically ensures this.

PHY instance

A PHY instance is an instance of a PHY, accessed via a PHY link.

In the case of osmo-bts-sysmo and osmo-bts-trx, there is only one instance in every PHY link. This is due to the fact that the API inside that PHY link does not permit for distinguishing multiple different logical TRXs.

Other PHY implementations like the OCTPHY however do support addressing multiple PHY instances via a single PHY link.

PHY instances are numbered 0..n inside each PHY link.

Each PHY instance is configured via the VTY as a separate node beneath each PHY link. Given the different bts-model / phy specific properties, the VTY configuration options (if any) of the PHY instance differ between BTS models.

Mapping PHY instances to TRXs

Each TRX node in the VTY must use the *phy N instance M* command in order to specify which PHY instance is allocated to this specific TRX.

Internal control flow

start-up / sequencing during OsmoBTS start

Table 7: Control flow at OsmoBTS start-up procedure

| section | function | description |
|--------------|---------------------------------------|---|
| bts-specific | main() | Entering main() from glibc |
| common | bts_main() | initialization of talloc contexts |
| common | bts_log_init() | initialization of logging |
| common | handle_options() | common option parsing |
| bts-specific | bts_model_handle_options() | model-specific option parsing |
| common | gsm_bts_alloc() | allocation of BTS/TRX/TS data structures |
| common | vty_init() | Initialziation of VTY core, libosmo-abis and osmo-bts VTY |
| common | main() | Setting of scheduler RR priority (if configured) |
| common | main() | Initialization of GSMTAP (if configured) |
| common | bts_init() | configuration of defaults in bts/trx/s object |
| bts-specific | bts_model_init | ? |
| common | abis_init() | Initialization of libosmo-abis |
| common | vty_read_config_file() | Reading of configuration file |
| bts-specific | bts_model_phy_link_set_defaults() | Called for every PHY link created |
| bts-specific | bts_model_phy_instance_set_defaults() | Called for every PHY Instance created |
| common | bts_controlif_setup() | Initialization of Control Interface |
| bts-specific | bts_model_ctrl_cmds_install() | Install model-specific control interface commands |
| common | telnet_init() | Initialization of telnet interface |
| common | pcu_sock_init() | Initialization of PCU socket |
| common | main() | Installation of signal handlers |
| common | abis_open() | Start of the A-bis connection to BSC |
| common | phy_links_open() | Iterate over list of configured PHY links |
| bts-specific | bts_model_phy_link_open() | Open each of the configured PHY links |
| common | write_pid_file() | Generate the pid file |
| common | osmo_daemonize() | Fork as daemon in background (if configured) |
| common | bts_main() | Run main loop until global variable quit >= 2 |

At time of OML establishment

Table 8: Control flow at time of OML establishment

| section | function | description |
|--------------|--------------------------|---|
| bts-specific | bts_model_oml_estab() | Called by core once OML link is established |
| bts-specific | bts_model_check_oml() | called each time OML sets some attributes on a MO, checks if attributes are valid |
| bts-specific | bts_model_apply_oml() | called each time OML sets some attributes on a MO, stores attribute contents in data structures |
| bts-specific | bts_model_opstart() | for NM_OC_BTS, NM_OC_SITE_MANAGER, NM_OC_GPRS_NSE, NM_OC_GPRS_CELL, NMO_OC_GPRS_NSVC |
| bts-specific | bts_model_opstart() | for NM_OC_RADIO_CARRIER for each trx |
| bts-specific | bts_model_opstart() | for NM_OC_BASEB_TRANSC for each trx |
| bts-specific | bts_model_opstart() | for NM_OC_CHANNEL for each timeslot on each trx |
| bts-specific | bts_model_change_power() | change transmit power for each trx (power ramp-up/ramp-down) |

At time of RSL connection loss

Table 9: Control flow at time of RSL connection loss

| section | function | description |
|--------------|------------------------|--|
| bts-specific | bts_model_abis_close() | called when either one of the RSL links or the OML link are down |

VTY Process and Thread management

Most Osmocom programs provide, some support to tune some system settings related to the running process, its threads, its scheduling policies, etc.

