# sysmocom

sysmocom - s.f.m.c. GmbH



# osmo-ePDG User Manual

by Alexander Couzens

DRAFT

Copyright © 2024 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 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".

The Asciidoc source code of this manual can be found at https://gitea.osmocom.org/erlang/osmo-epdg

| HISTORY | | | |
|---|---|---|---|
| NUMBER | DATE | DESCRIPTION | NAME |
| 1 | April 23, 2024 | Initial version. | AC |

Copyright © 2024 sysmocom - s.f.m.c. GmbH 

# 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 . . . 3
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 . . . 4
2.4.5 Documentation License . . . 4

**3 Introduction** **4**
3.1 Required Skills . . . 4
3.2 Getting assistance . . . 5

**4 Overview** **5**
4.1 About osmo-epdg . . . 5
4.2 Components of the osmo-epdg . . . 5
4.2.1 strongSwan . . . 6
4.2.1.1 Interfaces of strongSwan . . . 6
4.2.2 osmo-epdg/erlang . . . 6
4.2.2.1 Interfaces of osmo-epdg/erlang . . . 6
4.2.3 Relevant specifications . . . 6

**5 strongSwan** **6**
5.1 Concept of strongSwan within osmo-epdg . . . 7
5.2 CEIA (Charon External AKA Interface) . . . 7

**6 osmo-epdg/erlang** **7**
6.1 Debian packages . . . 8
6.2 Configuration . . . 8
6.2.1 CEIA: Connection for strongSwan . . . 8
6.2.2 SWx (diameter) . . . 8
6.2.3 S6b (diameter) . . . 8
6.2.4 S2b (GTPv2) . . . 9

Copyright © 2024 sysmocom - s.f.m.c. GmbH 

**7 Operating osmo-epdg** **9**

7.1 Linux requirements . . . . . 9

7.2 EPC requirements . . . . . 9

7.2.1 HSS . . . . . 9

7.2.2 PGW . . . . . 10

**8 Acknowledgement** **10**

8.1 NLNet User-Operated Internet Fund . . . . . 10

Copyright © 2024 sysmocom - s.f.m.c. GmbH 

# 1 Foreword

Digital cellular networks based on the GSM specification were designed in the late 1980s and first deployed in the early 1990s 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.

## 1.1 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.

Copyright © 2024 sysmocom - s.f.m.c. GmbH 

- 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
- NLnet Foundation, for providing funding for a number of individual work items within the Osmocom universe, such as LTE support in OsmoCBC or GPRS/EGPRS support for Ericsson RBS6000.
- WaveMobile Ltd, for many years of sponsoring.

May the source be with you!

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

### 1.2 Endorsements

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

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

## 2 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.

### 2.1 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 benefiting 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, workarounds
- 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.

Copyright © 2024 sysmocom - s.f.m.c. GmbH 

## 2.2 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.

## 2.3 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.

## 2.4 Legal disclaimers

### 2.4.1 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!

### 2.4.2 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.

### 2.4.3 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.

Copyright © 2024 sysmocom - s.f.m.c. GmbH 

*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.*

#### 2.4.4 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.

#### 2.4.5 Documentation License

Please see [?] for further information.

# 3 Introduction

## 3.1 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

### 3.2 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 https://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.

If you would like to obtain professional/commercial support on Osmocom CNI, you can always reach out to sales@sysmocom.de to discuss your support needs. Purchasing support from sysmocom helps to cover the ongoing maintenance of the Osmocom CNI software stack.

## 4 Overview

This manual should help you getting started with osmo-epdg. It will cover aspects of configuring and running the osmo-epdg.

### 4.1 About osmo-epdg

osmo-epdg is a combined ePDG (Evolved Packet Data Gateway) and AAA (Authorization, Authentication, Accounting). The ePDG is used by an UE to connect via an untrusted non-3GPP access network towards the PGW to allow VoWifi (Voice over Wifi).

The UE will establish a tunnel using IPSec (IKEv2 and ESP) towards the ePDG (SWu interface). The ePDG will authenticate and authorize the UE by using an AAA and will forward the traffic from the UE towards a PGW.

