osmopysim-usermanual

**Sylvain Munaut, Harald Welte, Philipp Maier,
Supreeth Herle, Merlin Chlosta**

Nov 15, 2026

CONTENTS:

1 Introduction **1**

1.1 pySim-shell . . . 1
- 1.1.1 Video Presentation . . . 2
- 1.1.2 Running pySim-shell . . . 2
- 1.1.3 Usage Examples . . . 4
- 1.1.4 Advanced Topics . . . 24
- 1.1.5 cmd2 basics . . . 30
- 1.1.6 pySim commands . . . 30
- 1.1.7 ISO7816 commands . . . 35
- 1.1.8 TS 102 221 commands . . . 40
- 1.1.9 Linear Fixed EF commands . . . 42
- 1.1.10 Transparent EF commands . . . 44
- 1.1.11 BER-TLV EF commands . . . 46
- 1.1.12 USIM commands . . . 47
- 1.1.13 File-specific commands . . . 48
- 1.1.14 UICC Administrative commands . . . 49
- 1.1.15 ARA-M commands . . . 52
- 1.1.16 GlobalPlatform commands . . . 54
- 1.1.17 eUICC ISD-R commands . . . 60
- 1.1.18 cmd2 settable parameters . . . 66

1.2 Card Filesystem Reference . . . 67
- 1.2.1 MF / TS 102 221 (UICC) . . . 67
- 1.2.2 ADF.USIM / TS 31.102 . . . 71
- 1.2.3 ADF.ISIM / TS 31.103 . . . 105
- 1.2.4 DF.GSM + DF.TELECOM / TS 51.011 (SIM) . . . 107
- 1.2.5 CDMA / IS-820 (RUIM) . . . 128
- 1.2.6 DF.EIRENE / GSM-R . . . 129
- 1.2.7 DF.SYSTEM / sysmocom SJA2+SJA5 . . . 138

1.3 pySim-trace . . . 144
- 1.3.1 Demo . . . 144
- 1.3.2 Running pySim-trace . . . 144
- 1.3.3 pySim-trace command line reference . . . 144
- 1.3.4 Constraints . . . 146

1.4 Legacy tools . . . 146
- 1.4.1 pySim-prog . . . 146
- 1.4.2 pySim-read . . . 150

1.5 pySim-smpp2sim . . . 152
- 1.5.1 Running pySim-smpp2sim . . . 152
- 1.5.2 Example execution with sample output . . . 154

1.6 pySim library . . . 154

1.6.1 pySim filesystem abstraction . . . 154
1.6.2 pySim commands abstraction . . . 166
1.6.3 pySim Transport . . . 171
1.6.4 pySim utility functions . . . 175
1.6.5 pySim exceptions . . . 179
1.6.6 pySim card_handler . . . 180
1.6.7 pySim card_key_provider . . . 180
1.7 pySim eSIM libraries . . . 183
1.7.1 GSMA SGP.21/22 Remote SIM Provisioning (RSP) - High Level . . . 184
1.7.2 GSMA SGP.21/22 Remote SIM Provisioning (RSP) - Low Level . . . 189
1.7.3 SIMalliance / TCA Interoperable Profile . . . 192
1.8 osmo-smdpp . . . 225
1.8.1 Running osmo-smdpp . . . 226
1.9 sim-rest-server . . . 228
1.9.1 REST API Calls . . . 228
1.9.2 Concrete example using the included sysmoISIM-SJA2 . . . 229
1.10 suci-keytool . . . 230
1.10.1 Generating keys . . . 230
1.10.2 Dumping public keys . . . 230
1.10.3 suci-keytool syntax . . . 230
1.11 saip-tool . . . 231
1.11.1 Profile Package Examples . . . 231
1.11.2 JAVA card applets . . . 231
1.11.3 saip-tool syntax . . . 233
1.12 smpp-ota-tool . . . 239
1.12.1 Applying OTA keys . . . 239
1.12.2 Addressing an Application . . . 240
1.12.3 A practical example . . . 240
1.12.4 Expanded remote application data format . . . 243
1.12.5 smpp-ota-tool syntax . . . 243

2 Indices and tables **247**

Python Module Index **249**

Index **251**

CHAPTER

ONE

INTRODUCTION

pySim is a python implementation of various software that helps you with managing subscriber identity cards for cellular networks, so-called SIM cards.

Many Osmocom (Open Source Mobile Communications) projects relate to operating private / custom cellular networks, and provisioning SIM cards for said networks is in many cases a requirement to operate such networks.

To make use of most of pySim's features, you will need a *programmable* SIM card, i.e. a card where you are the owner/operator and have sufficient credentials (such as the *ADM PIN*) in order to write to many if not most of the files on the card.

Such cards are, for example, available from sysmocom, a major contributor to pySim. See https://www.sysmocom.de/products/lab/sysmousim/ for more details.

pySim supports classic GSM SIM cards as well as ETSI UICC with 3GPP USIM and ISIM applications. It is easily extensible, so support for additional files, card applications, etc. can be added easily by any python developer. We do encourage you to submit your contributions to help this collaborative development project.

pySim consists of several parts:

- a python *library* containing plenty of objects and methods that can be used for writing custom programs interfacing with SIM cards.
- the [new] *interactive pySim-shell command line program*
- the [new] *pySim-trace APDU trace decoder*
- the [legacy] *pySim-prog and pySim-read tools*

1.1 pySim-shell

pySim-shell is an interactive command line shell for all kind of interactions with SIM cards, including classic GSM SIM, GSM-R SIM, UICC, USIM, ISIM, HPSIM and recently even eUICC.

If you're familiar with Unix/Linux shells: Think of it like *the bash for SIM cards*.

The pySim-shell interactive shell provides commands for

- navigating the on-card filesystem hierarchy
- authenticating with PINs such as ADM1
- CHV/PIN management (VERIFY, ENABLE, DISABLE, UNBLOCK)
- decoding of SELECT response (file control parameters)
- reading and writing of files and records in raw, hex-encoded binary format
- for most files (where related file-specific encoder/decoder classes have been developed):

- decoded reading (display file data represented in human and machine readable JSON format)
- decoded writing (encode from JSON to binary format, then write)
- if your card supports it, and you have the related privileges: resizing, creating, enabling and disabling of files
- performing GlobalPlatform operations, including establishment of Secure Channel Protocol (SCP), Installing applications, installing key material, etc.
- listing/enabling/disabling/deleting eSIM profiles on Consumer eUICC

By means of using the python cmd2 module, various useful features improve usability:

- history of commands (persistent across restarts)
- output re-direction to files on your computer
- output piping through external tools like grep
- tab completion of commands and SELECT-able files/directories
- interactive help for all commands

A typical interactive pySim workflow would look like this:

- starting the program, specifying which smart card interface to use to talk to the card
- verifying the PIN (if needed) or the ADM1 PIN in case you want to write/modify the card
- selecting on-card application dedicated files like ADF.USIM and navigating the tree of DFs
- reading and potentially modifying file contents, in raw binary (hex) or decoded JSON format

1.1.1 Video Presentation

There is a video recording of the presentation back when pySim-shell was originally released. While it is slightly dated, it should still provide a good introduction.

1.1.2 Running pySim-shell

pySim-shell has a variety of command line arguments to control

- which transport to use (how to use a reader to talk to the SIM card)
- whether to automatically verify an ADM pin (and in which format)
- whether to execute a start-up script

interactive SIM card shell

```
usage: pySim-shell.py [-h] [-d DEV] [-b BAUD] [--pcsc-shared] [-p PCSC |
                      --pcsc-regex REGEX] [--modem-device DEV]
                      [--modem-baud BAUD] [--osmocon PATH] [--apdu-trace]
                      [--script PATH] [--card_handler FILE] [--noprompt]
                      [--skip-card-init] [--verbose] [-a PIN_ADM |
                      -A PIN_ADM_HEX]
                      [--pin-adm-type {ADM1,ADM2,ADM3,ADM4,ADM5,ADM6,ADM7,ADM8,ADM9,
↪ADM10}]
                      [-e EXECUTE_COMMAND] [--csv FILE] [--pgsql FILE]
                      [--column-key FIELD:AES_KEY_HEX]
                      [command] ...
```

Positional Arguments

command A pySim-shell command that would optionally be executed at startup

command_args Optional Arguments for command

Named Arguments

--apdu-trace Trace the command/response APDUs exchanged with the card

Default: False

-e, --execute-command A pySim-shell command that will be executed at startup

Default: []

Serial Reader

Use a simple/ultra-low-cost serial reader attached to a (physical or USB/virtual) RS232 port. This doesn't work with all RS232-attached smart card readers, only with the very primitive readers following the ancient *Phoenix* or *Smart Mouse* design.

-d, --device Serial Device for SIM access

Default: "/dev/ttyUSB0"

-b, --baud Baud rate used for SIM access

Default: 9600

PC/SC Reader

Use a PC/SC card reader to talk to the SIM card. PC/SC is a standard API for how applications access smart card readers, and is available on a variety of operating systems, such as Microsoft Windows, MacOS X and Linux. Most vendors of smart card readers provide drivers that offer a PC/SC interface, if not even a generic USB CCID driver is used. You can use a tool like `pcsc_scan -r` to obtain a list of readers available on your system.

--pcsc-shared Open PC/SC reaer in SHARED access (default: EXCLUSIVE)

Default: False

-p, --pcsc-device Number of PC/SC reader to use for SIM access

--pcsc-regex Regex matching PC/SC reader to use for SIM access

AT Command Modem Reader

Talk to a SIM Card inside a mobile phone or cellular modem which is attached to this computer and offers an AT command interface including the AT+CSIM interface for Generic SIM access as specified in 3GPP TS 27.007.

--modem-device Serial port of modem for Generic SIM Access (3GPP TS 27.007)

--modem-baud Baud rate used for modem port

Default: 115200

OsmocomBB Reader

Use an OsmocomBB compatible phone to access the SIM inserted to the phone SIM slot. This will require you to run the OsmocomBB firmware inside the phone (can be ram-loaded). It also requires that you run the `osmocon` program, which provides a unix domain socket to which this reader driver can attach.

--osmocon Socket path for Calypso (e.g. Motorola C1XX) based reader (via OsmocomBB)

General Options

--script	script with pySim-shell commands to be executed automatically at start-up
--card_handler	Use automatic card handling machine
--noprompt	Run in non interactive mode Default: False
--skip-card-init	Skip all card/profile initialization Default: False
--verbose	Enable verbose logging Default: False
-a, --pin-adm	ADM PIN used for provisioning (overwrites default)
-A, --pin-adm-hex	ADM PIN used for provisioning, as hex string (16 characters long)
--pin-adm-type	Possible choices: ADM1, ADM2, ADM3, ADM4, ADM5, ADM6, ADM7, ADM8, ADM9, ADM10 Override ADM number. Default is card-model-specific, usually 1

Card Key Provider Options

--csv	Read card data from CSV file Default: "~/.osmocom/pysim/card_data.csv"
--pgsql	Read card data from PostgreSQL database (config file) Default: "~/.osmocom/pysim/card_data_pgsql.cfg"
--column-key	per-column AES transport key Default: []

1.1.3 Usage Examples

Guide: Enabling 5G SUCI

SUPI/SUCI Concealment is a feature of 5G-Standalone (SA) to encrypt the IMSI/SUPI with a network operator public key. 3GPP Specifies two different variants for this:

- SUCI calculation *in the UE*, using key data from the SIM
- SUCI calculation *on the card itself*

pySim supports writing the 5G-specific files for *SUCI calculation in the UE* on USIM cards, assuming that your cards contain the required files, and you have the privileges/credentials to write to them. This is the case using sysmocom sysmoISIM-SJA2 or any flavor of sysmoISIM-SJA5.

There is no 3GPP/ETSI standard method for configuring *SUCI calculation on the card*; pySim currently supports the vendor-specific method for the sysmoISIM-SJA5-S17).

This document describes both methods.

Technical References

This guide covers the basic workflow of provisioning SIM cards with the 5G SUCI feature. For detailed information on the SUCI feature and file contents, the following documents are helpful:

- USIM files and structure: 3GPP TS 31.102
- USIM tests (incl. file content examples): 3GPP TS 31.121
- Test keys for SUCI calculation: 3GPP TS 33.501

For specific information on sysmocom SIM cards, refer to

- the sysmoISIM-SJA5 User Manual for the current sysmoISIM-SJA5 product
- the sysmoISIM-SJA2 User Manual for the older sysmoISIM-SJA2 product

---

Enabling 5G SUCI *calculated in the UE*

In short, you can enable *SUCI calculation in the UE* with these steps:

- activate USIM **Service 124**
- make sure USIM **Service 125** is disabled
- store the public keys in **EF.SUCI_Calc_Info**
- set the **Routing Indicator** (required)

If you want to disable the feature, you can just disable USIM Service 124 (and 125) in *EF.UST*.

Admin PIN

The usual way to authenticate yourself to the card as the cellular operator is to validate the so-called ADM1 (admin) PIN. This may differ from card model/vendor to card model/vendor.

Start pySIM-shell and enter the admin PIN for your card. If you bought the SIM card from your network operator and don't have the admin PIN, you cannot change SIM contents!

Launch pySIM:

```
$ ./pySim-shell.py -p 0

Using PC/SC reader interface
Autodetected card type: sysmoISIM-SJA2
Welcome to pySim-shell!
pySIM-shell (00:MF)>
```

Enter the ADM PIN:

```
pySIM-shell (00:MF)> verify_adm XXXXXXXX
```

Otherwise, write commands will fail with SW Mismatch: Expected 9000 and got 6982.

Key Provisioning

```
pySIM-shell (00:MF)> select MF
pySIM-shell (00:MF)> select ADF.USIM
pySIM-shell (00:MF/ADF.USIM)> select DF.5GS
pySIM-shell (00:MF/ADF.USIM/DF.5GS)> select EF.SUCI_Calc_Info
```

By default, the file is present but empty:

```
pySIM-shell (00:MF/ADF.USIM/DF.5GS/EF.SUCI_Calc_Info)> read_binary_decoded
missing Protection Scheme Identifier List data object tag
9000: ffffff
```

(continued from previous page)

```
pySIM-shell (00:MF/ADF.USIM/DF.5GS/EF.Routing_Indicator)> read_binary_decoded
9000: ffffffff -> {'raw': 'ffffffff'}
```

The Routing Indicator is a four-byte file but the actual Routing Indicator goes into bytes 0 and 1 (the other bytes are reserved). To set the Routing Indicator to 0x71:

```
pySIM-shell (00:MF/ADF.USIM/DF.5GS/EF.Routing_Indicator)> update_binary 17ffffff
```

You can also set the routing indicator to **0x0**, which is *valid* and means "routing indicator not specified", leaving it to the modem.

USIM Service Table

First, check out the USIM Service Table (UST):

```
pySIM-shell (00:MF)> select MF
pySIM-shell (00:MF)> select ADF.USIM
pySIM-shell (00:MF/ADF.USIM)> select EF.UST
pySIM-shell (00:MF/ADF.USIM/EF.UST)> read_binary_decoded
9000: beff9f9de73e0408400170730000002e00000000 -> [2, 3, 4, 5, 6, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20,␣
↪21, 25, 27, 28, 29, 33, 34, 35, 38, 39, 42, 43, 44, 45, 46, 51, 60, 71, 73, 85, 86, 87, 89, 90, 93, 94, 95, 122,␣
↪123, 124, 126]
```

Table 1: From 3GPP TS 31.102

Service No.	Description
122	5GS Mobility Management Information
123	5G Security Parameters
124	Subscription identifier privacy support
125	SUCI calculation by the USIM
126	UAC Access Identities support
129	5GS Operator PLMN List

If you'd like to enable/disable any UST service:

```
pySIM-shell (00:MF/ADF.USIM/EF.UST)> ust_service_deactivate 124
pySIM-shell (00:MF/ADF.USIM/EF.UST)> ust_service_activate 124
pySIM-shell (00:MF/ADF.USIM/EF.UST)> ust_service_deactivate 125
```

In this case, UST Service 124 is already enabled and you're good to go. The sysmoISIM-SJA2 does not support on-SIM calculation, so service 125 must be disabled.

USIM Error with 5G and sysmoISIM

sysmoISIM-SJA2 come 5GS-enabled. By default however, the configuration stored in the card file-system is **not valid** for 5G networks: Service 124 is enabled, but EF.SUCI_Calc_Info and EF.Routing_Indicator are empty files (hence do not contain valid data).