All of these settings can be configured through the VTY, either during startup by means of usual config files or through direct human interaction at the telnet VTY interface while the process is running.

Scheduling Policy

The scheduler to use as well as some of its properties (such as realtime priority) can be configured at any time for the entire process. This sort of functionality is useful in order to increase priority for processes running time-constrained procedures, such as those acting on the Um interface, like *osmo-trx* or *osmo-bts*, where use of this feature is highly recommended.

Example: Set process to use RR scheduler

```
cpu-sched
policy rr 1 ❶
```

- ❶ Configure process to use *SCHED_RR* policy with real time priority 1

CPU-Affinity Mask

Most operating systems allow for some sort of configuration on restricting the amount of CPUs a given process or thread can run on. The procedure is sometimes called as *cpu-pinning* since it allows to keep different processes pinned on a subset of CPUs to make sure the scheduler won't run two CPU-hungry processes on the same CPU.

The set of CPUs where each thread is allowed to run on is expressed by means of a bitmask in hexadecimal representation, where the right most bit relates to CPU 0, and the Nth most significant bit relates to CPU *N-1*. Setting the bit means the process is allowed to run on that CPU, while clearing it means the process is forbidden to run on that CPU.

Hence, for instance a cpu-affinity mask of *0x00* means the thread is not allowed on any CPU, which will cause the thread to stall until a new value is applied. A mask of *0x01* means the thread is only allowed to run on the 1st CPU (CPU 0). A mask of *0xff00* means CPUs 8-15 are allowed, while 0-7 are not.

For single-threaded processes (most of Osmocom are), it is usually enough to set this line in VTY config file as follows:

```
cpu-sched
cpu-affinity self 0x01 ❶
```

- ❶ Allow main thread (the one managing the VTY) only on CPU 0

Or otherwise:

```
cpu-sched
cpu-affinity all 0x01 ❶
```

- ❶ Allow all threads only on CPU 0

For multi-threaded processes, it may be desired to run some threads on a subset of CPUs while another subset may run on another one. In order to identify threads, one can either use the TID of the thread (each thread has its own PID in Linux), or its specific Thread Name in case it has been set by the application.

The related information on all threads available in the process can be listed through VTY. This allows identifying quickly the different threads, its current cpu-affinity mask, etc.

Example: Get osmo-trx Thread list information from VTY

```
OsmoTRX> show cpu-sched threads
Thread list for PID 338609:
TID: 338609, NAME: 'osmo-trx-uhd', cpu-affinity: 0x3
TID: 338610, NAME: 'osmo-trx-uhd', cpu-affinity: 0x3
TID: 338611, NAME: 'osmo-trx-uhd', cpu-affinity: 0x3
TID: 338629, NAME: 'osmo-trx-uhd', cpu-affinity: 0x3
TID: 338630, NAME: 'osmo-trx-uhd', cpu-affinity: 0x3
TID: 338631, NAME: 'osmo-trx-uhd', cpu-affinity: 0x3
TID: 338634, NAME: 'UHDAsyncEvent', cpu-affinity: 0x3
TID: 338635, NAME: 'TxLower', cpu-affinity: 0x3
TID: 338636, NAME: 'RxLower', cpu-affinity: 0x3
TID: 338637, NAME: 'RxUpper0', cpu-affinity: 0x3
TID: 338638, NAME: 'TxUpper0', cpu-affinity: 0x3
TID: 338639, NAME: 'RxUpper1', cpu-affinity: 0x3
TID: 338640, NAME: 'TxUpper1', cpu-affinity: 0x3
```

At runtime, one can change the cpu-affinity mask for a given thread identifying it by either TID or name:

Example: Set CPU-affinity from VTY telnet interface

```
OsmoTRX> cpu-affinity TxLower 0x02 ❶
OsmoTRX> cpu-affinity TxLower 0x03 ❷
```

- ❶ Allow thread named *TxLower* (338635) only on CPU 1
- ❷ Allow with TID 338636 (*RxLower*) only on CPU 0 and 1

Since thread names are set dynamically by the process during startup or at a later point after creating the thread itself, One may need to specify in the config file that the mask must be applied by the thread itself once being configured rather than trying to apply it immediately. To specify so, the *delay* keyword is using when configuring in the VTY. If the *delay* keyword is not used, the VTY will report an error and fail at startup when trying to apply a cpu-affinity mask for a yet-to-be-created thread.