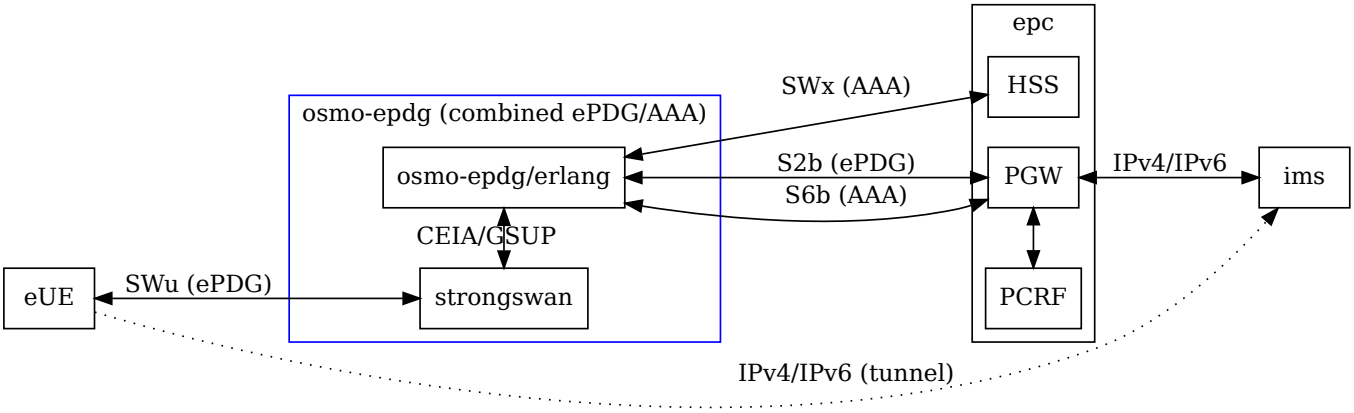



Figure 1: `osmo-epdg` structure

The core of the osmo-epdg is the erlang daemon osmo-epdg/erlang. It communicates with core components by 3GPP protocols and contains the state of UEs. Furthermore, osmo-epdg speaks with strongSwan to handle communication towards UEs. Both components communicate over the internal protocol CEIA based on GSUP.

### 4.2 Components of the osmo-epdg

The combined osmo-epdg consist of:

- strongSwan (IKEv2/ESP)

Copyright © 2024 sysmocom - s.f.m.c. GmbH 

- osmo-epdg/erlang (state handling, communication with other core components)
- Linux kernel (policy routing, firewall, gtp, esp for user-plane traffic)

#### 4.2.1 strongSwan

strongSwan is a FOSS IPSec daemon developed by the strongSwan community. strongSwan has been extended and modified to allow the osmo-epdg/erlang component to control UE sessions, pass authentication, authorisation and configuration.

See Section 5 for further information.

##### 4.2.1.1 Interfaces of strongSwan

- SWu: IPsec/IKEv2 towards the UE.
- netlink: with the Linux kernel to encrypt and decrypt ESP traffic.
- CEIA: internal communication to osmo-epdg/erlang to forward authentication and state based on GSUP.

#### 4.2.2 osmo-epdg/erlang

The osmo-epdg/erlang daemon is the core of the combined ePDG and AAA. It communicates to all relevant core components and control UE sessions of the strongSwan over the CEIA protocol.

##### 4.2.2.1 Interfaces of osmo-epdg/erlang

- SWx: with HSS for authentication, authorisation, accounting (AAA)
- S2b: with PGW to create GTP session (ePDG)
- S6b: with PGW for authorisation of a GTP session (AAA)
- netlink: with the Linux kernel to create GTP tunnels (ePDG)
- CEIA: internal communication to osmo-epdg/erlang to forward authentication and state based on GSUP. (ePDG/AAA)

#### 4.2.3 Relevant specifications

- [3gpp-ts-23-402] 3GPP TS 23.402 Architecture enhancements for non-3GPP accesses
- [3gpp-ts-24-302] 3GPP TS 24.302 Architecture enhancements for non-3GPP accesses
- [3gpp-ts-29-273] 3GPP TS 29.273: 3GPP EPS AAA interfaces
- [3gpp-ts-20-274] 3GPP TS 29.274: Tunnelling Protocol for Control plane (GTPv2-C)
- [3gpp-ts-33-402] 3GPP TS 33.402: Security aspects of non-3GPP accesses
- [ieft-rfc-4555] IETF RFC 4555: IKEv2 Mobility and Multihoming Protocol (MOBIKE)
- [ieft-rfc-5996] IETF RFC 5996: Internet Key Exchange Protocol Version 2 (IKEv2)

## 5 strongSwan

strongSwan is a free software implementation of IPsec, it is developed by the strongSwan community. strongSwan has been extended and modified to allow the osmo-epdg/erlang component to control UE sessions, pass authentication, authorisation and configuration.