At least for Qualcomm's X55 modem, this results in an USIM error and the whole modem shutting 5G down. If you don't need SUCI concealment but the smartphone refuses to connect to any 5G network, try to disable the UST service 124.

sysmoISIM-SJA5 are shipped with a more forgiving default, with valid EF.Routing_Indicator contents and disabled Service 124

SUCI calculation by the USIM

The SUCI calculation can also be performed by the USIM application on the UICC directly. The UE then uses the GET IDENTITY command (see also 3GPP TS 31.102, section 7.5) to retrieve a SUCI value.

The sysmoISIM-SJA5-S17 supports *SUCI calculation by the USIM*. The configuration is not much different to the above described configuration of *SUCI calculation in the UE*.

The main difference is how the key provisioning is done. When the SUCI calculation is done by the USIM, then the key material is not accessed by the UE. The specification (see also 3GPP TS 31.102, section 7.5.1.1), also does not specify any file or file format to store the key material. This means the exact way to perform the key provisioning is an implementation detail of the USIM card application.

In the case of sysmoISIM-SJA5-S17, the key material for *SUCI calculation by the USIM* is stored in *ADF.USIM/DF.SAIP/EF.SUCI_Calc_Info* (**not** in *ADF.USIM/DF.5GS/EF.SUCI_Calc_Info*!).

```
pySIM-shell (00:MF)> select MF
pySIM-shell (00:MF)> select ADF.USIM
pySIM-shell (00:MF/ADF.USIM)> select DF.SAIP
pySIM-shell (00:MF/ADF.USIM/DF.SAIP)> select EF.SUCI_Calc_Info
```

The file format is exactly the same as specified in 3GPP TS 31.102, section 4.4.11.8. This means the above described key provisioning procedure can be applied without any changes, except that the file location is different.

To signal to the UE that the USIM is setup up for SUCI calculation, service 125 must be enabled in addition to service 124 (see also 3GPP TS 31.102, section 5.3.48)

```
pySIM-shell (00:MF/ADF.USIM/EF.UST)> ust_service_activate 124
pySIM-shell (00:MF/ADF.USIM/EF.UST)> ust_service_activate 125
```

To verify that the SUCI calculation works as expected, it is possible to issue a GET IDENTITY command using pySim-shell:

```
select ADF.USIM
get_identity
```

The USIM should then return a SUCI TLV Data object that looks like this:

```
SUCI TLV Data Object:
↪0199f90717ff021b027a2c58ce1c6b89df088a9eb4d242596dd75746bb5f3503d2cf58a7461e4fd106e205c86f76544e9d732226a4e1
```

Guide: Installing JAVA-card applets

Almost all modern-day UICC cards have some form of JAVA-card / Sim-Toolkit support, which allows the installation of customer specific JAVA-card applets. The installation of JAVA-card applets is usually done via the standardized GlobalPlatform (GPC_SPE_034) ISD (Issuer Security Domain) application interface during the card provisioning process. (it is also possible to load JAVA-card applets in field via OTA-SMS, but that is beyond the scope of this guide). In this guide we will go through the individual steps that are required to load JAVA-card applet onto an UICC card.

Preparation

In this example we will install the CAP file HelloSTK_09122024.cap [1] on an sysmoISIM-SJA2 card. Since the interface is standardized, the exact card model does not matter.

The example applet makes use of the STK (Sim-Toolkit), so we must supply STK installation parameters. Those parameters are supplied in the form of a hexstring and should be provided by the applet manufacturer. The available parameters and their exact encoding is specified in ETSI TS 102 226, section

8.2.1.3.2.1. The installation of HelloSTK_09122024.cap [1], will require the following STK installation parameters: “010001001505000000000000000000000000”

During the installation, we also have to set a memory quota for the volatile and for the non volatile card memory. Those values also should be provided by the applet manufacturer. In this example, we will allow 255 bytes of volatile memory and 255 bytes of non volatile memory to be consumed by the applet.

To install JAVA-card applets, one must be in the possession of the key material belonging to the card. The keys are usually provided by the card manufacturer. The following example will use the following keyset:

Keyname	Keyvalue
DEK/KIK	5524F4BECFE96FB63FC29D6BAAC6058B
ENC/KIC	542C37A6043679F2F9F71116418B1CD5
MAC/KID	34F11BAC8E5390B57F4E601372339E3C

[1] https://osmocom.org/projects/cellular-infrastructure/wiki/HelloSTK

Applet Installation

To prepare the installation, a secure channel to the ISD must be established first:

```
pySIM-shell (00:MF)> select ADF.ISD
{
   "application_id": "a000000003000000",
   "proprietary_data": {
      "maximum_length_of_data_field_in_command_message": 255
   }
}
pySIM-shell (00:MF/ADF.ISD)> establish_scp02 --key-dek 5524F4BECFE96FB63FC29D6BAAC6058B --
↪key-enc 542C37A6043679F2F9F71116418B1CD5 --key-mac 34F11BAC8E5390B57F4E601372339E3C --
↪security-level 1
Successfully established a SCP02[01] secure channel
```

Warning

In case you get an “EXCEPTION of type ‘ValueError’ occurred with message: card cryptogram doesn’t match” error message, it is very likely that there is a problem with the key material. The card may lock the ISD access after a certain amount of failed tries. Carefully check the key material any try again.

When the secure channel is established, we are ready to install the applet. The installation normally is a multi step procedure, where the loading of an executable load file is announced first, then loaded and then installed in a final step. The pySim-shell command install_cap automatically takes care of those three steps.

```
pySIM-shell (SCP02[01]:00:MF/ADF.ISD)> install_cap /home/user/HelloSTK_09122024.cap --install-
↪parameters-non-volatile-memory-quota 255 --install-parameters-volatile-memory-quota 255 --install-
↪parameters-stk 010001001505000000000000000000000000
loading cap file: /home/user/HelloSTK_09122024.cap ...
parameters:
 security-domain-aid: a000000003000000
 load-file: 569 bytes
 load-file-aid: d07002ca44
 module-aid: d07002ca44900101
```

(continues on next page)

(continued from previous page)

```
 application-aid: d07002ca44900101
 install-parameters: c900ef1cc80200ffc70200ffca12010001001505000000000000000000000000
step #1: install for load...
step #2: load...
Loaded a total of 573 bytes in 3 blocks. Don't forget install_for_install (and make selectable) now!
step #3: install_for_install (and make selectable)...
done.
```

The applet is now installed on the card. We can now quit pySim-shell and remove the card from the reader and test the applet in a mobile phone. There should be a new STK application with one menu entry shown, that will greet the user when pressed.

Applet Removal

To remove the applet, we must establish a secure channel to the ISD (see above). Then we can delete the applet using the delete_card_content command.

```
pySIM-shell (SCP02[01]:00:MF/ADF.ISD)> delete_card_content D07002CA44 --delete-related-objects
```

The parameter "D07002CA44" is the load-file-AID of the applet. The load-file-AID is encoded in the .cap file and also displayed during the installation process. It is also important to note that when the applet is installed, it cannot be installed (under the same AID) again until it is removed.

Guide: Managing GP Keys

Most of today's smartcards follow the GlobalPlatform Card Specification and the included Security Domain model. UICCs and eUCCCs are no exception here.

The Security Domain acts as an on-card representative of a card authority or administrator. It is used to perform tasks like the installation of applications or the provisioning and rotation of secure channel keys. It also acts as a secure key storage and offers all kinds of cryptographic services to applications that are installed under a specific Security Domain (see also GlobalPlatform Card Specification, section 7).

In this tutorial, we will show how to work with the key material (keysets) stored inside a Security Domain and how to rotate (replace) existing keys. We will also show how to provision new keys.

Warning

Making changes to keysets requires extreme caution as misconfigured keysets may lock you out permanently. It's also strongly recommended to maintain at least one backup keyset that you can use as fallback in case the primary keyset becomes unusable for some reason.

Selecting a Security Domain

A typical smartcard, such as an UICC will have one primary Security Domain, called the Issuer Security Domain (ISD). When working with those cards, the ISD will show up in the UICC filesystem tree as *ADF.ISD* and can be selected like any other file.

```
pySIM-shell (00:MF)> select ADF.ISD
{
  "application_id": "a0000000003000000",
  "proprietary_data": {
    "maximum_length_of_data_field_in_command_message": 255
```

(continues on next page)

(continued from previous page)

```
  }
}
```

When working with eUICCs, multiple Security Domains are involved. The model is fundamentally different from the classic model with one primary Security Domain (ISD). In the case of eUICCs, an ISD-R (Issuer Security Domain - Root) and an ISD-P (Issuer Security Domain - Profile) exist (see also: GSMA SGP.02, section 2.2.1).

The ISD-P is established by the ISD-R during the profile installation and serves as a secure container for an eSIM profile. Within the ISD-P the eSIM profile establishes a dedicated Security Domain called *MNO-SD* (see also GSMA SGP.02, section 2.2.4). This *MNO-SD* is comparable to the Issuer Security Domain (ISD) we find on UICCs. The AID of *MNO-SD* is either the default AID for the Issuer Security Domain (see also GlobalPlatform, section H.1.3) or a different value specified by the provider of the eSIM profile.

Since the AID of the *MNO-SD* is not a fixed value, it is not known by *pySim-shell*. This means there will be no *ADF.ISD* file shown in the file system, but we can simply select the *ADF.ISD-R* first and then select the *MNO-SD* using a raw APDU. In the following example we assume that the default AID (a000000151000000) is used The APDU would look like this: 00a4040408 + a000000151000000 + 00

```
pySIM-shell (00:MF)> select ADF.ISD-R
{
   "application_id": "a0000005591010ffffffff8900000100",
   "proprietary_data": {
      "maximum_length_of_data_field_in_command_message": 255
   },
   "isdr_proprietary_application_template": {
      "supported_version_number": "020300"
   }
}
pySIM-shell (00:MF/ADF.ISD-R)> apdu 00a4040408a00000015100000000
SW: 9000, RESP: 6f108408a000000151000000a5049f6501ff
```

After that, the prompt will still show the *ADF.ISD-R*, but we are actually in *ADF.ISD* and the standard GlobalPlatform operations like *establish_scpXX*, *get_data*, and *put_key* should work. By doing this, we simply have tricked *pySim-shell* into making the GlobalPlatform related commands available for some other Security Domain we are not interested in. With the raw APDU we then have swapped out the Security Domain under the hood. The same workaround can be applied to any Security Domain, provided that the AID is known to the user.

Establishing a secure channel

Before we can make changes to the keysets in the currently selected Security Domain we must first establish a secure channel with that Security Domain. In the following examples we will use *SCP02* (see also GlobalPlatform Card Specification, section E.1.1) and *SCP03* (see also GlobalPlatform Card Specification – Amendment D) to establish the secure channel. *SCP02* is slightly older than *SCP03*. The main difference between the two is that *SCP02* uses 3DES while *SCP03* is based on AES.

Warning

Secure channel protocols like *SCP02* and *SCP03* may manage an error counter to count failed login attempts. This means attempting to establish a secure channel with a wrong keyset multiple times may lock you out permanently. Double check the applied keyset before attempting to establish a secure channel.

Warning

The key values used in the following examples are random key values used for illustration purposes only. Each UICC or eSIM profile is shipped with individual keys, which means that the keys used below will not work with your UICC or eSIM profile. You must replace the key values with the values you have received from your UICC vendor or eSIM profile provider.

Example: *SCP02*

In the following example, we assume that we want to establish a secure channel with the ISD of a *sysmoUSIM-SJA5* UICC. Along with the card we have received the following keyset:

Keyname	Keyvalue
ENC/KIC	F09C43EE1A0391665CC9F05AF4E0BD10
MAC/KID	01981F4A20999F62AF99988007BAF6CA
DEK/KIK	8F8AEE5CDCC5D361368BC45673D99195

This keyset is tied to the key version number KVN 122 and is configured as a DES keyset. We can use this keyset to establish a secure channel using the SCP02 Secure Channel Protocol.

```
pySIM-shell (00:MF/ADF.ISD)> establish_scp02 --key-enc F09C43EE1A0391665CC9F05AF4E0BD10 --
↪key-mac 01981F4A20999F62AF99988007BAF6CA --key-dek 8F8AEE5CDCC5D361368BC45673D99195␣
↪ --key-ver 112 --security-level 3
Successfully established a SCP02[03] secure channel
```

Example: *SCP03*

The establishment of a secure channel via SCP03 works just the same. In the following example we will establish a secure channel to the *MNO-SD* of an eSIM profile. The SCP03 keyset we use is tied to KVN 48 and looks like this:

Keyname	Keyvalue
ENC/KIC	63af517c29ad6ac6fcadfe6ac8a3c8a041d8141c7eb845ef1cba6112a325e430
MAC/KID	54b9ad6713ae922f54014ed762132e7b59bdcd2a2a6beba98fb9afe6b4df27e1
DEK/KIK	cbb933ba2389da93c86c112739cd96389139f16c6f80f7d16bf3593e407ca893

We assume that the *MNO-SD* is already selected (see above). We may now establish the SCP03 secure channel:

```
pySIM-shell (00:MF/ADF.ISD-R)> establish_scp03 --key-enc␣
↪63af517c29ad6ac6fcadfe6ac8a3c8a041d8141c7eb845ef1cba6112a325e430 --key-mac␣
↪54b9ad6713ae922f54014ed762132e7b59bdcd2a2a6beba98fb9afe6b4df27e1 --key-dek␣
↪cbb933ba2389da93c86c112739cd96389139f16c6f80f7d16bf3593e407ca893 --key-ver 48 --security-level 3
Successfully established a SCP03[03] secure channel
```

Understanding Keysets

Before making any changes to keysets, it is recommended to check the status of the currently installed keysets. To do so, we use the *get_data* command to retrieve the *key_information*. This command does not require the establishment of a secure channel. We also cannot read back the key values themselves, but we get a summary of the installed keys together with their KVN numbers, IDs, algorithm and key length values.

Example: *key_information* from a *sysmoISIM-SJA5*:

```
pySIM-shell (SCP02[03]:00:MF/ADF.ISD)> get_data key_information
{
   "key_information": [
      {
         "key_information_data": {
            "key_identifier": 1,
            "key_version_number": 112,
            "key_types": [
               {
                  "type": "des",
                  "length": 16
               }
            ]
         }
      },
      {
         "key_information_data": {
            "key_identifier": 2,
            "key_version_number": 112,
            "key_types": [
               {
                  "type": "des",
                  "length": 16
               }
            ]
         }
      },
      {
         "key_information_data": {
            "key_identifier": 3,
            "key_version_number": 112,
            "key_types": [
               {
                  "type": "des",
                  "length": 16
               }
            ]
         }
      },
      {
         "key_information_data": {
            "key_identifier": 1,
            "key_version_number": 1,
            "key_types": [
               {
                  "type": "des",
                  "length": 16
               }
            ]
         }
      },
```

(continues on next page)

(continued from previous page)


```
    {
        "key_information_data": {
            "key_identifier": 2,
            "key_version_number": 1,
            "key_types": [
                {
                    "type": "des",
                    "length": 16
                }
            ]
        }
    },
    {
        "key_information_data": {
            "key_identifier": 3,
            "key_version_number": 1,
            "key_types": [
                {
                    "type": "des",
                    "length": 16
                }
            ]
        }
    },
    {
        "key_information_data": {
            "key_identifier": 1,
            "key_version_number": 2,
            "key_types": [
                {
                    "type": "des",
                    "length": 16
                }
            ]
        }
    },
    {
        "key_information_data": {
            "key_identifier": 2,
            "key_version_number": 2,
            "key_types": [
                {
                    "type": "des",
                    "length": 16
                }
            ]
        }
    },
    {
        "key_information_data": {
            "key_identifier": 3,
            "key_version_number": 2,
```


(continues on next page)

(continued from previous page)


```
            "key_types": [
               {
                  "type": "des",
                  "length": 16
               }
            ]
         }
      },
      {
         "key_information_data": {
            "key_identifier": 1,
            "key_version_number": 47,
            "key_types": [
               {
                  "type": "des",
                  "length": 16
               }
            ]
         }
      },
      {
         "key_information_data": {
            "key_identifier": 2,
            "key_version_number": 47,
            "key_types": [
               {
                  "type": "des",
                  "length": 16
               }
            ]
         }
      },
      {
         "key_information_data": {
            "key_identifier": 3,
            "key_version_number": 47,
            "key_types": [
               {
                  "type": "des",
                  "length": 16
               }
            ]
         }
      }
   ]
}
```


Example: *key_information* from a *sysmoEUICC1-C2T*:

```
pySIM-shell (SCP03[03]:00:MF/ADF.ISD-R)> get_data key_information
{
   "key_information": [
```

(continues on next page)

(continued from previous page)


```
    {
        "key_information_data": {
            "key_identifier": 3,
            "key_version_number": 50,
            "key_types": [
                {
                    "type": "aes",
                    "length": 32
                }
            ]
        }
    },
    {
        "key_information_data": {
            "key_identifier": 2,
            "key_version_number": 50,
            "key_types": [
                {
                    "type": "aes",
                    "length": 32
                }
            ]
        }
    },
    {
        "key_information_data": {
            "key_identifier": 1,
            "key_version_number": 50,
            "key_types": [
                {
                    "type": "aes",
                    "length": 32
                }
            ]
        }
    },
    {
        "key_information_data": {
            "key_identifier": 2,
            "key_version_number": 64,
            "key_types": [
                {
                    "type": "aes",
                    "length": 16
                }
            ]
        }
    },
    {
        "key_information_data": {
            "key_identifier": 1,
            "key_version_number": 64,
```


(continues on next page)

(continued from previous page)

```
            "key_types": [
               {
                  "type": "tls_psk",
                  "length": 16
               }
            ]
         }
      }
   ]
}
```

The output from those two examples above may seem lengthy, but in order to move on and to provision own keys successfully, it is important to understand each aspect of it.