Example: Set CPU-affinity from VTY config file

```
cpu-sched
cpu-affinity TxLower 0x01 delay ❶
```

- ❶ Allow thread named *TxLower* (338635) only on CPU 1. It will be applied by the thread itself when created.

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 $0 \leq X \leq N$, the ports related to a channel $0 \leq X \leq N$ 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 by default very low, unless set explicitly by SETPOWER/ADJPOWER beforehand while in POWEROFF state. 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>
```

NOMTXPOWER is used by the BTS to retrieve the nominal output transmit power of the transceiver. SETPOWER/ADJPOWER attenuations (dB) are expected to be applied based on this value (dBm).

```
CMD NOMTXPOWER
RSP NOMTXPOWER <status> <dBm>
```

SETPOWER sets transmit power attenuation wrt the nominal transmit power of the transceiver, in dB.

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

ADJPOWER adjusts by the given dB the transmit power attenuation of the transceiver. Response returns resulting transmit power attenuation wrt the nominal transmit power of the transceiver.

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

RFMUTE locks the RF transceiver, hence disabling emission and reception of information on Air interface of the channel associated to the TRXC connection the command is sent on. Parameter with value of 1 is used to mute, and value of 0 is used to unmute.

```
CMD RFMUTE <1|0>
RSP RFMUTE <status> <1|0>
```

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

| value | Channel Combination |
|-------|--|
| 0 | Channel is transmitted, but unused |
| 1 | TCH/FS |
| 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_rsp> <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.

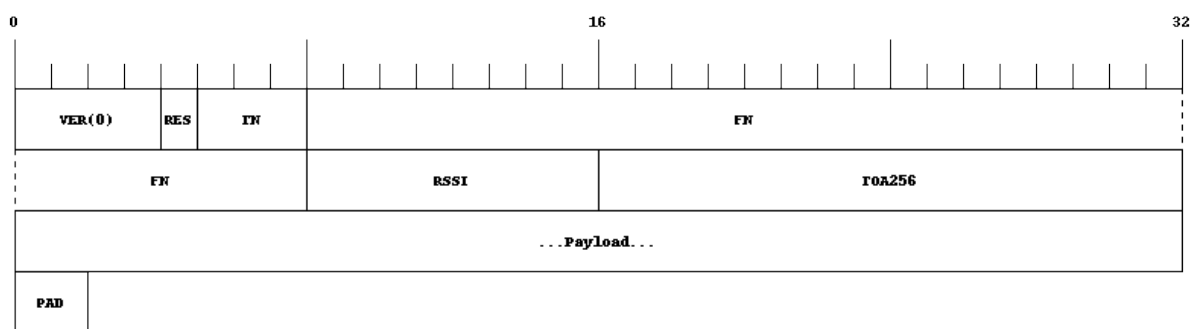


Figure 5: TRXDv0 Uplink data burst message structure

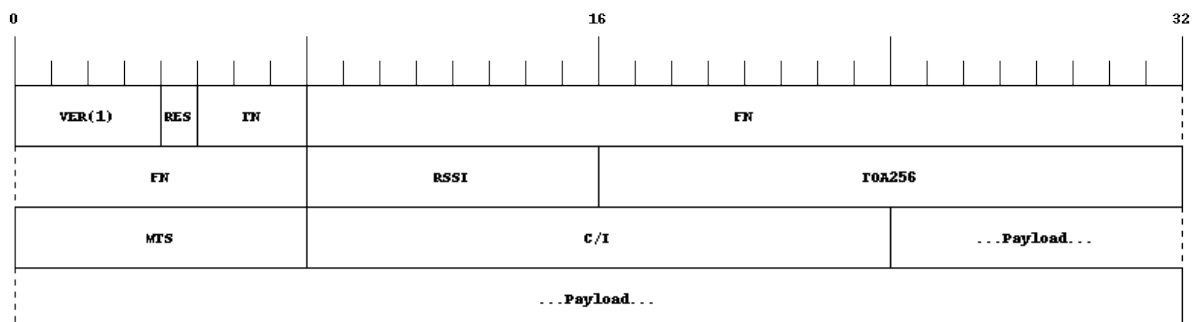


Figure 6: TRXDv1 Uplink data burst message structure

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 24.3.1.1 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

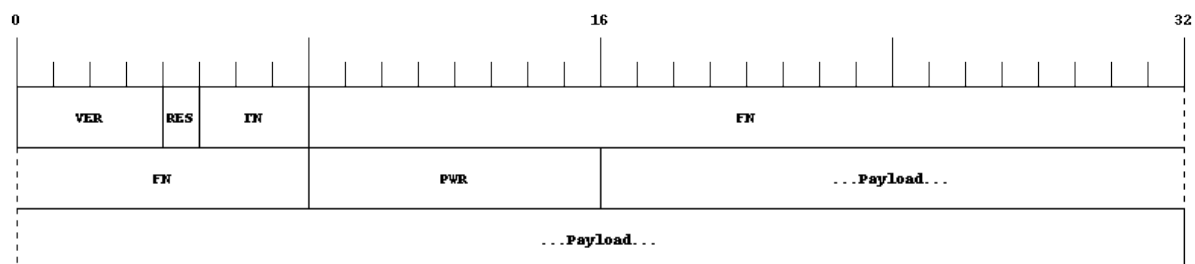


Figure 7: TRXD Downlink data burst message structure

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).

Osmocom Control Interface

The VTY interface as described in Section 12 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).

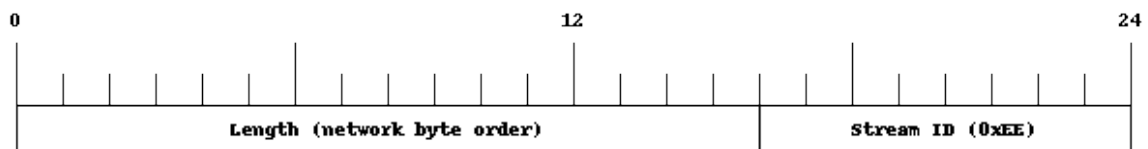


Figure 8: IPA header for control protocol