The modified strongSwan can be found at https://gitea.osmocom.org/ims-volte-vowifi/strongswan

Example configuration for strongSwan can be found under https://gitea.osmocom.org/ims-volte-vowifi/ansible-prototype

Copyright © 2024 sysmocom - s.f.m.c. GmbH 

### 5.1 Concept of strongSwan within osmo-epdg

The main role of strongSwan in the osmo-epdg is terminating SWu traffic (IKEv2 and ESP) of UEs. strongSwan will also handle EAP traffic, as part of an AAA component. The osmo-epdg/erlang component is the core of the osmo-epdg and the primary source of the truth. For this reason strongSwan should keep as little state as possible while osmo-epdg/erlang keeps the full state.

strongSwan/osmo-epdg and osmo-epdg/erlang communicates via the CEIA (Charon External AKA Interface) and share state over it. Authentication, authorisation, configuration are communicated over the CEIA.

In the default configuration of osmo-epdg, strongSwan will use Linux kernel to handle ESP to achieve high performance on the user-plane. The Linux kernel will decrypt, decapsulate and forward traffic towards GTP tunnels.

### 5.2 CEIA (Charon External AKA Interface)

The Charon External AKA interface is used by strongSwan/osmo-epdg to communicate with osmo-epdg/erlang. It is based on GSUP (https://ftp.osmocom.org/docs/osmo-hlr/master/osmohlr-usermanual.pdf).

strongSwan/osmo-epdg is using the CEAI to:

- Authenticate UEs
- Authorize UEs
- Prepare user plane to forward traffic
- Notify about termination of UEs
- Terminate UE sessions request by HSS (SWu)

The initial connection of GSUP is done by strongSwan (client) towards the osmo-epdg/erlang (server). The protocol is re-using already defined PDU and messages of GSUP. The default configuration will use TCP/IPA port 4222.

## 6 osmo-epdg/erlang

The core component of the osmo-epdg is the osmo-epdg/erlang daemon. It holds the state of the UE and communicates with the relevant core components.

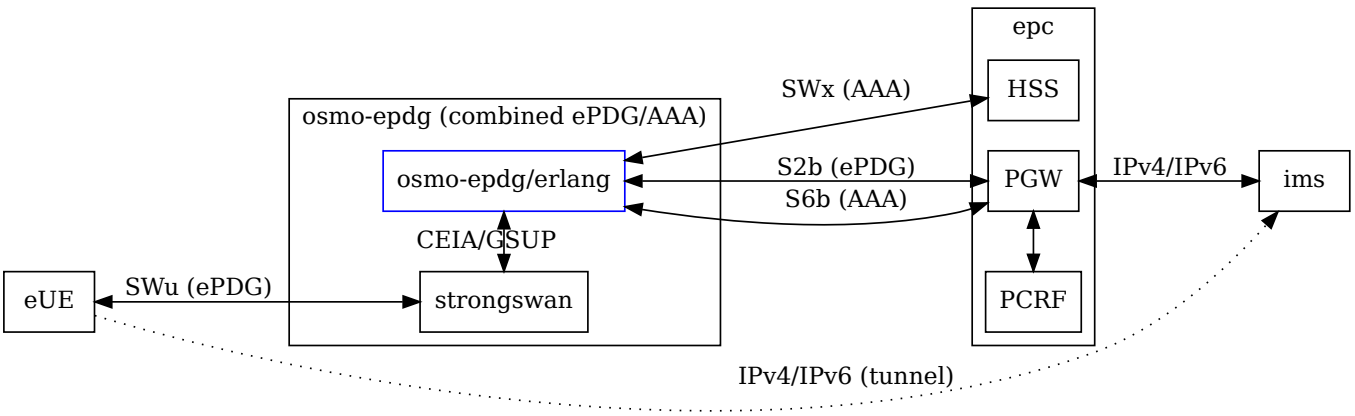



Figure 2: osmo-epdg/erlang structure

Copyright © 2024 sysmocom - s.f.m.c. GmbH 

## 6.1 Debian packages

For Debian based distributions, Osmocom provides package repositories:

- https://osmocom.org/projects/cellular-infrastructure/wiki/Latest_Builds
- https://osmocom.org/projects/cellular-infrastructure/wiki/Nightly_Builds

The osmo-epdg package contains a systemd.service file and a default example configuration.

## 6.2 Configuration

The osmo-epdg/erlang is configured via Erlang configuration (https://www.erlang.org/doc/man/config.html). When using the osmo-epdg debian package, the default location for the configuration is **/etc/osmocom/osmo-epdg.config**.

### 6.2.1 CEIA: Connection for strongSwan

The CEIA is the internal protocol between strongSwan and osmo-epdg/erlang. See [CEIA]

| key | default | description |
|---|---|---|
| gsup_local_ip | 127.0.0.1 | bind IP of the GSUP server (TCP). strongSwan connects to this IP. |
| gsup_local_port | 4222 | bind TCP server port |

### 6.2.2 SWx (diameter)

SWx ([3gpp-ts-29-273]): Interface between AAA and HSS. The AAA will retrieve authentication, authorize the UE and update the location (AAA, PGW address) in the HSS.