Key Version Number (KVN)

Each key is associated with a Key Version Number (KVN). Multiple keys that share the same KVN belong to the same keyset. In the first example above we can see that four keysets with KVN numbers 112, 1, 2 and 47 are provisioned. In the second example we see two keysets. One with KVN 50 and one with KVN 64.

The term "Key Version Number" is misleading as this number is not really a version number. It's actually a unique identifier for a specific keyset that also defines with which Secure Channel Protocol a key can be used. This means that the KVN is not just an arbitrary number. The following (incomplete) table gives a hint which KVN numbers may be used with which Secure Channel Protocol.

KVN range	Secure Channel Protocol
1-15	reserved for *SCP80* (OTA SMS)
17	reserved for DAP specified in ETSI TS 102 226
32-47	reserved for *SCP02*
48-63	reserved for *SCP03*
64-79	reserved for *SCP81* (GSMA SGP.02, section 2.2.5.1)
112	Token key (RSA public or DES, also used with *SCP02*)
113	Receipt key (DES)
115	DAP verification key (RS public or DES)
116	reserved for CASD
117	16-byte DES key for Ciphered Load File Data Block
255	reserved for ISD with SCP02 without SCP80 support

With that we can now understand that in the first example, the first and the last keyset is intended to be used with *SCP02* and that the second and the third keyset is intended to be used with *SCP80* (OTA SMS). In the second example we can see that the first keyset is intended to be used with *SCP03*, wheres the second should be usable with *SCP81*.

Key Identifier

Each keyset consists of a number of keys, where each key has a different Key Identifier. The Key Identifier is usually an incrementing number that starts counting at 1. The Key Identifier is used to distinguish the keys within the keyset. The exact number of keys and their attributes depends on the secure channel protocol for which the keyset is intended for. Each secure channel protocol may have its specific requirements on how many keys of which which type, length or Key Identifier have to be present.

However, almost all of the classic secure channel protocols (including *SCP02*, *SCP03* and *SCP81*) make use of the following three-key scheme:

Key Identifier	Keyname	Purpose
1	ENC/KIC	encryption/decryption
2	MAC/KID	cryptographic checksumming/signing
3	DEK/KIK	encryption/decryption of key material

In this case, all three keys share the same length and are used with the same algorithm. The key length is often used to implicitly select sub-types of an algorithm. (e.g. a 16 byte key of type *aes* is associated with *AES128*, where a 32 byte key would be associated with *AES256*).

The second example shows that different schemes are possible. The *SCP80* keyset from the second example uses a scheme that works with two keys:

Key Identifier	Keyname	Purpose
1	TLS-PSK	pre-shared key used for TLS
2	DEK/KIK	encryption/decryption of key material

It should also be noted that the order in which keysets and keys appear is an implementation detail of the UICC/eUICC O/S. The order has no influence on how a keyset is interpreted. Only the Key Version Number (KVN) and the Key Identifier matter.

Rotating a keyset

Rotating keys is one of the most basic tasks one might want to perform on an UICC/eUICC before using it productively. In the following example we will illustrate how key rotation can be done. When rotating keys, only the key itself may change. For example it is not possible to change the key length or the algorithm used (see also GlobalPlatform Card Specification, section 11.8.2.3.3). Any key of the current Security Domain can be rotated, this also includes the key that was used to establish the secure channel.

In the following example we assume that the Security Domain is selected and a secure channel is already established. We intend to rotate the keyset with KVN 112. Since this keyset uses triple DES keys with a key length of 16, we must replace it with a keyset with keys of the same nature.

The new keyset shall look like this:

Key Identifier	Keyname	Keyvalue
1	ENC/KIC	542C37A6043679F2F9F71116418B1CD5
2	MAC/KID	34F11BAC8E5390B57F4E601372339E3C
3	DEK/KIK	5524F4BECFE96FB63FC29D6BAAC6058B

When passing the keys to the *put_key* commandline, we set the Key Identifier of the first key using the *–key-id* parameter. This Key Identifier will be valid for the first key (KIC) we pass. For all consecutive keys, the Key Identifier will be incremented automatically (see also GlobalPlatform Card Specification, section 11.8.2.2). To Ensure that the new KIC, KID and KIK keys get the correct Key Identifiers, it is crucial to maintain order when passing the keys in the *–key-data* arguments. It is also important that each *–key-data* argument is preceded by a *–key-type* argument that sets the algorithm correctly (*des* in this case).

Finally we have to target the keyset we want to rotate by its KVN. The *–old-key-version-nr* argument is set to 112 as this identifies the keyset we want to rotate. The *–key-version-nr* is also set to 112 as we do not want KVN to be changed in this example. Changing the KVN while rotating a keyset is possible. In case the KVN has to change for some reason, the new KVN must be selected carefully to keep the key usable with the associated Secure Channel Protocol.

The commandline that matches the keyset we had laid out above looks like this:

```
pySIM-shell (SCP02[03]:00:MF/ADF.ISD)> put_key --key-id 1 --key-type des --key-data␣
↪542C37A6043679F2F9F71116418B1CD5 --key-type des --key-data␣
↪34F11BAC8E5390B57F4E601372339E3C --key-type des --key-data␣
↪5524F4BECFE96FB63FC29D6BAAC6058B --old-key-version-nr 112 --key-version-nr 112
```

After executing this put_key commandline, the keyset identified by KVN 122 is equipped with new keys. We can use *get_data key_information* to inspect the currently installed keysets. The output should appear unchanged as we only swapped out the keys. All other parameters, identifiers etc. should remain constant.

Warning

It is technically possible to rotate a keyset in a *non atomic* way using one *put_key* commandline for each key. However, in case the targeted keyset is the one used to establish the current secure channel, this method should not be used since, depending on the UICC/eUICC model, half-written key material may interrupt the current secure channel.

Removing a keyset

In some cases it is necessary to remove a keyset entirely. This can be done with the *delete_key* command. Here it is important to understand that *delete_key* only removes one specific key from a specific keyset. This means that you need to run a separate *delete_key* command for each key inside a keyset.

In the following example we assume that the Security Domain is selected and a secure channel is already established. We intend to remove the keyset with KVN 112. This keyset consists of three keys.

```
pySIM-shell (SCP02[03]:00:MF/ADF.ISD)> delete_key --key-ver 112 --key-id 1
pySIM-shell (SCP02[03]:00:MF/ADF.ISD)> delete_key --key-ver 112 --key-id 2
pySIM-shell (SCP02[03]:00:MF/ADF.ISD)> delete_key --key-ver 112 --key-id 3
```

To verify that the keyset has been deleted properly, we can use the *get_data key_information* command to inspect the current status of the installed keysets. We should see that the key with KVN 112 is no longer present.

Adding a keyset

In the following we will discuss how to add an entirely new keyset. The procedure is almost identical with the key rotation procedure we have already discussed and it is assumed that all details about the key rotation are understood. In this section we will go into more detail and illustrate how to provision new 3DES, *AES128* and *AES256* keysets.

It is important to keep in mind that storage space on smartcard is a precious resource. In many cases the amount of keysets that a Security Domain can store is limited. In some situations you may be forced to sacrifice one of your existing keysets in favor of a new keyset.

The main difference between key rotation and the adding of new keys is that we do not simply replace an existing key. Instead an entirely new key is programmed into the Security Domain. Therefore the *put_key* commandline will have no *–old-key-version-nr* parameter. From the commandline perspective, this is already the only visible difference from a commandline that simply rotates a keyset. Since we are writing an entirely new keyset, we are free to chose the algorithm and the key length within the parameter range permitted by the targeted secure channel protocol. Otherwise the same rules apply.

For reference, it should be mentioned that it is also possible to add or rotate keyset using multiple *put_key* commandlines. In this case one *put_key* commandline for each key is used. Each commandline will specify *–key-id* and *–key-version-nr* and one *–key-type* and *–key-data* tuple. However, when rotating or adding a keyset step-by-step, the whole process happens in a *non-atomic* way, which is less reliable. Therefore we will favor the *atomic method*

In the following examples we assume that the Security Domain is selected and a secure channel is already established.

Example: *3DES* key for *SCP02*

Let's assume we want to provision a new 3DES keyset that we can use for SCP02. The keyset shall look like this:

Key Identifier	Keyname	Keyvalue
1	ENC/KIC	542C37A6043679F2F9F71116418B1CD5
2	MAC/KID	34F11BAC8E5390B57F4E601372339E3C
3	DEK/KIK	5524F4BECFE96FB63FC29D6BAAC6058B

The keyset shall be a associated with the KVN 46. We have made sure before that KVN 46 is still unused and that this KVN number is actually suitable for SCP02 keys. As we are using 3DES, it is obvious that we have to pass 3 keys with 16 byte length.

To program the key, we may use the following commandline. As we can see, this commandline is almost the exact same as the one from the key rotation example where we were rotating a 3DES key. The only difference is that we didn't specify an old KVN number and that we have chosen a different KVN.

```
pySIM-shell (SCP02[03]:00:MF/ADF.ISD)> put_key --key-id 1 --key-type des --key-data␣
↪542C37A6043679F2F9F71116418B1CD5 --key-type des --key-data␣
↪34F11BAC8E5390B57F4E601372339E3C --key-type des --key-data␣
↪5524F4BECFE96FB63FC29D6BAAC6058B --key-version-nr 46
```

In case of success, the keyset should appear in the *key_information* among the other keysets that are already present.

```
pySIM-shell (SCP02[03]:00:MF/ADF.ISD)> get_data key_information
{
   "key_information": [
      {
         "key_information_data": {
            "key_identifier": 1,
            "key_version_number": 46,
            "key_types": [
               {
                  "type": "des",
                  "length": 16
               }
            ]
         }
      },
      {
         "key_information_data": {
            "key_identifier": 2,
            "key_version_number": 46,
            "key_types": [
               {
                  "type": "des",
                  "length": 16
               }
            ]
         }
      },
      {
         "key_information_data": {
```

(continues on next page)

(continued from previous page)

```
            "key_identifier": 3,
            "key_version_number": 46,
            "key_types": [
               {
                  "type": "des",
                  "length": 16
               }
            ]
         }
      },
      ...
   ]
}
```

Example: *AES128* key for *SCP80*

In this example we intend to provision a new *AES128* keyset that we can use with SCP80 (OTA SMS). The keyset shall look like this:

Key Identifier	Keyname	Keyvalue
1	ENC/KIC	542C37A6043679F2F9F71116418B1CD5
2	MAC/KID	34F11BAC8E5390B57F4E601372339E3C
3	DEK/KIK	5524F4BECFE96FB63FC29D6BAAC6058B

In addition to that, we want to associate this key with KVN 3. We have inspected the currently installed keysets before and made sure that KVN 3 is still unused. We are also aware that for SCP80 we may only use KVN values from 1 to 15.

For *AES128*, we specify the algorithm using the *–key-type aes* parameter. The selection between *AES128* and *AES256* is done implicitly using the key length. Since we want to use *AES128* in this case, all three keys have a length of 16 byte.

```
pySIM-shell (SCP02[03]:00:MF/ADF.ISD)> put_key --key-id 1 --key-type aes --key-data␣
↪542C37A6043679F2F9F71116418B1CD5 --key-type aes --key-data␣
↪34F11BAC8E5390B57F4E601372339E3C --key-type aes --key-data␣
↪5524F4BECFE96FB63FC29D6BAAC6058B --key-version-nr 3
```

In case of success, the keyset should appear in the *key_information* among the other keysets that are already present.

```
pySIM-shell (SCP02[03]:00:MF/ADF.ISD)> get_data key_information
{
   "key_information": [
      {
         "key_information_data": {
            "key_identifier": 1,
            "key_version_number": 3,
            "key_types": [
               {
                  "type": "aes",
                  "length": 16
               }
```

(continues on next page)

(continued from previous page)

```
            ]
         }
      },
      {
         "key_information_data": {
            "key_identifier": 2,
            "key_version_number": 3,
            "key_types": [
               {
                  "type": "aes",
                  "length": 16
               }
            ]
         }
      },
      {
         "key_information_data": {
            "key_identifier": 3,
            "key_version_number": 3,
            "key_types": [
               {
                  "type": "aes",
                  "length": 16
               }
            ]
         }
      },
      ...
   ]
}
```

Example: *AES256* key for *SCP03*

Let's assume we want to provision a new *AES256* keyset that we can use for SCP03. The keyset shall look like this:

Key Identifier	Keyname	Keyvalue
1	ENC/KIC	542C37A6043679F2F9F71116418B1CD5542C37A6043679F2F9F71116418B1CD5
2	MAC/KID	34F11BAC8E5390B57F4E601372339E3C34F11BAC8E5390B57F4E601372339E3C
3	DEK/KIK	5524F4BECFE96FB63FC29D6BAAC6058B5524F4BECFE96FB63FC29D6BAAC6058B

In addition to that, we assume that we want to associate this key with KVN 51. This KVN number falls in the range of 48 - 63 and is therefore suitable for a key that shall be usable with SCP03. We also made sure before that KVN 51 is still unused.

With that we can go ahead and make up the following commandline:

```
pySIM-shell (SCP02[03]:00:MF/ADF.ISD)> put_key --key-id 1 --key-type aes --key-data␣
↪542C37A6043679F2F9F71116418B1CD5542C37A6043679F2F9F71116418B1CD5 --key-type aes --key-
↪data 34F11BAC8E5390B57F4E601372339E3C34F11BAC8E5390B57F4E601372339E3C --key-type aes --
```

(continues on next page)

(continued from previous page)

```
↪key-data 5524F4BECFE96FB63FC29D6BAAC6058B5524F4BECFE96FB63FC29D6BAAC6058B --key-
↪version-nr 51
```

In case of success, we should see the keyset in the *key_information*

```
pySIM-shell (SCP02[03]:00:MF/ADF.ISD)> get_data key_information
{
   "key_information": [
      {
         "key_information_data": {
            "key_identifier": 1,
            "key_version_number": 51,
            "key_types": [
               {
                  "type": "aes",
                  "length": 32
               }
            ]
         }
      },
      {
         "key_information_data": {
            "key_identifier": 2,
            "key_version_number": 51,
            "key_types": [
               {
                  "type": "aes",
                  "length": 32
               }
            ]
         }
      },
      {
         "key_information_data": {
            "key_identifier": 3,
            "key_version_number": 51,
            "key_types": [
               {
                  "type": "aes",
                  "length": 32
               }
            ]
         }
      },
      ...
   ]
}
```

Example: *AES128* key for *SCP81*

In this example we will show how to provision a new *AES128* keyset for *SCP81*. We will provision this keyset under KVN 64. The keyset we intend to apply shall look like this:

Key Identifier	Keyname	Keyvalue
1	TLS-PSK	000102030405060708090a0b0c0d0e0f
2	DEK/KIK	000102030405060708090a0b0c0d0e0f

With that we can put together the following command line:

```
put_key --key-id 1 --key-type tls_psk --key-data 000102030405060708090a0b0c0d0e0f --key-type aes --key-
↪data 000102030405060708090a0b0c0d0e0f --key-version-nr 64
```

In case of success, the keyset should appear in the *key_information* as follows:

```
pySIM-shell (SCP03[03]:00:MF/ADF.ISD-R)> get_data key_information
{
   "key_information": [
      ...,
      {
         "key_information_data": {
            "key_identifier": 2,
            "key_version_number": 64,
            "key_types": [
               {
                  "type": "aes",
                  "length": 16
               }
            ]
         }
      },
      {
         "key_information_data": {
            "key_identifier": 1,
            "key_version_number": 64,
            "key_types": [
               {
                  "type": "tls_psk",
                  "length": 16
               }
            ]
         }
      }
   ]
}
```

1.1.4 Advanced Topics

Retrieving card-individual keys via CardKeyProvider

When working with a batch of cards, or more than one card in general, it is a lot of effort to manually retrieve the card-specific PIN (like ADM1) or key material (like SCP02/SCP03 keys).