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

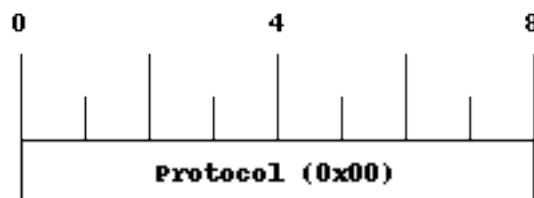


Figure 9: IPA extension header for control protocol

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.

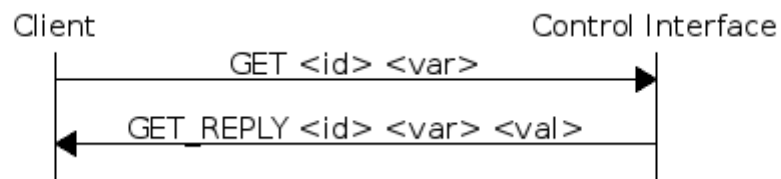


Figure 10: Control Interface GET operation (successful outcome)

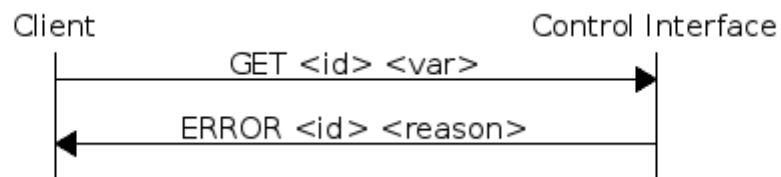


Figure 11: Control Interface GET operation (unsuccessful outcome)

SET operation

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

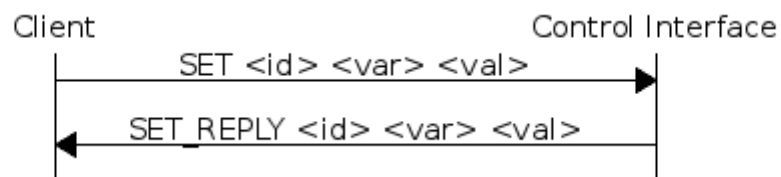


Figure 12: Control Interface SET operation (successful outcome)

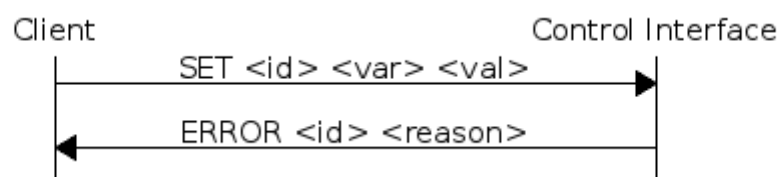


Figure 13: Control Interface SET operation (unsuccessful outcome)

TRAP operation

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

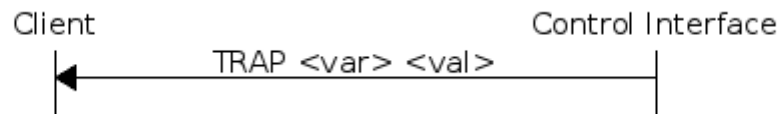


Figure 14: Control Interface TRAP operation

Common variables

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

Table 11: 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 then 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"
"elinp.0","hdlc:abort","0","0","0","0","0"
"elinp.0","hdlc:bad_fcs","0","0","0","0","0"
"elinp.0","hdlc:overrun","0","0","0","0","0"
"elinp.0","alarm","0","0","0","0","0"
"elinp.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

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

Circuit-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

dvnxload

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 subscriber 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

MSC pool

A number of redundant MSCs serving the same core network, which a BSC / RNC distributes load across; see also the "MSC Pooling" chapter in OsmoBSC's user manual [\[userman-osmobsc\]](#) and *3GPP TS 23.236* [\[3gpp-ts-23-236\]](#)

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

NRI

Network Resource Indicator, typically 10 bits of a TMSI indicating which MSC of an MSC pool attached the subscriber; see also the "MSC Pooling" chapter in OsmoBSC's user manual [\[userman-osmobsc\]](#) and *3GPP TS 23.236* [\[3gpp-ts-23-236\]](#)

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

RS�

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 12: 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 |
| 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 |
| TCP | 4268 | telnet (VTY) | osmo-uecups |
| SCTP | 4268 | UECUPS | osmo-uecups |
| TCP | 4269 | telnet (VTY) | osmo-eld |
| TCP | 4271 | telnet (VTY) | osmo-smlc |