See Erlang diameter specific behaviour on https://www.erlang.org/doc/man/diameter.html

| key | default | description |
|---|---|---|
| dia_swx_remote_ip | 127.0.0.1 | Remote IP of the HSS |
| dia_swx_remote_port | 3868 | SCTP port of the HSS |
| dia_swx_proto | sctp | Protocol. sctp or tcp |
| dia_swx_connect_timer | 30000 | Watchdog Connect timer in ms. RFC 6733.dia_swx_watchdog_timer |
| dia_swx_watchdog_config | [ok, suspect] | See https://www.erlang.org/doc/man/-diameter.html |
| dia_swx_transmit_timer | 10000 | SWx Transmit Timeout (ms) |
| dia_swx_vendor_id | 0 | Diameter Vendor Id |
| dia_swx_origin_host | epdg.localdomain | Diameter Origin Host |
| dia_swx_origin_realm | localdomain | Diameter Origin Realm |

### 6.2.3 S6b (diameter)

S6b ([3gpp-ts-29-273]): Interface between AAA and PGW. Authorize the GTP session of the ePDG.

See Erlang diameter specific behaviour on https://www.erlang.org/doc/man/diameter.html

| key | default | description |
|---|---|---|
| dia_s6b_local_ip | 127.0.0.10 | local bind IP |
| dia_s6b_local_port | 3868 | local bind Port |
| dia_s6b_proto | sctp | Protocol. sctp or tcp |

Copyright © 2024 sysmocom - s.f.m.c. GmbH 

| key | default | description |
|---|---|---|
| dia_s6b_connect_timer | 30000 | Watchdog Connect timer in ms. RFC 6733. |
| dia_s6b_watchdog_timer | 30000 | Watchdog TwInit. RFC 3539. |
| dia_s6b_watchdog_config | [ok, suspect] | See https://www.erlang.org/doc/man/-diameter.html |
| dia_s6b_vendor_id | 0 | Diameter Vendor Id |
| dia_s6b_origin_host | aaa.localdomain | Diameter Origin Host |
| dia_s6b_origin_realm | localdomain | Diameter Origin Realm |

#### 6.2.4 S2b (GTPv2)

S2b ([3gpp-ts-29-273]): Interface between ePDG and PGW. GTP control plane for the UE traffic.

| key | default | description |
|---|---|---|
| gtpc_local_ip | 127.0.0.2 | local bind IP |
| gtpc_local_port | 2123 | local bind Port |
| gtpc_remote_ip | 127.0.0.1 | remote IP of the PGW |
| gtpc_remote_port | 2123 | remote port of the PGW |

# 7 Operating osmo-epdg

The osmo-epdg requires to run on Linux. osmo-epdg has been tested and developed with Debian 12. Linux kernel and/or erlang/OTP from Debian 11 are known to cause problems.

## 7.1 Linux requirements

The osmo-epdg is using the following Linux subsystems:

- nftables (soft)
- IP policy routing (soft)
- ESP user-plane (soft)
- GTP user-plane (hard)

Soft dependencies can be changed by configuration. Hard dependencies are required and can't be changed without code changes.

The user-plane is configured in the default configuration to use nftables and policy routing to ensure the traffic from UEs will only flow between the GTP network interface and ESP encrypted tunnels.

strongSwan will use the kernel ESP subsystem to encrypt, decrypt and route traffic. strongSwan can be configured to replace the kernel ESP subsystem with a user space imeplementation, which comes with a performance impact.

osmo-epdg/erlang is using the GTP subsystem to de- and encapsulate. It only supports using the kernel subsystem.

## 7.2 EPC requirements

The osmo-epdg was tested and developed against open5gs and requires version 2.7.1 or newer.

#### 7.2.1 HSS

When a UE connects it requests an APN via SWu. By default the UE will request the APN "ims". The subscriber entry must allow connections to the requested APN to succeed.

Copyright © 2024 sysmocom - s.f.m.c. GmbH 

#### 7.2.2 PGW

The osmo-epdg/erlang expects the PGW to support S2b and S6b. An UE will request attributes via IKEv2 which are translated into Additional PCO (APCO) in GTP on the S2b reference point. The PGW should support Additional PCO via S2b. open5gs supports Additional PCO since 2.7.1.

osmo-epdg only supports S2b over GTP.

# 8 Acknowledgement

## 8.1 NLNet User-Operated Internet Fund

osmo-epdg was funded by NLNet https://nlnet.nl/project/Osmocom-ePDG/ under the User-Operated Internet Fund.

Thank you for sponsoring this project.

Copyright © 2024 sysmocom - s.f.m.c. GmbH 