To increase productivity in that regard, pySim has a concept called the *CardKeyProvider*. This is a generic mechanism by which different parts of the pySim[-shell] code can programmatically request card-specific key material from some data source (*provider*).

For example, when you want to verify the ADM1 PIN using the *verify_adm* command without providing an ADM1 value yourself, pySim-shell will request the ADM1 value for the ICCID of the card via the CardKeyProvider.

There can in theory be multiple different CardKeyProviders. You can for example develop your own CardKeyProvider that queries some kind of database for the key material, or that uses a key derivation function to derive card-specific key material from a global master key.

pySim already includes two CardKeyProvider implementations. One to retrieve key material from a CSV file (*CardKeyProviderCsv*) and a second one that allows to retrieve the key material from a PostgreSQL database (*CardKeyProviderPgsql*). Both implementations equally implement a column encryption scheme that allows to protect sensitive columns using a *transport key*

The CardKeyProviderCsv

The *CardKeyProviderCsv* allows you to retrieve card-individual key material from a CSV (comma separated value) file that is accessible to pySim.

The CSV file must have the expected column names, for example *ICCID* and *ADM1* in case you would like to use that CSV to obtain the card-specific ADM1 PIN when using the *verify_adm* command.

You can specify the CSV file to use via the *–csv* command-line option of pySim-shell. If you do not specify a CSV file, pySim will attempt to open a CSV file from the default location at *~/.osmocom/pysim/card_data.csv*, and use that, if it exists.

The *CardKeyProviderCsv* is suitable to manage small amounts of key material locally. However, if your card inventory is very large and the key material must be made available on multiple sites, the *CardKeyProviderPgsql* is the better option.

The CardKeyProviderPgsql

With the *CardKeyProviderPgsql* you can use a PostgreSQL database as storage medium. The implementation comes with a CSV importer tool that consumes the same CSV files you would normally use with the *CardKeyProviderCsv*, so you can just use your existing CSV files and import them into the database.

Requirements

The *CardKeyProviderPgsql* uses the *Psycopg* PostgreSQL database adapter (https://www.psycopg.org). *Psycopg* is not part of the default requirements of pySim-shell and must be installed separately. *Psycopg* is available as Python package under the name *psycopg2-binary*.

Setting up the database

From the perspective of the database, the *CardKeyProviderPgsql* has only minimal requirements. You do not have to create any tables in advance. An empty database and at least one user that may create, alter and insert into tables is sufficient. However, for increased reliability and as a protection against incorrect operation, the *CardKeyProviderPgsql* supports a hierarchical model with three users (or roles):

- **admin**: This should be the owner of the database. It is intended to be used for administrative tasks like adding new tables or adding new columns to existing tables. This user should not be used to insert new data into tables or to access data from within pySim-shell using the *CardKeyProviderPgsql*
- **importer**: This user is used when feeding new data into an existing table. It should only be able to insert new rows into existing tables. It should not be used for administrative tasks or to access data from within pySim-shell using the *CardKeyProviderPgsql*

- **reader**: To access data from within pySim shell using the *CardKeyProviderPgsql* the reader user is the correct one to use. This user should have no write access to the database or any of the tables.

Creating a config file

The default location for the config file is *~/.osmocom/pysim/card_data_pgsql.cfg* The file uses *yaml* syntax and should look like the example below:

```
host: "127.0.0.1"
db_name: "my_database"
table_names:
- "uicc_keys"
- "euicc_keys"
db_users:
  admin:
    name: "my_admin_user"
    pass: "my_admin_password"
  importer:
    name: "my_importer_user"
    pass: "my_importer_password"
  reader:
    name: "my_reader_user"
    pass: "my_reader_password"
```

This file is used by pySim-shell and by the importer tool. Both expect the file in the aforementioned location. In case you want to store the file in a different location you may use the *–pgsql* commandline option to provide a custom config file path.

The hostname and the database name for the PostgreSQL database is set with the *host* and *db_name* fields. The field *db_users* sets the user names and passwords for each of the aforementioned users (or roles). In case only a single admin user is used, all three entries may be populated with the same user name and password (not recommended)

The field *table_names* sets the tables that the *CardKeyProviderPgsql* shall use to query to locate card key data. You can set up as many tables as you want, *CardKeyProviderPgsql* will query them in order, one by one until a matching entry is found.

NOTE: In case you do not want to disclose the admin and the importer credentials to pySim-shell you may remove those lines. pySim-shell will only require the *reader* entry under *db_users*.

Using the Importer

Before data can be imported, you must first create a database table. Tables are created with the provided importer tool, which can be found under *contrib/csv-to-pgsql.py*. This tool is used to create the database table and read the data from the provided CSV file into the database.

As mentioned before, all CSV file formats that work with *CardKeyProviderCsv* may be used. To demonstrate how the import process works, let's assume you want to import a CSV file format that looks like the following example. Let's also assume that you didn't get the Global Platform keys from your card vendor for this batch of UICC cards, so your CSV file lacks the columns for those fields.

```
"id","imsi","iccid","acc","pin1","puk1","pin2","puk2","ki","opc","adm1"
"card1","999700000000001","8900000000000000001","0001","1111","11111111","0101","01010101",
↪"11111111111111111111111111111111","11111111111111111111111111111111","11111111"
"card2","999700000000002","8900000000000000002","0002","2222","22222222","0202","02020202",
↪"22222222222222222222222222222222","22222222222222222222222222222222","22222222"
```

(continues on next page)

(continued from previous page)

```
"card3","999700000000003","8900000000000000003","0003","3333","22222222","0303","03030303",
↪"33333333333333333333333333333333","33333333333333333333333333333333","33333333"
```

Since this is your first import, the database still lacks the table. To instruct the importer to create a new table, you may use the *–create-table* option. You also have to pick an appropriate name for the table. Any name may be chosen as long as it contains the string *uicc_keys* or *euicc_keys*, depending on the type of data (*UICC* or *eUICC*) you intend to store in the table. The creation of the table is an administrative task and can only be done with the *admin* user. The *admin* user is selected using the *–admin* switch.

```
$ PYTHONPATH=../ ./csv-to-pgsql.py --csv ./csv-to-pgsql_example_01.csv --table-name uicc_keys --
↪create-table --admin
INFO: CSV file: ./csv-to-pgsql_example_01.csv
INFO: CSV file columns: ['ID', 'IMSI', 'ICCID', 'ACC', 'PIN1', 'PUK1', 'PIN2', 'PUK2', 'KI', 'OPC',
↪'ADM1']
INFO: Using config file: /home/user/.osmocom/pysim/card_data_pgsql.cfg
INFO: Database host: 127.0.0.1
INFO: Database name: my_database
INFO: Database user: my_admin_user
INFO: New database table created: uicc_keys
INFO: Database table: uicc_keys
INFO: Database table columns: ['ICCID', 'IMSI']
INFO: Adding missing columns: ['PIN2', 'PUK1', 'PUK2', 'ACC', 'ID', 'PIN1', 'ADM1', 'KI', 'OPC']
INFO: Changes to table uicc_keys committed!
```

The importer has created a new table with the name *uicc_keys*. The table is now ready to be filled with data.

```
$ PYTHONPATH=../ ./csv-to-pgsql.py --csv ./csv-to-pgsql_example_01.csv --table-name uicc_keys
INFO: CSV file: ./csv-to-pgsql_example_01.csv
INFO: CSV file columns: ['ID', 'IMSI', 'ICCID', 'ACC', 'PIN1', 'PUK1', 'PIN2', 'PUK2', 'KI', 'OPC',
↪'ADM1']
INFO: Using config file: /home/user/.osmocom/pysim/card_data_pgsql.cfg
INFO: Database host: 127.0.0.1
INFO: Database name: my_database
INFO: Database user: my_importer_user
INFO: Database table: uicc_keys
INFO: Database table columns: ['ICCID', 'IMSI', 'PIN2', 'PUK1', 'PUK2', 'ACC', 'ID', 'PIN1', 'ADM1',
↪'KI', 'OPC']
INFO: CSV file import done, 3 rows imported
INFO: Changes to table uicc_keys committed!
```

A quick *SELECT * FROM uicc_keys;* at the PostgreSQL console should now display the contents of the CSV file you have fed into the importer.

Let's now assume that with your next batch of UICC cards your vendor includes the Global Platform keys so your CSV format changes. It may now look like this:

```
"id","imsi","iccid","acc","pin1","puk1","pin2","puk2","ki","opc","adm1","scp02_dek_1","scp02_enc_1
↪","scp02_mac_1"
"card4","999700000000004","8900000000000000004","0004","4444","44444444","0404","04040404",
↪"44444444444444444444444444444444","44444444444444444444444444444444","44444444",
↪"44444444444444444444444444444444","44444444444444444444444444444444",
↪"44444444444444444444444444444444"
"card5","999700000000005","8900000000000000005","0005","4444","55555555","0505","05050505",
```

(continues on next page)

(continued from previous page)

```
↪"55555555555555555555555555555555","55555555555555555555555555555555","55555555",
↪"55555555555555555555555555555555","55555555555555555555555555555555",
↪"55555555555555555555555555555555"
"card6","999700000000006","8900000000000000006","0006","4444","66666666","0606","06060606",
↪"66666666666666666666666666666666","66666666666666666666666666666666","66666666",
↪"66666666666666666666666666666666","66666666666666666666666666666666",
↪"66666666666666666666666666666666"
```

When importing data from an updated CSV format the database table also has to be updated. This is done using the *–update-columns* switch. Like when creating new tables, this operation also requires admin privileges, so the *–admin* switch is required again.

```
$ PYTHONPATH=../ ./csv-to-pgsql.py --csv ./csv-to-pgsql_example_02.csv --table-name uicc_keys --
↪update-columns --admin
INFO: CSV file: ./csv-to-pgsql_example_02.csv
INFO: CSV file columns: ['ID', 'IMSI', 'ICCID', 'ACC', 'PIN1', 'PUK1', 'PIN2', 'PUK2', 'KI', 'OPC',
↪'ADM1', 'SCP02_DEK_1', 'SCP02_ENC_1', 'SCP02_MAC_1']
INFO: Using config file: /home/user/.osmocom/pysim/card_data_pgsql.cfg
INFO: Database host: 127.0.0.1
INFO: Database name: my_database
INFO: Database user: my_admin_user
INFO: Database table: uicc_keys
INFO: Database table columns: ['ICCID', 'IMSI', 'PIN2', 'PUK1', 'PUK2', 'ACC', 'ID', 'PIN1', 'ADM1',
↪'KI', 'OPC']
INFO: Adding missing columns: ['SCP02_ENC_1', 'SCP02_MAC_1', 'SCP02_DEK_1']
INFO: Changes to table uicc_keys committed!
```

When the new table columns are added, the import may be continued like the first one:

```
$ PYTHONPATH=../ ./csv-to-pgsql.py --csv ./csv-to-pgsql_example_02.csv --table-name uicc_keys
INFO: CSV file: ./csv-to-pgsql_example_02.csv
INFO: CSV file columns: ['ID', 'IMSI', 'ICCID', 'ACC', 'PIN1', 'PUK1', 'PIN2', 'PUK2', 'KI', 'OPC',
↪'ADM1', 'SCP02_DEK_1', 'SCP02_ENC_1', 'SCP02_MAC_1']
INFO: Using config file: /home/user/.osmocom/pysim/card_data_pgsql.cfg
INFO: Database host: 127.0.0.1
INFO: Database name: my_database
INFO: Database user: my_importer_user
INFO: Database table: uicc_keys
INFO: Database table columns: ['ICCID', 'IMSI', 'PIN2', 'PUK1', 'PUK2', 'ACC', 'ID', 'PIN1', 'ADM1',
↪'KI', 'OPC', 'SCP02_ENC_1', 'SCP02_MAC_1', 'SCP02_DEK_1']
INFO: CSV file import done, 3 rows imported
INFO: Changes to table uicc_keys committed!
```

On the PostgreSQL console a *SELECT * FROM uicc_keys;* should now show the imported data with the added columns. All important data should now also be available from within pySim-shell via the *CardKeyProviderPgsql*.

Column-Level CSV encryption

pySim supports column-level CSV encryption. This feature will make sure that your key material is not stored in plaintext in the CSV file (or database).

The encryption mechanism uses AES in CBC mode. You can use any key length permitted by AES (128/192/256 bit).

Following GSMA FS.28, the encryption works on column level. This means different columns can be decrypted using

different key material. This means that leakage of a column encryption key for one column or set of columns (like a specific security domain) does not compromise various other keys that might be stored in other columns.

You can specify column-level decryption keys using the *–csv-column-key* command line argument. The syntax is *FIELD:AES_KEY_HEX*, for example:

pySim-shell.py –csv-column-key SCP03_ENC_ISDR:000102030405060708090a0b0c0d0e0f

In order to avoid having to repeat the column key for each and every column of a group of keys within a keyset, there are pre-defined column group aliases, which will make sure that the specified key will be used by all columns of the set:

- *UICC_SCP02* is a group alias for *UICC_SCP02_KIC1*, *UICC_SCP02_KID1*, *UICC_SCP02_KIK1*
- *UICC_SCP03* is a group alias for *UICC_SCP03_KIC1*, *UICC_SCP03_KID1*, *UICC_SCP03_KIK1*
- *SCP03_ECASD* is a group alias for *SCP03_ENC_ECASD*, *SCP03_MAC_ECASD*, *SCP03_DEK_ECASD*
- *SCP03_ISDA* is a group alias for *SCP03_ENC_ISDA*, *SCP03_MAC_ISDA*, *SCP03_DEK_ISDA*
- *SCP03_ISDR* is a group alias for *SCP03_ENC_ISDR*, *SCP03_MAC_ISDR*, *SCP03_DEK_ISDR*

NOTE: When using *CardKeyProviderPqsl*, the input CSV files must be encrypted before import.

Field naming

- For look-up of UICC/SIM/USIM/ISIM or eSIM profile specific key material, pySim uses the *ICCID* field as lookup key.
- For look-up of eUICC specific key material (like SCP03 keys for the ISD-R, ECASD), pySim uses the *EID* field as lookup key.

As soon as the CardKeyProvider finds a line (row) in your CSV file (or database) where the ICCID or EID match, it looks for the column containing the requested data.

ADM PIN

The *verify_adm* command will attempt to look up the *ADM1* column indexed by the ICCID of the SIM/UICC.

SCP02 / SCP03

SCP02 and SCP03 each use key triplets consisting of ENC, MAC and DEK keys. For more details, see the applicable GlobalPlatform specifications.

If you do not want to manually enter the key material for each specific card as arguments to the *establish_scp02* or *establish_scp03* commands, you can make use of the *–key-provider-suffix* option. pySim uses this suffix to compose the column names for the CardKeyProvider as follows.

- *SCP02_ENC_* + suffix for the SCP02 ciphering key
- *SCP02_MAC_* + suffix for the SCP02 MAC key
- *SCP02_DEK_* + suffix for the SCP02 DEK key
- *SCP03_ENC_* + suffix for the SCP03 ciphering key
- *SCP03_MAC_* + suffix for the SCP03 MAC key
- *SCP03_DEK_* + suffix for the SCP03 DEK key

So for example, if you are using a command like *establish_scp03 –key-provider-suffix ISDR*, then the column names for the key material look-up are *SCP03_ENC_ISDR*, *SCP03_MAC_ISDR* and *SCP03_DEK_ISDR*, respectively.