| TCP | 4272 | Control Interface | osmo-smlc |
| 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-23-236] 3GPP TS 23.236: Intra-domain connection of Radio Access Network (RAN) nodes to multiple Core Network (CN) nodes <http://www.3gpp.org/DynaReport/23236.htm>
- [20] [3gpp-ts-24-007] 3GPP TS 24.007: Mobile radio interface signalling layer 3; General Aspects <http://www.3gpp.org/DynaReport/24007.htm>

- [21] [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>
- [22] [3gpp-ts-31-101] 3GPP TS 31.101: UICC-terminal interface; Physical and logical characteristics <http://www.3gpp.org/DynaReport/31101.htm>
- [23] [3gpp-ts-31-102] 3GPP TS 31.102: Characteristics of the Universal Subscriber Identity Module (USIM) application <http://www.3gpp.org/DynaReport/31102.htm>
- [24] [3gpp-ts-31-103] 3GPP TS 31.103: Characteristics of the IMS Subscriber Identity Module (ISIM) application <http://www.3gpp.org/DynaReport/31103.htm>
- [25] [3gpp-ts-31-111] 3GPP TS 31.111: Universal Subscriber Identity Module (USIM) Application Toolkit (USAT) <http://www.3gpp.org/DynaReport/31111.htm>
- [26] [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>
- [27] [3gpp-ts-31-116] 3GPP TS 31.116: Remote APDU Structure for (U)SIM Toolkit applications <http://www.3gpp.org/DynaReport/31116.htm>
- [28] [3gpp-ts-35-205] 3GPP TS 35.205: 3G Security; Specification of the MILENAGE algorithm set: General
- [29] [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>
- [30] [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>
- [31] [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>
- [32] [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>
- [33] [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>
- [34] [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>
- [35] [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>
- [36] [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>
- [37] [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>
- [38] [3gpp-ts-51-011] 3GPP TS 51.011: Specification of the Subscriber Identity Module - Mobile Equipment (SIM-ME) interface
- [39] [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>
- [40] [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>
- [41] [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
- [42] [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

- [43] [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
- [44] [ietf-rfc768] IETF RFC 768: Internet Protocol <https://tools.ietf.org/html/rfc791>
- [45] [ietf-rfc793] IETF RFC 793: Transmission Control Protocol <https://tools.ietf.org/html/rfc793>
- [46] [ietf-rfc1035] IETF RFC 1035: Domain Names - Implementation and Specification <https://tools.ietf.org/html/rfc1035>
- [47] [ietf-rfc1350] IETF RFC 1350: Trivial File Transfer Protocol <https://tools.ietf.org/html/rfc1350>
- [48] [ietf-rfc2131] IETF RFC 2131: Dynamic Host Configuration Protocol <https://tools.ietf.org/html/rfc2131>
- [49] [ietf-rfc2719] IETF RFC 2719: Signal Transport over IP <https://tools.ietf.org/html/rfc2719>
- [50] [ietf-rfc3331] IETF RFC 3331: Message Transfer Part 2 User Adaptation Layer <https://tools.ietf.org/html/rfc3331>
- [51] [ietf-rfc3550] IETF RFC 3550: RTP: A Transport protocol for Real-Time Applications <https://tools.ietf.org/html/rfc3550>
- [52] [ietf-rfc3596] IETF RFC 3596: DNS Extensions to Support IP Version 6 <https://tools.ietf.org/html/rfc3596>
- [53] [ietf-rfc3868] IETF RFC 3868: SCCP User Adaptation Layer <https://tools.ietf.org/html/rfc3868>
- [54] [ietf-rfc4165] IETF RFC 4165: Message Transfer Part 2 Peer-to-Peer Adaptation Layer <https://tools.ietf.org/html/rfc4165>
- [55] [ietf-rfc4251] IETF RFC 4251: The Secure Shell (SSH) Protocol Architecture <https://tools.ietf.org/html/rfc4251>
- [56] [ietf-rfc4666] IETF RFC 4666: Message Transfer Part 3 User Adaptation Layer <https://tools.ietf.org/html/rfc4666>
- [57] [ietf-rfc5771] IETF RFC 5771: IANA Guidelines for IPv4 Multicast Address Assignments <https://tools.ietf.org/html/rfc5771>
- [58] [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/>
- [59] [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/>
- [60] [itu-t-q713] ITU-T Q.713: Signalling connection control part formats and codes <https://www.itu.int/rec/T-REC-Q.713/en/>
- [61] [itu-t-q714] ITU-T Q.714: Signalling connection control part procedures <https://www.itu.int/rec/T-REC-Q.714/en/>
- [62] [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>
- [63] [smpp-34] SMPP Developers Forum. Short Message Peer-to-Peer Protocol Specification v3.4 http://docs.nimta.com/SMPP_v3_4_Issue1_2.pdf
- [64] [gnu-agplv3] Free Software Foundation. GNU Affero General Public License. <http://www.gnu.org/licenses/agpl-3.0.en.html>
- [65] [freeswitch_pbx] FreeSWITCH SIP PBX <https://freeswitch.org>

GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

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. Secondly, 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.