The identifier used for look-up is determined by the definition of the Security Domain. For example, the eUICC ISD-R and ECASD will use the EID of the eUICC. On the other hand, the ISD-P of an eSIM or the ISD of an UICC will use the ICCID.

Remote access to an UICC/eUICC

To access a card with pySim-shell, it is not strictly necessary to have physical access to it. There are solutions that allow remote access to UICC/eUICC cards. In this section we will give a brief overview.

osmo-remsim

osmo-remsim is a suite of software programs enabling physical/geographic separation of a cellular phone (or modem) on the one hand side and the UICC/eUICC card on the other side.

Using osmo-remsim, you can operate an entire fleet of modems/phones, as well as banks of SIM cards and dynamically establish or remove the connections between modems/phones and cards.

To access remote cards with pySim-shell via osmo-remseim (RSPRO), the provided libifd_remsim_client would be used to provide a virtual PC/SC reader on the local machine. pySim-shell can then access this reader like any other PC/SC reader.

More information on osmo-remsim can be found under:

- https://osmocom.org/projects/osmo-remsim/wiki
- https://ftp.osmocom.org/docs/osmo-remsim/master/osmo-remsim-usermanual.pdf

Android APDU proxy

Android APDU proxy is an Android app that provides a bridge between a host computer and the UICC/eUICC slot of an Android smartphone.

The APDU proxy connects to VPCD server that runs on the remote host (in this case the local machine where pySim-shell is running). The VPCD server then provides a virtual PC/SC reader, that pySim-shell can access like any other PC/SC reader.

On the Android side the UICC/eUICC is accessed via OMAPI (Open Mobile API), which is available in Android since API level Android 8 (API level 29).

More information Android APDU proxy can be found under:

- https://gitea.osmocom.org/sim-card/android-apdu-proxy

1.1.5 cmd2 basics

As pySim-shell is built upon cmd2, some generic cmd2 commands/features are available. You may want to check out the cmd2 Builtin commands to learn about those.

1.1.6 pySim commands

Commands in this category are pySim specific; they do not have a 1:1 correspondence to ISO 7816 or 3GPP commands. Mostly they will operate either only on local (in-memory) state, or execute a complex sequence of card-commands.

desc

Display human readable file description for the currently selected file.

dir

```
usage: build.py [-h] [--fids] [--names] [--apps] [--all]
```

Named Arguments

--fids	Show file identifiers
	Default: False
--names	Show file names
	Default: False
--apps	Show applications
	Default: False
--all	Show all selectable identifiers and names
	Default: False

Example:

```
pySIM-shell (00:MF)> dir
MF
3f00
 ..          ADF.USIM    DF.SYSTEM   EF.DIR      EF.UMPC
 ADF.ARA-M   DF.EIRENE   DF.TELECOM  EF.ICCID    MF
 ADF.ISIM    DF.GSM      EF.ARR      EF.PL
14 files
```

export

```
usage: build.py [-h] [--filename FILENAME] [--json]
```

Named Arguments

--filename	only export specific file
--json	export as JSON (less reliable)
	Default: False

Please note that *export* works relative to the current working directory, so if you are in *MF*, then the export will contain all known files on the card. However, if you are in *ADF.ISIM*, only files below that ADF will be part of the export.

Furthermore, it is strongly advised to first enter the ADM1 pin (*verify_adm*) to maximize the chance of having permission to read all/most files.

Example:

```
pySIM-shell (00:MF)> export --json > /tmp/export.json
EXCEPTION of type 'RuntimeError' occurred with message: 'unable to export 50 elementary file(s) and␣
↪2 dedicated file(s), also had to stop early due to exception:6e00: ARA-M - Invalid class'
To enable full traceback, run the following command: 'set debug true'
pySIM-shell (00:MF)>
```

The exception above is more or less expected. It just means that 50 files which are defined (most likely as optional files in some later 3GPP release) were not found on the card, or were invalidated/disabled when trying to SELECT them.

fsdump

```
usage: build.py [-h] [--filename FILENAME] [--json]
```

Named Arguments

--filename	only export specific (named) file
--json	export file contents as JSON (less reliable)
	Default: False

Please note that *fsdump* works relative to the current working directory, so if you are in *MF*, then the dump will contain all known files on the card. However, if you are in *ADF.ISIM*, only files below that ADF will be part of the dump.

Furthermore, it is strongly advised to first enter the ADM1 pin (*verify_adm*) to maximize the chance of having permission to read all/most files.

One use case for this is to systematically analyze the differences between the contents of two cards. To do this, you can create fsdumps of the two cards, and then use some general-purpose JSON diffing tool like *jycm –show* (see https://github.com/eggachecat/jycm).

Example:

```
pySIM-shell (00:MF)> fsdump > /tmp/fsdump.json
pySIM-shell (00:MF)>
```

tree

Display a tree of the card filesystem. It is important to note that this displays a tree of files that might potentially exist (based on the card profile). In order to determine if a given file really exists on a given card, you have to try to select that file.

Example:

```
pySIM-shell (00:MF)> tree
EF.DIR                2f00 Application Directory
EF.ICCID              2fe2 ICC Identification
EF.PL                 2f05 Preferred Languages
EF.ARR                2f06 Access Rule Reference
EF.UMPC               2f08 UICC Maximum Power Consumption
DF.TELECOM               7f10 None
  EF.ADN              6f3a Abbreviated Dialing Numbers
...
```

verify_adm

```
usage: build.py [-h] [--pin-is-hex]
                [--adm-type {ADM1,ADM2,ADM3,ADM4,ADM5,ADM6,ADM7,ADM8,ADM9,ADM10}]
                [ADM]
```

Positional Arguments

ADM	ADM pin value. If none given, CSV file will be queried

Named Arguments

--pin-is-hex ADM pin value is specified as hex-string (not decimal)

Default: False

--adm-type Possible choices: ADM1, ADM2, ADM3, ADM4, ADM5, ADM6, ADM7, ADM8, ADM9, ADM10

Override ADM number. Default is card-model-specific, usually 1

Example (successful):

```
pySIM-shell (00:MF)> verify_adm 11111111
pySIM-shell (00:MF)>
```

In the above case, the ADM was successfully verified. Please make always sure to use the correct ADM1 for the specific card you have inserted! If you present a wrong ADM1 value several times consecutively, your card ADM1 will likely be permanently locked, meaning you will never be able to reach ADM1 privilege level. For sysmoUSIM/ISIM products, three consecutive wrong ADM1 values will lock the ADM1.

Example (erroneous):

```
pySIM-shell (00:MF)> verify_adm 1
EXCEPTION of type 'RuntimeError' occurred with message: 'Failed to verify chv_no 0x0A with code␣
↪0x31FFFFFFFFFFFFFF, 2 tries left.'
To enable full traceback, run the following command: 'set debug true'
```

If you frequently work with the same set of cards that you need to modify using their ADM1, you can put a CSV file with those cards ICCID + ADM1 values into a CSV (comma separated value) file at ~/.osmocom/pysim/card_data.csv. In this case, you can use the verify_adm command *without specifying an ADM1 value*.

Example (successful):

```
pySIM-shell (00:MF)> verify_adm
found ADM-PIN '11111111' for ICCID '8988211900000000512'
pySIM-shell (00:MF)>
```

In this case, the CSV file contained a record for the ICCID of the card (11111111) and that value was used to successfully verify ADM1.

Example (erroneous):

```
pySIM-shell (00:MF)> verify_adm
EXCEPTION of type 'ValueError' occurred with message: 'cannot find ADM-PIN for ICCID
↪'8988211900000000512''
To enable full traceback, run the following command: 'set debug true'
```

In this case there was no record for the ICCID of the card in the CSV file.

reset

Perform card reset and display the card ATR.

Example:

```
pySIM-shell (00:MF)> reset
Card ATR: 3b9f96801f878031e073fe211b674a357530350259c4
pySIM-shell (00:MF)> reset
```

intro

[Re-]Display the introductory banner

Example:

```
pySIM-shell (00:MF)> intro
Welcome to pySim-shell!
(C) 2021-2023 by Harald Welte, sysmocom - s.f.m.c. GmbH and contributors
Online manual available at https://downloads.osmocom.org/docs/pysim/master/html/shell.html
```

equip

Equip pySim-shell with a card; particularly useful if the program was started before a card was present, or after a card has been replaced by the user while pySim-shell was kept running.

bulk_script

```
usage: build.py [-h] [--halt_on_error] [--tries TRIES]
                [--on_stop_action ON_STOP_ACTION]
                [--pre_card_action PRE_CARD_ACTION]
                SCRIPT_PATH
```

Positional Arguments

SCRIPT_PATH	path to the script file

Named Arguments

--halt_on_error	stop card handling if an exception occurs
	Default: False
--tries	how many tries before trying the next card
	Default: 2
--on_stop_action	commandline to execute when card handling has stopped
--pre_card_action	commandline to execute before actually talking to the card

echo

```
usage: build.py [-h] STRING [STRING ...]
```

Positional Arguments

STRING	string to echo on the shell

apdu

```
usage: build.py [-h] [--expect-sw EXPECT_SW]
                [--expect-response-regex EXPECT_RESPONSE_REGEX] [--raw]
                APDU
```

Positional Arguments

APDU APDU as hex string (see also: ISO/IEC 7816-3, section 12.1

Named Arguments

--expect-sw expect a specified status word

--expect-response-regex match response against regex

--raw Bypass the logical channel (and secure channel)

Default: False

Example:

```
pySIM-shell (00:MF)> apdu 00a40400023f00
SW: 6700
```

In the above case the raw APDU hex-string 00a40400023f00 was sent to the card, to which it responded with status word 6700. Keep in mind that pySim-shell has no idea what kind of raw commands you are sending to the card, and it hence is unable to synchronize its internal state (such as the currently selected file) with the card. The use of this command should hence be constrained to commands that do not have any high-level support in pySim-shell yet.

1.1.7 ISO7816 commands

This category of commands relates to commands that originate in the ISO 7861-4 specifications, most of them have a 1:1 resemblance in the specification.

select

The select command is used to select a file, either by its FID, AID or by its symbolic name.

Try select with tab-completion to get a list of all current selectable items:

```
pySIM-shell (00:MF)> select
..              2fe2            a0000000871004  EF.ARR          MF
2f00            3f00            ADF.ISIM        EF.DIR
2f05            7f10            ADF.USIM        EF.ICCID
2f06            7f20            DF.GSM          EF.PL
2f08            a0000000871002  DF.TELECOM      EF.UMPC
```

Use select with a specific FID or name to select the new file.

This will

- output the [JSON decoded, if possible] select response
- change the prompt to the newly selected file
- enable any commands specific to the newly-selected file

```
pySIM-shell (00:MF)> select ADF.USIM
{
   "file_descriptor": {
      "file_descriptor_byte": {
         "shareable": true,
         "file_type": "df",
         "structure": "no_info_given"
```

(continues on next page)

(continued from previous page)

```
        }
    },
    "df_name": "A0000000871002FFFFFFFF8907090000",
    "proprietary_info": {
        "uicc_characteristics": "71",
        "available_memory": 101640
    },
    "life_cycle_status_int": "operational_activated",
    "security_attrib_compact": "00",
    "pin_status_template_do": {
        "ps_do": "70",
        "key_reference": 11
    }

}
pySIM-shell (00:MF/ADF.USIM)>
```

status

The status command [re-]obtains the File Control Template of the currently-selected file and print its decoded output.

Example:

```
pySIM-shell (00:MF/ADF.ISIM)> status
{
    "file_descriptor": {
        "file_descriptor_byte": {
            "shareable": true,
            "file_type": "df",
            "structure": "no_info_given"
        },
        "record_len": null,
        "num_of_rec": null
    },
    "file_identifier": "ff01",
    "df_name": "a0000000871004ffffffff8907090000",
    "proprietary_information": {
        "uicc_characteristics": "71",
        "available_memory": 101640
    },
    "life_cycle_status_integer": "operational_activated",
    "security_attrib_compact": "00",
    "pin_status_template_do": {
        "ps_do": "70",
        "key_reference": 11
    }
}
```

change_chv

```
usage: build.py [-h]
                [--pin-type {PIN1,PIN2,PIN3,PIN4,PIN5,PIN6,PIN7,PIN8,2PIN1,2PIN2,2PIN3,2PIN4,2PIN5,
↪2PIN6,2PIN7,2PIN8} |
                --pin-nr PIN_NR]
                [NEWPIN] [PIN]
```

Positional Arguments

NEWPIN PIN code value. If none given, CSV file will be queried

PIN PIN code value. If none given, CSV file will be queried

Named Arguments

--pin-type Possible choices: PIN1, PIN2, PIN3, PIN4, PIN5, PIN6, PIN7, PIN8, 2PIN1, 2PIN2, 2PIN3, 2PIN4, 2PIN5, 2PIN6, 2PIN7, 2PIN8

Specifiy pin type (default is PIN1)

--pin-nr PIN Number, 1=PIN1, 0x81=2PIN1 or custom value (see also TS 102 221, Table 9.3")

Default: 1

disable_chv

```
usage: build.py [-h]
                [--pin-type {PIN1,PIN2,PIN3,PIN4,PIN5,PIN6,PIN7,PIN8,2PIN1,2PIN2,2PIN3,2PIN4,2PIN5,
↪2PIN6,2PIN7,2PIN8} |
                --pin-nr PIN_NR]
                [PIN]
```

Positional Arguments

PIN PIN code value. If none given, CSV file will be queried

Named Arguments

--pin-type Possible choices: PIN1, PIN2, PIN3, PIN4, PIN5, PIN6, PIN7, PIN8, 2PIN1, 2PIN2, 2PIN3, 2PIN4, 2PIN5, 2PIN6, 2PIN7, 2PIN8

Specifiy pin type (default is PIN1)

--pin-nr PIN Number, 1=PIN1, 0x81=2PIN1 or custom value (see also TS 102 221, Table 9.3")

Default: 1

enable_chv

```
usage: build.py [-h]
                [--pin-type {PIN1,PIN2,PIN3,PIN4,PIN5,PIN6,PIN7,PIN8,2PIN1,2PIN2,2PIN3,2PIN4,2PIN5,
↪2PIN6,2PIN7,2PIN8} |
                --pin-nr PIN_NR]
                [PIN]
```

Positional Arguments

PIN PIN code value. If none given, CSV file will be queried

Named Arguments

--pin-type Possible choices: PIN1, PIN2, PIN3, PIN4, PIN5, PIN6, PIN7, PIN8, 2PIN1, 2PIN2, 2PIN3, 2PIN4, 2PIN5, 2PIN6, 2PIN7, 2PIN8

Specifiy pin type (default is PIN1)

--pin-nr PIN Number, 1=PIN1, 0x81=2PIN1 or custom value (see also TS 102 221, Table 9.3")

Default: 1

unblock_chv

```
usage: build.py [-h]
                [--pin-type {PIN1,PIN2,PIN3,PIN4,PIN5,PIN6,PIN7,PIN8,2PIN1,2PIN2,2PIN3,2PIN4,2PIN5,
↪2PIN6,2PIN7,2PIN8} |
                --pin-nr PIN_NR]
                [PUK] [NEWPIN]
```

Positional Arguments

PUK PUK code value. If none given, CSV file will be queried

NEWPIN PIN code value. If none given, CSV file will be queried

Named Arguments

--pin-type Possible choices: PIN1, PIN2, PIN3, PIN4, PIN5, PIN6, PIN7, PIN8, 2PIN1, 2PIN2, 2PIN3, 2PIN4, 2PIN5, 2PIN6, 2PIN7, 2PIN8

Specifiy pin type (default is PIN1)

--pin-nr PIN Number, 1=PIN1, 0x81=2PIN1 or custom value (see also TS 102 221, Table 9.3")

Default: 1

verify_chv

```
usage: build.py [-h]
                [--pin-type {PIN1,PIN2,PIN3,PIN4,PIN5,PIN6,PIN7,PIN8,2PIN1,2PIN2,2PIN3,2PIN4,2PIN5,
↪2PIN6,2PIN7,2PIN8} |
                --pin-nr PIN_NR]
                [PIN]
```

Positional Arguments

PIN PIN code value. If none given, CSV file will be queried

Named Arguments

--pin-type Possible choices: PIN1, PIN2, PIN3, PIN4, PIN5, PIN6, PIN7, PIN8, 2PIN1, 2PIN2, 2PIN3, 2PIN4, 2PIN5, 2PIN6, 2PIN7, 2PIN8

Specifiy pin type (default is PIN1)

--pin-nr PIN Number, 1=PIN1, 0x81=2PIN1 or custom value (see also TS 102 221, Table 9.3")

Default: 1

deactivate_file

Deactivate the currently selected file. A deactivated file can no longer be accessed for any further operation (such as selecting and subsequently reading or writing).

Any access to a file that is deactivated will trigger the error *SW 6283 'Selected file invalidated/disabled'*

In order to re-access a deactivated file, you need to activate it again, see the *activate_file* command below. Note that for *deactivation* the to-be-deactivated EF must be selected, but for *activation*, the DF above the to-be-activated EF must be selected!

This command sends a DEACTIVATE FILE APDU to the card (used to be called INVALIDATE in TS 11.11 for classic SIM).

activate_file

```
usage: build.py [-h] NAME
```

Positional Arguments

NAME File name or FID of file to activate

open_channel

```
usage: build.py [-h] {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15}
```

Positional Arguments

chan_nr Possible choices: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

Channel Number

Default: 1

close_channel

```
usage: build.py [-h] {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15}
```

Positional Arguments

chan_nr Possible choices: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

Channel Number

Default: 1

switch_channel

```
usage: build.py [-h] {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15}
```

Positional Arguments

chan_nr Possible choices: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

Channel Number

Default: 0

1.1.8 TS 102 221 commands

These are commands as specified in ETSI TS 102 221, the core UICC specification.

suspend_uicc

This command allows you to perform the SUSPEND UICC command on the card. This is a relatively recent power-saving addition to the UICC specifications, allowing for suspend/resume while maintaining state, as opposed to a full power-off (deactivate) and power-on (activate) of the card.

The pySim command just sends that SUSPEND UICC command and doesn't perform the full related sequence including the electrical power down.

```
usage: build.py [-h] [--min-duration-secs MIN_DURATION_SECS]
                [--max-duration-secs MAX_DURATION_SECS]
```

Named Arguments

--min-duration-secs Proposed minimum duration of suspension

Default: 60

--max-duration-secs Proposed maximum duration of suspension

Default: 86400

resume_uicc

This command allows you to perform the SUSPEND UICC command for the RESUME operation on the card.

Suspend/Resume is a relatively recent power-saving addition to the UICC specifications, allowing for suspend/resume while maintaining state, as opposed to a full power-off (deactivate) and power-on (activate) of the card.

The pySim command just sends that SUSPEND UICC (RESUME) command and doesn't perform the full related sequence including the electrical power down.

```
usage: build.py [-h] TOKEN
```

Positional Arguments

TOKEN Token provided during SUSPEND

terminal_capability

This command allows you to perform the TERMINAL CAPABILITY command towards the card.

TS 102 221 specifies the TERMINAL CAPABILITY command using which the terminal (Software + hardware talking to the card) can expose their capabilities. This is also used in the eUICC universe to let the eUICC know which features are supported.

```
usage: build.py [-h] [--used-supply-voltage-class {a,b,c,d,e}]
                [--maximum-available-power-supply MAXIMUM_AVAILABLE_POWER_SUPPLY]
                [--actual-used-freq-100k ACTUAL_USED_FREQ_100K]
                [--extended-logical-channel] [--uicc-clf] [--lui-d] [--lpd-d]
                [--lds-d] [--lui-e-scws] [--metadata-update-alerting]
                [--enterprise-capable-device] [--lui-e-e4e] [--lpr]
```

Terminal Power Supply

--used-supply-voltage-class Possible choices: a, b, c, d, e

Actual used Supply voltage class

--maximum-available-power-supply Maximum available power supply of the terminal

--actual-used-freq-100k Actual used clock frequency (in units of 100kHz)

Extended logical channels terminal support

--extended-logical-channel Extended Logical Channel supported

Default: False

Additional interfaces support

--uicc-clf Local User Interface in the Device (LUId) supported

Default: False

Additional Terminal capability indications related to eUICC

--lui-d Local User Interface in the Device (LUId) supported

Default: False

--lpd-d Local Profile Download in the Device (LPDd) supported

Default: False

--lds-d Local Discovery Service in the Device (LPDd) supported

Default: False

--lui-e-scws LUIe based on SCWS supported

Default: False

--metadata-update-alerting Metadata update alerting supported

Default: False

--enterprise-capable-device Enterprise Capable Device

Default: False

--lui-e-e4e LUIe using E4E (ENVELOPE tag E4) supported

Default: False

--lpr LPR (LPA Proxy) supported

Default: False

1.1.9 Linear Fixed EF commands

These commands become enabled only when your currently selected file is of *Linear Fixed EF* type.

read_record

```
usage: build.py [-h] [--count COUNT] RECORD_NR
```

Positional Arguments

RECORD_NR Number of record to be read

Named Arguments

--count Number of records to be read, beginning at record_nr

Default: 1

read_record_decoded

```
usage: build.py [-h] [--oneline] RECORD_NR
```

Positional Arguments

RECORD_NR Number of record to be read

Named Arguments

--oneline No JSON pretty-printing, dump as a single line

Default: False

If this command fails, it means that the record is not decodable, and you should use the *read_record* command and proceed with manual decoding of the contents.

read_records

```
usage: build.py [-h]
```

read_records_decoded

```
usage: build.py [-h] [--oneline]
```

Named Arguments

--oneline No JSON pretty-printing, dump as a single line

Default: False

If this command fails, it means that the record[s] are not decodable, and you should use the *read_records* command and proceed with manual decoding of the contents.

update_record

```
usage: build.py [-h] RECORD_NR DATA
```

Positional Arguments

RECORD_NR Number of record to be read

DATA Data bytes (hex format) to write

update_record_decoded

```
usage: build.py [-h] [--json-path JSON_PATH] RECORD_NR data
```

Positional Arguments

RECORD_NR Number of record to be read

data Abstract data (JSON format) to write

Named Arguments

--json-path JSON path to modify specific element of record only

If this command fails, it means that the record is not encodable; please check your input and/or use the raw *update_record* command.

edit_record_decoded

```
usage: build.py [-h] RECORD_NR
```

Positional Arguments

RECORD_NR Number of record to be edited

This command will read the selected record, decode it to its JSON representation, save that JSON to a temporary file on your computer, and launch your configured text editor.

You may then perform whatever modifications to the JSON representation, save + leave your text editor.

Afterwards, the modified JSON will be re-encoded to the binary format, and the result written back to the record on the SIM card.

This allows for easy interactive modification of records.

If this command fails before the editor is spawned, it means that the current record contents is not decodable, and you should use the *update_record_decoded* or *update_record* command.

If this command fails after making your modifications in the editor, it means that the new file contents is not encodable; please check your input and/or use the raw *update_record* command.

decode_hex

```
usage: build.py [-h] [--oneline] HEXSTR
```

Positional Arguments

HEXSTR Hex-string of encoded data to decode

Named Arguments

--oneline No JSON pretty-printing, dump as a single line

Default: False

1.1.10 Transparent EF commands

These commands become enabled only when your currently selected file is of *Transparent EF* type.

read_binary

```
usage: build.py [-h] [--offset OFFSET] [--length LENGTH]
```

Named Arguments

--offset Byte offset for start of read

Default: 0

--length Number of bytes to read

read_binary_decoded

```
usage: build.py [-h] [--oneline]
```

Named Arguments

--oneline No JSON pretty-printing, dump as a single line

Default: False

If this command fails, it means that the file is not decodable, and you should use the *read_binary* command and proceed with manual decoding of the contents.

update_binary

```
usage: build.py [-h] [--offset OFFSET] DATA
```

Positional Arguments

DATA Data bytes (hex format) to write

Named Arguments

--offset Byte offset for start of read

Default: 0

update_binary_decoded

```
usage: build.py [-h] [--json-path JSON_PATH] DATA
```

Positional Arguments

DATA Abstract data (JSON format) to write

Named Arguments

--json-path JSON path to modify specific element of file only

In normal operation, update_binary_decoded needs a JSON document representing the entire file contents as input. This can be inconvenient if you want to keep 99% of the content but just toggle one specific parameter. That's where the JSONpath support comes in handy: You can specify a JSONpath to an element inside the document as well as a new value for tat field:

The below example demonstrates this by modifying the ciphering indicator field within EF.AD:

```
pySIM-shell (00:MF/ADF.USIM/EF.AD)> read_binary_decoded

{
   "ms_operation_mode": "normal_and_specific_facilities",
   "additional_info": {
      "ciphering_indicator": false,
      "csg_display_control": false,
      "prose_services": false,
      "extended_drx": true
   },
   "rfu": 0,
   "mnc_len": 2,
   "extensions": "ff"
}
pySIM-shell (00:MF/ADF.USIM/EF.AD)> update_binary_decoded --json-path additional_info.
↪ciphering_indicator true
"01000902ff"
pySIM-shell (00:MF/ADF.USIM/EF.AD)> read_binary_decoded
{
   "ms_operation_mode": "normal_and_specific_facilities",
   "additional_info": {
      "ciphering_indicator": true,
      "csg_display_control": false,
      "prose_services": false,
      "extended_drx": true
   },
   "rfu": 0,
   "mnc_len": 2,
   "extensions": "ff"
}
```

If this command fails, it means that the file is not encodable; please check your input and/or use the raw *update_binary* command.

edit_binary_decoded

This command will read the selected binary EF, decode it to its JSON representation, save that JSON to a temporary file on your computer, and launch your configured text editor.

You may then perform whatever modifications to the JSON representation, save + leave your text editor.

Afterwards, the modified JSON will be re-encoded to the binary format, and the result written to the SIM card.

This allows for easy interactive modification of file contents.

If this command fails before the editor is spawned, it means that the current file contents is not decodable, and you should use the *update_binary_decoded* or *update_binary* command.

If this command fails after making your modifications in the editor, it means that the new file contents is not encodable; please check your input and/or use the raw *update_binary* command.

decode_hex

```
usage: build.py [-h] [--oneline] HEXSTR
```

Positional Arguments

HEXSTR Hex-string of encoded data to decode

Named Arguments

--oneline No JSON pretty-printing, dump as a single line

Default: False

1.1.11 BER-TLV EF commands

BER-TLV EFs are files that contain BER-TLV structured data. Every file can contain any number of variable-length IEs (DOs). The tag within a BER-TLV EF must be unique within the file.

The commands below become enabled only when your currently selected file is of *BER-TLV EF* type.

retrieve_tags

Retrieve a list of all tags present in the currently selected file.

retrieve_data

```
usage: build.py [-h] TAG
```

Positional Arguments

TAG BER-TLV Tag of value to retrieve

set_data

```
usage: build.py [-h] TAG data
```

Positional Arguments

TAG	BER-TLV Tag of value to set
data	Data bytes (hex format) to write

del_data

```
usage: build.py [-h] TAG
```

Positional Arguments

TAG	BER-TLV Tag of value to set

1.1.12 USIM commands

These commands are available only while ADF.USIM (or ADF.ISIM, respectively) is selected.

authenticate

```
usage: build.py [-h] RAND AUTN
```

Positional Arguments

RAND	Random challenge
AUTN	Authentication Nonce

terminal_profile

```
usage: build.py [-h] PROFILE
```

Positional Arguments

PROFILE	Hexstring of encoded terminal profile

envelope

```
usage: build.py [-h] PAYLOAD
```

Positional Arguments

PAYLOAD	Hexstring of encoded payload to ENVELOPE

envelope_sms

```
usage: build.py [-h] TPDU
```

Positional Arguments

TPDU Hexstring of encoded SMS TPDU

get_identity

```
usage: build.py [-h] [--nswo-context]
```

Named Arguments

--nswo-context use SUCI 5G Non-Seamless WLAN Offload context

Default: False

1.1.13 File-specific commands

These commands are valid only if the respective file is currently selected. They perform some operation that's specific to this file only.

EF.ARR: read_arr_record

Read one EF.ARR record in flattened, human-friendly form.

EF.ARR: read_arr_records

Read + decode all EF.ARR records in flattened, human-friendly form.

DF.GSM/EF.SST: sst_service_allocate

Mark a given single service as allocated in EF.SST. Requires service number as argument.

DF.GSM/EF.SST: sst_service_activate

Mark a given single service as activated in EF.SST. Requires service number as argument.

DF.GSM/EF.SST: sst_service_deallocate

Mark a given single service as deallocated in EF.SST. Requires service number as argument.

DF.GSM/EF.SST: sst_service_deactivate

Mark a given single service as deactivated in EF.SST. Requires service number as argument.

ADF.USIM/EF.EST: est_service_enable

Enables a single service in EF.EST. Requires service number as argument.

ADF.USIM/EF.EST: est_service_disable

Disables a single service in EF.EST. Requires service number as argument.

EF.IMSI: update_imsi_plmn

Change the PLMN part (MCC+MNC) of the IMSI. Requires a single argument consisting of 5/6 digits of concatenated MCC+MNC.

ADF.USIM/EF.UST: ust_service_activate

Activates a single service in EF.UST. Requires service number as argument.

ADF.USIM/EF.UST: ust_service_deactivate

Deactivates a single service in EF.UST. Requires service number as argument.

ADF.USIM/EF.UST: ust_service_check

Check consistency between services of this file and files present/activated. Many services determine if one or multiple files shall be present/activated or if they shall be absent/deactivated. This performs a consistency check to ensure that no services are activated for files that are not - and vice-versa, no files are activated for services that are not. Error messages are printed for every inconsistency found.

ADF.ISIM/EF.IST: ist_service_activate

Activates a single service in EF.IST. Requires service number as argument.

ADF.ISIM/EF.IST: ist_service_deactivate

Deactivates a single service in EF.UST. Requires service number as argument.

ADF.ISIM/EF.IST: ist_service_check

Check consistency between services of this file and files present/activated. Many services determine if one or multiple files shall be present/activated or if they shall be absent/deactivated. This performs a consistency check to ensure that no services are activated for files that are not - and vice-versa, no files are activated for services that are not. Error messages are printed for every inconsistency found.

1.1.14 UICC Administrative commands

ETSI TS 102 222 specifies a set of *Administrative Commands*, which can be used by the card issuer / operator to modify the file system structure (delete files, create files) or even to terminate individual files or the entire card.

pySim-shell supports those commands, but **use extreme caution**. Unless you know exactly what you're doing, it's very easy to render your card unusable. You've been warned!

delete_file

```
usage: build.py [-h] [--force-delete] NAME
```

Positional Arguments

NAME File name or FID to delete

Named Arguments

--force-delete I really want to permanently delete the file. I know pySim cannot re-create it yet!

Default: False

terminate_df

```
usage: build.py [-h] [--force] NAME
```

Positional Arguments

NAME File name or FID

Named Arguments

--force I really want to terminate the file. I know I can not recover from it!

Default: False

terminate_ef

```
usage: build.py [-h] [--force] NAME
```

Positional Arguments

NAME File name or FID

Named Arguments

--force I really want to terminate the file. I know I can not recover from it!

Default: False

terminate_card

```
usage: build.py [-h] [--force-terminate-card]
```

Named Arguments

--force-terminate-card I really want to permanently terminate the card. It will not be usable afterwards!

Default: False

create_ef

```
usage: build.py [-h] --ef-arr-file-id EF_ARR_FILE_ID
                --ef-arr-record-nr EF_ARR_RECORD_NR --file-size FILE_SIZE
                --structure {transparent,linear_fixed,ber_tlv}
                [--short-file-id SHORT_FILE_ID] [--shareable]
                [--record-length RECORD_LENGTH]
                FILE_ID
```

Positional Arguments

FILE_ID File Identifier as 4-character hex string

required arguments

--ef-arr-file-id Referenced Security: File Identifier of EF.ARR

--ef-arr-record-nr Referenced Security: Record Number within EF.ARR

--file-size Size of file in octets

--structure Possible choices: transparent, linear_fixed, ber_tlv

Structure of the to-be-created EF

Named Arguments

--short-file-id Short File Identifier as 2-digit hex string

--shareable Should the file be shareable?

Default: False

--record-length Length of each record in octets

create_df

```
usage: build.py [-h] --ef-arr-file-id EF_ARR_FILE_ID
                --ef-arr-record-nr EF_ARR_RECORD_NR [--shareable] [--aid AID]
                [--total-file-size TOTAL_FILE_SIZE] [--permit-rfm-create]
                [--permit-rfm-delete-terminate] [--permit-other-applet-create]
                [--permit-other-applet-delete-terminate]
                FILE_ID
```

Positional Arguments

FILE_ID File Identifier as 4-character hex string

required arguments

--ef-arr-file-id Referenced Security: File Identifier of EF.ARR

--ef-arr-record-nr Referenced Security: Record Number within EF.ARR

Named Arguments

--shareable Should the file be shareable?

Default: False

--aid Application ID (creates an ADF, instead of a DF)

--total-file-size Physical memory allocated for DF/ADi in octets

sysmoISIM-SJA optional arguments

--permit-rfm-create Default: False

--permit-rfm-delete-terminate Default: False

--permit-other-applet-create Default: False

--permit-other-applet-delete-terminate Default: False

resize_ef

```
usage: build.py [-h] --file-size FILE_SIZE NAME
```

Positional Arguments

NAME Name or FID of file to be resized

required arguments

--file-size Size of file in octets

1.1.15 ARA-M commands

The ARA-M commands exist to manage the access rules stored in an ARA-M applet on the card.

ARA-M in the context of SIM cards is primarily used to enable Android UICC Carrier Privileges, please see https://source.android.com/devices/tech/config/uicc for more details on the background.

aram_get_all

Obtain and decode all access rules from the ARA-M applet on the card.

NOTE: if the total size of the access rules exceeds 255 bytes, this command will fail, as it doesn't yet implement fragmentation/reassembly on rule retrieval. YMMV

```
pySIM-shell (00:MF/ADF.ARA-M)> aram_get_all

[
  {
    "response_all_ref_ar_do": [
      {
        "ref_ar_do": [
          {
            "ref_do": [
              {
                "aid_ref_do": "ffffffffff"
              },
              {
                "dev_app_id_ref_do": "e46872f28b350b7e1f140de535c2a8d5804f0be3"
              }
            ]
          },
          {
            "ar_do": [
              {
                "apdu_ar_do": {
                  "generic_access_rule": "always"
                }
              },
              {
                "perm_ar_do": {
                  "permissions": "0000000000000001"
                }
              }
            ]
          }
        ]
      }
```

(continues on next page)

(continued from previous page)

```
      ]
   }
]
```

aram_get_config

Perform Config handshake with ARA-M applet: Tell it our version and retrieve its version.

NOTE: Not supported in all ARA-M implementations.

aram_store_ref_ar_do

```
usage: build.py [-h] --device-app-id DEVICE_APP_ID [--aid AID | --aid-empty]
                [--pkg-ref PKG_REF] [--apdu-never | --apdu-always |
                --apdu-filter APDU_FILTER] [--nfc-never | --nfc-always]
                [--android-permissions ANDROID_PERMISSIONS]
```

Named Arguments

--device-app-id Identifies the specific device application that the rule applies to. Hash of Certificate of Application Provider, or UUID. (20/32 hex bytes)

--aid Identifies the specific SE application for which rules are to be stored. Can be a partial AID, containing for example only the RID. (5-16 or 0 hex bytes)

--aid-empty No specific SE application, applies to implicitly selected application (all channels)

Default: False

--pkg-ref Full Android Java package name (up to 127 chars ASCII)

--apdu-never APDU access is not allowed

Default: False

--apdu-always APDU access is allowed

Default: False

--apdu-filter APDU filter: multiple groups of 8 hex bytes (4 byte CLA/INS/P1/P2 followed by 4 byte mask)

--nfc-never NFC event access is not allowed

Default: False

--nfc-always NFC event access is allowed

Default: False

--android-permissions Android UICC Carrier Privilege Permissions (8 hex bytes)

For example, to store an Android UICC carrier privilege rule for the SHA1 hash of the certificate used to sign the CoIMS android app of Supreeth Herle (https://github.com/herlesupreeth/CoIMS_Wiki) you can use the following command:

```
pySIM-shell (00:MF/ADF.ARA-M)> aram_store_ref_ar_do --aid FFFFFFFFFFFF --device-app-id␣
↪E46872F28B350B7E1F140DE535C2A8D5804F0BE3 --android-permissions 0000000000000001 --apdu-
↪always
```

aram_delete_all

This command will request deletion of all access rules stored within the ARA-M applet. Use it with caution, there is no undo. Any rules later intended must be manually inserted again using *aram_store_ref_ar_do*

aram_lock

This command allows to lock the access to the STORE DATA command. This renders all access rules stored within the ARA-M applet effectively read-only. The lock can only be removed via a secure channel to the security domain and is therefore suitable to prevent unauthorized changes to ARA-M rules.

Removal of the lock:

```
pySIM-shell (SCP02[01]:00:MF/ADF.ISD)> install_for_personalization A00000015141434C00
pySIM-shell (SCP02[01]:00:MF/ADF.ISD)> apdu --expect-sw 9000 80E2900001A2
```

NOTE: ARA-M Locking is a proprietary feature that is specific to sysmocom's fork of Bertrand Martel's ARA-M implementation. ARA-M Locking is supported in newer (2025) applet versions from v0.1.0 onward.

1.1.16 GlobalPlatform commands

pySim-shell has only the mots rudimentary support for GlobalPlatform at this point. Please use dedicated projects like GlobalPlatformPro meanwhile.

get_data

```
usage: build.py [-h] data_object_name
```

Positional Arguments

data_object_name Name of the data object to be retrieved from the card

get_status

```
usage: build.py [-h] [--aid AID] {isd,applications,files,files_and_modules}
```

Positional Arguments

subset Possible choices: isd, applications, files, files_and_modules

Subset of statuses to be included in the response

Named Arguments

--aid AID Search Qualifier (search only for given AID)

Default: ""

set_status

```
usage: build.py [-h] [--aid AID]
                {isd,app_or_ssd,isd_and_assoc_apps}
                {loaded,installed,selectable,personalized,locked}
```

Positional Arguments

scope Possible choices: isd, app_or_ssd, isd_and_assoc_apps

Defines the scope of the requested status change

status Possible choices: loaded, installed, selectable, personalized, locked

Specify the new intended status

Named Arguments

--aid AID of the target Application or Security Domain

store_data

```
usage: build.py [-h] [--data-structure {none,dgi,ber_tlv,rfu}]
                [--encryption {none,application_dependent,rfu,encrypted}]
                [--response {not_expected,may_be_returned}]
                DATA
```

Positional Arguments

DATA

Named Arguments

--data-structure Possible choices: none, dgi, ber_tlv, rfu

Default: “none”

--encryption Possible choices: none, application_dependent, rfu, encrypted

Default: “none”

--response Possible choices: not_expected, may_be_returned

Default: “not_expected”

put_key

```
usage: build.py [-h] [--old-key-version-nr OLD_KEY_VERSION_NR]
                --key-version-nr KEY_VERSION_NR --key-id KEY_ID
                --key-type {des,tls_psk,aes,hmac_sha1,hmac_sha1_160,rsa_public_exponent_e_cleartex,
↪rsa_modulus_n_cleartext,rsa_modulus_n,rsa_private_exponent_d,rsa_chines_remainder_p,rsa_
↪chines_remainder_q,rsa_chines_remainder_pq,rsa_chines_remainder_dpi,rsa_chines_remainder_
↪dqi,ecc_public_key,ecc_private_key,ecc_field_parameter_p,ecc_field_parameter_a,ecc_field_
↪parameter_b,ecc_field_parameter_g,ecc_field_parameter_n,ecc_field_parameter_k,ecc_key_
↪parameters_reference,not_available}
                --key-data KEY_DATA [--key-check KEY_CHECK]
                [--suppress-key-check]
```

Named Arguments

--old-key-version-nr Old Key Version Number

Default: 0

--key-version-nr Key Version Number

--key-id Key Identifier (base)

--key-type Possible choices: des, tls_psk, aes, hmac_sha1, hmac_sha1_160, rsa_public_exponent_e_cleartex, rsa_modulus_n_cleartext, rsa_modulus_n, rsa_private_exponent_d, rsa_chines_remainder_p, rsa_chines_remainder_q, rsa_chines_remainder_pq, rsa_chines_remainder_dpi, rsa_chines_remainder_dqi, ecc_public_key, ecc_private_key, ecc_field_parameter_p, ecc_field_parameter_a, ecc_field_parameter_b, ecc_field_parameter_g, ecc_field_parameter_n, ecc_field_parameter_k, ecc_key_parameters_reference, not_available

Key Type

--key-data Key Data Block

--key-check Key Check Value

--suppress-key-check Suppress generation of Key Check Values

Default: False

delete_key

```
usage: build.py [-h] [--key-id KEY_ID] [--key-ver KEY_VER]
                [--delete-related-objects]
```

Named Arguments

--key-id Key Identifier (KID)

--key-ver Key Version Number (KVN)

--delete-related-objects Delete not only the object but also its related objects

Default: False

load

```
usage: build.py [-h] (--from-hex FROM_HEX | --from-file FROM_FILE |
                --from-cap-file FROM_CAP_FILE) [--chunk-len CHUNK_LEN]
```

Named Arguments

--from-hex load from hex string

--from-file load from binary file

--from-cap-file load from JAVA-card CAP file

--chunk-len Block size for the LOAD command; default: as large as the current secure channel overhead permits, at most 240

install_cap

```
usage: build.py FILE [--install-parameters | --install-parameters-*]
```

Positional Arguments

FILE JAVA-CARD CAP file to install

Named Arguments

--chunk-len Block size for the LOAD command; default: as large as the current secure channel overhead permits, at most 240

Install Parameters

--install-parameters install Parameters (GPC_SPE_034, section 11.5.2.3.7, table 11-49)

--install-parameters-volatile-memory-quota volatile memory quota (GPC_SPE_034, section 11.5.2.3.7, table 11-49)

--install-parameters-non-volatile-memory-quota non volatile memory quota (GPC_SPE_034, section 11.5.2.3.7, table 11-49)

--install-parameters-stk Load Parameters (ETSI TS 102 226, section 8.2.1.3.2.1)

install_for_personalization

```
usage: build.py [-h] application_aid
```

Positional Arguments

application_aid Application AID

install_for_install

```
usage: build.py [-h] [--load-file-aid LOAD_FILE_AID] [--module-aid MODULE_AID]
                --application-aid APPLICATION_AID
                [--install-parameters INSTALL_PARAMETERS]
                [--privilege {security_domain,dap_verification,delegated_management,card_lock,card_
↪terminate,card_reset,cvm_management,mandated_dap_verification,trusted_path,authorized_
↪management,token_management,global_delete,global_lock,global_registry,final_application,global_
↪service,receipt_generation,ciphered_load_file_data_block,contactless_activation,contactless_self_
↪activation}]
                [--install-token INSTALL_TOKEN] [--make-selectable]
```

Named Arguments

--load-file-aid Executable Load File AID

Default: “”

--module-aid Executable Module AID

Default: “”

--application-aid Application AID

--install-parameters Install Parameters (GPC_SPE_034, section 11.5.2.3.7, table 11-49)

Default: “”

--privilege Possible choices: security_domain, dap_verification, delegated_management, card_lock, card_terminate, card_reset, cvm_management, mandated_dap_verification, trusted_path, authorized_management, token_management, global_delete, global_lock, global_registry, final_application, global_service, receipt_generation, ciphered_load_file_data_block, contactless_activation, contactless_self_activation

Privilege granted to newly installed Application

Default: []

--install-token Install Token (Section GPCS C.4.2/C.4.7)

Default: “”

--make-selectable Install and make selectable

Default: False

install_for_load

```
usage: build.py [-h] --load-file-aid LOAD_FILE_AID
                [--security-domain-aid SECURITY_DOMAIN_AID]
                [--load-file-hash LOAD_FILE_HASH]
                [--load-parameters LOAD_PARAMETERS] [--load-token LOAD_TOKEN]
```

Named Arguments

--load-file-aid AID of the loaded file

--security-domain-aid AID of the Security Domain into which the file shalle be added

Default: “”

--load-file-hash Load File Data Block Hash (GPC_SPE_034, section C.2)

Default: “”

--load-parameters Load Parameters (GPC_SPE_034, section 11.5.2.3.7, table 11-49)

Default: “”

--load-token Load Token (GPC_SPE_034, section C.4.1)

Default: “”

delete_card_content

```
usage: build.py [-h] [--delete-related-objects] aid
```

Positional Arguments

aid Executable Load File or Application AID

Named Arguments

--delete-related-objects Delete not only the object but also its related objects

Default: False

establish_scp02

```
usage: build.py [-h] [--key-ver KEY_VER] [--host-challenge HOST_CHALLENGE]
                [--security-level SECURITY_LEVEL] [--key-enc KEY_ENC]
                [--key-mac KEY_MAC] [--key-dek KEY_DEK]
                [--key-provider-suffix KEY_PROVIDER_SUFFIX]
```

Named Arguments

--key-ver Key Version Number (KVN)

Default: 0

--host-challenge Hard-code the host challenge; default: random

--security-level Security Level. Default: 0x01 (C-MAC only)

Default: 1

Manual key specification

--key-enc Secure Channel Encryption Key

--key-mac Secure Channel MAC Key

--key-dek Data Encryption Key

Obtain keys from CardKeyProvider (e.g. CSV

--key-provider-suffix Suffix for key names in CardKeyProvider

establish_scp03

```
usage: build.py [-h] [--key-ver KEY_VER] [--host-challenge HOST_CHALLENGE]
                [--security-level SECURITY_LEVEL] [--key-enc KEY_ENC]
                [--key-mac KEY_MAC] [--key-dek KEY_DEK]
                [--key-provider-suffix KEY_PROVIDER_SUFFIX] [--s16-mode]
```

Named Arguments

--key-ver Key Version Number (KVN)

Default: 0

--host-challenge Hard-code the host challenge; default: random

--security-level Security Level. Default: 0x01 (C-MAC only)

Default: 1

--s16-mode S16 mode (S8 is default)

Default: False

Manual key specification

--key-enc Secure Channel Encryption Key

--key-mac Secure Channel MAC Key

--key-dek Data Encryption Key

Obtain keys from CardKeyProvider (e.g. CSV

--key-provider-suffix Suffix for key names in CardKeyProvider

release_scp

Release any previously established SCP (Secure Channel Protocol)

1.1.17 eUICC ISD-R commands

These commands are to perform a variety of operations against eUICC for GSMA consumer eSIM. They implement the so-called ES10a, ES10b and ES10c interface. Basically they perform the tasks that usually would be done by the LPAd in the UE.

In order to use those commands, you need to go through the specified steps as documented in GSMA SGP.22:

- open a new logical channel (and start to use it)
- select the ISD-R application

Example:

```
pySIM-shell (00:MF)> open_channel 2
pySIM-shell (00:MF)> switch_channel 2
pySIM-shell (02:MF)> select ADF.ISD-R
{
   "application_id": "a0000005591010ffffffff8900000100",
   "proprietary_data": {
      "maximum_length_of_data_field_in_command_message": 255
   },
   "isdr_proprietary_application_template": {
      "supported_version_number": "020200"
   }
}
pySIM-shell (02:ADF.ISD-R)>
```

Once you are at this stage, you can issue the various eUICC related commands against the ISD-R application

es10x_store_data

```
usage: build.py [-h] TX_DO
```

Positional Arguments

TX_DO Hexstring of encoded to-be-transmitted DO

get_euicc_configured_addresses

Obtain the configured SM-DP+ and/or SM-DS addresses using the ES10a GetEuiccConfiguredAddresses() function.

Example:

```
pySIM-shell (00:MF/ADF.ISD-R)> get_euicc_configured_addresses
{
```

(continues on next page)

(continued from previous page)

```
    "root_ds_address": "testrootsmds.gsma.com"
}
```

set_default_dp_address

```
usage: build.py [-h] DP_ADDRESS
```

Positional Arguments

DP_ADDRESS Default SM-DP+ address as UTF-8 string

get_euicc_challenge

Obtain an authentication challenge from the eUICC using the ES10b GetEUICCChallenge() function.

Example:

```
pySIM-shell (00:MF/ADF.ISD-R)> get_euicc_challenge
{
    "euicc_challenge": "3668f20d4e6c8e85609bbca8c14873fd"
}
```

get_euicc_info1

Obtain EUICC Information (1) from the eUICC using the ES10b GetEUICCCInfo() function.

Example:

```
pySIM-shell (00:MF/ADF.ISD-R)> get_euicc_info1
{
    "svn": "2.2.0",
    "euicc_ci_pki_list_for_verification": [
        {
            "subject_key_identifier_seq": {
                "unknown_ber_tlv_ie_c0": null
            }
        },
        {
            "subject_key_identifier_seq": {
                "unknown_ber_tlv_ie_f5": {
                    "raw": "72bdf98a95d65cbeb88a38a1c11d800a85c3"
                }
            }
        }
    ],
    "euicc_ci_pki_list_for_signing": [
        {
            "subject_key_identifier_seq": {
                "unknown_ber_tlv_ie_c0": null
            }
        },
        {
            "subject_key_identifier_seq": {
```

(continues on next page)

(continued from previous page)

```
            "unknown_ber_tlv_ie_f5": {
               "raw": "72bdf98a95d65cbeb88a38a1c11d800a85c3"
            }
         }
      }
   ]
}
```

get_euicc_info2

Obtain EUICC Information (2) from the eUICC using the ES10b GetEUICCCInfo() function.

Example:

```
pySIM-shell (00:MF/ADF.ISD-R)> get_euicc_info2
{
   "profile_version": "2.1.0",
   "svn": "2.2.0",
   "euicc_firmware_ver": "4.4.0",
   "ext_card_resource": "81010082040006ddc68304000016e0",
   "uicc_capability": "067f36c0",
   "ts102241_version": "9.2.0",
   "global_platform_version": "2.3.0",
   "rsp_capability": "0490",
   "euicc_ci_pki_list_for_verification": [
      {
         "subject_key_identifier_seq": {
            "unknown_ber_tlv_ie_c0": null
         }
      },
      {
         "subject_key_identifier_seq": {
            "unknown_ber_tlv_ie_f5": {
               "raw": "72bdf98a95d65cbeb88a38a1c11d800a85c3"
            }
         }
      }
   ],
   "euicc_ci_pki_list_for_signing": [
      {
         "subject_key_identifier_seq": {
            "unknown_ber_tlv_ie_c0": null
         }
      },
      {
         "subject_key_identifier_seq": {
            "unknown_ber_tlv_ie_f5": {
               "raw": "72bdf98a95d65cbeb88a38a1c11d800a85c3"
            }
         }
      }
   ],
```

(continues on next page)

(continued from previous page)

```
    "unknown_ber_tlv_ie_99": {
        "raw": "06c0"
    },
    "pp_version": "0.0.1",
    "ss_acreditation_number": "G&DAccreditationNbr",
    "unknown_ber_tlv_ie_ac": {
        "raw":
↪"801f312e322e3834302e313233343536372f6d79506c6174666f726d4c6162656c812568747470733a2f2f6d79636f6d70616e792e63
↪"
    }
}
```

list_notification

Obtain the list of notifications from the eUICC using the ES10b ListNotification() function.

Example:

```
pySIM-shell (00:MF/ADF.ISD-R)> list_notification
{
    "notification_metadata_list": {
        "notification_metadata": {
            "seq_number": 61,
            "profile_mgmt_operation": {
                "pmo": {
                    "install": true,
                    "enable": false,
                    "disable": false,
                    "delete": false
                }
            },
            "notification_address": "testsmdpplus1.example.com",
            "iccid": "89000123456789012358"
        }
    }
}
```

remove_notification_from_list

```
usage: build.py [-h] SEQ_NR
```

Positional Arguments

SEQ_NR Sequence Number of the to-be-removed notification

Example:

```
pySIM-shell (00:MF/ADF.ISD-R)> remove_notification_from_list 60
{
    "delete_notification_status": "ok"
}
```

get_profiles_info

Obtain information about the profiles present on the eUICC using the ES10c GetProfilesInfo() function.

Example:

```
pySIM-shell (00:MF/ADF.ISD-R)> get_profiles_info
{
   "profile_info_seq": [
      {
         "profile_info": {
            "iccid": "89000123456789012341",
            "isdp_aid": "a0000005591010ffffffff8900001100",
            "profile_state": "disabled",
            "service_provider_name": "GSMA Test 1A",
            "profile_name": "GSMA Generic eUICC Test Profile 1A",
            "profile_class": "operational"
         }
      },
      {
         "profile_info": {
            "iccid": "89000123456789012358",
            "isdp_aid": "a0000005591010ffffffff8900001200",
            "profile_state": "disabled",
            "service_provider_name": "OsmocomSPN",
            "profile_name": "OsmocomProfile",
            "profile_class": "operational"
         }
      }
   ]
}
```

enable_profile

```
usage: build.py [-h] [--isdp-aid ISDP_AID | --iccid ICCID]
                [--refresh-required]
```

Named Arguments

--isdp-aid	Profile identified by its ISD-P AID
--iccid	Profile identified by its ICCID
--refresh-required	whether a REFRESH is required Default: False

Example (successful):

```
pySIM-shell (00:MF/ADF.ISD-R)> enable_profile --iccid 89000123456789012358
{
   "enable_result": "ok"
}
```

Example (failed attempt enabling a profile that's already enabled):

```
pySIM-shell (00:MF/ADF.ISD-R)> enable_profile --iccid 89000123456789012358
{
   "enable_result": "profileNotInDisabledState"
}
```

disable_profile

```
usage: build.py [-h] [--isdp-aid ISDP_AID | --iccid ICCID]
                [--refresh-required]
```

Named Arguments

--isdp-aid	Profile identified by its ISD-P AID
--iccid	Profile identified by its ICCID
--refresh-required	whether a REFRESH is required Default: False

Example (successful):

```
pySIM-shell (00:MF/ADF.ISD-R)> disable_profile --iccid 89000123456789012358
{
   "disable_result": "ok"
}
```

delete_profile

```
usage: build.py [-h] [--isdp-aid ISDP_AID | --iccid ICCID]
```

Named Arguments

--isdp-aid	Profile identified by its ISD-P AID
--iccid	Profile identified by its ICCID

Example:

```
pySIM-shell (00:MF/ADF.ISD-R)> delete_profile --iccid 89000123456789012358
{
   "delete_result": "ok"
}
```

euicc_memory_reset

```
usage: build.py [-h] [--delete-operational] [--delete-test-field-installed]
                [--reset-smdp-address]
```

Named Arguments

--delete-operational	Delete all operational profiles Default: False

--delete-test-field-installed Delete all test profiles, except pre-installed ones

Default: False

--reset-smdp-address Reset the SM-DP+ address

Default: False

get_eid

Obtain the EID of the eUICC using the ES10c GetEID() function.

Example:

```
pySIM-shell (00:MF/ADF.ISD-R)> get_eid
{
   "eid_value": "89049032123451234512345678901235"
}
```

set_nickname

```
usage: build.py [-h] [--profile-nickname PROFILE_NICKNAME] ICCID
```

Positional Arguments

ICCID ICCID of the profile whose nickname to set

Named Arguments

--profile-nickname Nickname of the profile

Example:

```
pySIM-shell (00:MF/ADF.ISD-R)> set_nickname --profile-nickname asdf 89000123456789012358
{
   "set_nickname_result": "ok"
}
```

get_certs

Obtain the certificates from an IoT eUICC using the ES10c GetCerts() function.

get_eim_configuration_data

Obtain the eIM configuration data from an IoT eUICC using the ES10b GetEimConfigurationData() function.

1.1.18 cmd2 settable parameters

cmd2 has the concept of *settable parameters* which act a bit like environment variables in an OS-level shell: They can be read and set, and they will influence the behavior somehow.

conserve_write

If enabled, pySim will (when asked to write to a card) always first read the respective file/record and verify if the to-be-written value differs from the current on-card value. If not, the write will be skipped. Writes will only be performed if the new value is different from the current on-card value.

If disabled, pySim will always write irrespective of the current/new value.

json_pretty_print

This parameter determines if generated JSON output should (by default) be pretty-printed (multi-line output with indent level of 4 spaces) or not.

The default value of this parameter is 'true'.

debug

If enabled, full python back-traces will be displayed in case of exceptions

apdu_trace

Boolean variable that determines if a hex-dump of the command + response APDU shall be printed.

numeric_path

Boolean variable that determines if path (e.g. in prompt) is displayed with numeric FIDs or string names.

```
pySIM-shell (00:MF/EF.ICCID)> set numeric_path True
numeric_path - was: False
now: True
pySIM-shell (00:3f00/2fe2)> set numeric_path False
numeric_path - was: True
now: False
pySIM-shell (00:MF/EF.ICCID)> help set
```

1.2 Card Filesystem Reference

This page documents all Elementary Files (EFs) and Dedicated Files (DFs) implemented in pySim, organised by their location in the card filesystem.

1.2.1 MF / TS 102 221 (UICC)

EF.DIR (2F00) — Application Directory

Examples

```
// file: 61294f10a0000000871002ffffffff8907090000050055553696d31730ea00c80011781025f608203454150
{
  "application_template": [
    {
      "application_id": "a0000000871002ffffffff8907090000"
    },
    {
      "application_label": "USim1"
    },
    {
      "discretionary_template": "a00c80011781025f608203454150"
    }
  ]
}
```

```
// file: 61194f10a0000000871004ffffffff8907090000050054953696d31
{
  "application_template": [
    {
      "application_id": "a0000000871004ffffffff8907090000"
    },
    {
      "application_label": "ISim1"
    }
  ]
}
```

EF.ICCID (2FE2) — ICC Identification

Examples

```
// file: 988812010000400310f0
{
  "iccid": "89882110000000430010"
}
```

EF.PL (2F05) — Preferred Languages

Examples

```
// file: 6465
[
  "de"
]
```

```
// file: 656e
[
  "en"
]
```

```
// file: ffff
[
  null
]
```

```
// file: 656e64657275ffffffff
[
  "en",
  "de",
  "ru",
  null,
  null
]
```

EF.ARR (2F06) — Access Rule Reference

Examples

```
// file: 800101a40683010a95010880010690008001609700 8401d4a40683010a950108
[
  [
    {
      "access_mode": [
        "read_search_compare"
      ]
    },
    {
      "control_reference_template": "ADM1"
    }
  ],
  [
    {
      "access_mode": [
        "write_append",
        "update_erase"
      ]
    },
    {
      "always": null
    }
  ],
  [
    {
      "access_mode": [
        "delete_file",
        "terminate_ef"
      ]
    },
    {
      "never": null
    }
  ],
  [
    {
      "command_header": {
        "INS": 212
      }
    },
    {
      "control_reference_template": "ADM1"
    }
  ]
]
```

```
// file: 8001019000800102970080011 8a40683010a9501088401d4a40683010a950108
[
  [
    {
```

(continues on next page)

(continued from previous page)

```
      "access_mode": [
        "read_search_compare"
      ]
    },
    {
      "always": null
    }
  ],
  [
    {
      "access_mode": [
        "update_erase"
      ]
    },
    {
      "never": null
    }
  ],
  [
    {
      "access_mode": [
        "activate_file_or_record",
        "deactivate_file_or_record"
      ]
    },
    {
      "control_reference_template": "ADM1"
    }
  ],
  [
    {
      "command_header": {
        "INS": 212
      }
    },
    {
      "control_reference_template": "ADM1"
    }
  ]
]
```

EF.UMPC (2F08) — UICC Maximum Power Consumption

Examples

```
// file: 3cff02
{
  "max_current_mA": 60,
  "t_op_s": 255,
  "addl_info": {
    "req_inc_idle_current": false,
    "support_uicc_suspend": true
```

(continues on next page)

(continued from previous page)

```
  }
}
```

```
// file: 320500
{
  "max_current_mA": 50,
  "t_op_s": 5,
  "addl_info": {
    "req_inc_idle_current": false,
    "support_uicc_suspend": false
  }
}
```

1.2.2 ADF.USIM / TS 31.102

ADF.USIM/EF.LI (6F05) — Language Indication

ADF.USIM/EF.Keys (6F08) — Ciphering and Integrity Keys

ADF.USIM/EF.KeysPS (6F09) — Ciphering and Integrity Keys for PS domain

ADF.USIM/EF.PLMNwAcT (6F60) — User controlled PLMN Selector with Access Technology

Examples

```
// file: 62f2104000
[
  {
    "mcc": "262",
    "mnc": "01",
    "act": [
      "E-UTRAN NB-S1",
      "E-UTRAN WB-S1"
    ]
  }
]
```

```
// file: 62f2108000
[
  {
    "mcc": "262",
    "mnc": "01",
    "act": [
      "UTRAN"
    ]
  }
]
```

```
// file: 62f220488c
[
  {
    "mcc": "262",
    "mnc": "02",
```

(continues on next page)

(continued from previous page)

```
    "act": [
      "E-UTRAN NB-S1",
      "E-UTRAN WB-S1",
      "EC-GSM-IoT",
      "GSM",
      "NG-RAN"
    ]
  }
]
```

ADF.USIM/EF.HPPLMN (6F31) — Higher Priority PLMN search period

Examples

```
// file: 05
5
```

ADF.USIM/EF.ACMmax (6F37) — ACM maximum value

Examples

```
// file: 000000
{
  "acm_max": 0
}
```

ADF.USIM/EF.UST (6F38) — USIM Service Table

ADF.USIM/EF.ACM (6F39) — Accumulated call meter

ADF.USIM/EF.GID1 (6F3E) — Group Identifier Level 1

ADF.USIM/EF.GID2 (6F3F) — Group Identifier Level 2

ADF.USIM/EF.SPN (6F46) — Service Provider Name

Examples

```
// file: 0147534d2d52204348ffffffffffffff
{
  "rfu": 0,
  "hide_in_oplmn": false,
  "show_in_hplmn": true,
  "spn": "GSM-R CH"
}
```

ADF.USIM/EF.PUCT (6F41) — Price per unit and currency table

ADF.USIM/EF.CBMI (6F45) — Cell Broadcast message identifier selection

ADF.USIM/EF.ACC (6F78) — Access Control Class

Examples

```
// file: 0000
{
  "ACC0": false,
  "ACC1": false,
  "ACC2": false,
  "ACC3": false,
  "ACC4": false,
  "ACC5": false,
  "ACC6": false,
  "ACC7": false,
  "ACC8": false,
  "ACC9": false,
  "ACC10": false,
  "ACC11": false,
  "ACC12": false,
  "ACC13": false,
  "ACC14": false,
  "ACC15": false
}
```

```
// file: 0001
{
  "ACC0": true,
  "ACC1": false,
  "ACC2": false,
  "ACC3": false,
  "ACC4": false,
  "ACC5": false,
  "ACC6": false,
  "ACC7": false,
  "ACC8": false,
  "ACC9": false,
  "ACC10": false,
  "ACC11": false,
  "ACC12": false,
  "ACC13": false,
  "ACC14": false,
  "ACC15": false
}
```

```
// file: 0002
{
  "ACC0": false,
  "ACC1": true,
  "ACC2": false,
  "ACC3": false,
  "ACC4": false,
  "ACC5": false,
  "ACC6": false,
  "ACC7": false,
  "ACC8": false,
  "ACC9": false,
```

(continues on next page)

(continued from previous page)

```
  "ACC10": false,
  "ACC11": false,
  "ACC12": false,
  "ACC13": false,
  "ACC14": false,
  "ACC15": false
}
```

```
// file: 0100
{
  "ACC0": false,
  "ACC1": false,
  "ACC2": false,
  "ACC3": false,
  "ACC4": false,
  "ACC5": false,
  "ACC6": false,
  "ACC7": false,
  "ACC8": true,
  "ACC9": false,
  "ACC10": false,
  "ACC11": false,
  "ACC12": false,
  "ACC13": false,
  "ACC14": false,
  "ACC15": false
}
```

```
// file: 8000
{
  "ACC0": false,
  "ACC1": false,
  "ACC2": false,
  "ACC3": false,
  "ACC4": false,
  "ACC5": false,
  "ACC6": false,
  "ACC7": false,
  "ACC8": false,
  "ACC9": false,
  "ACC10": false,
  "ACC11": false,
  "ACC12": false,
  "ACC13": false,
  "ACC14": false,
  "ACC15": true
}
```

```
// file: 802b
{
  "ACC0": true,
```

(continues on next page)

(continued from previous page)


```
  "ACC1": true,
  "ACC2": false,
  "ACC3": true,
  "ACC4": false,
  "ACC5": true,
  "ACC6": false,
  "ACC7": false,
  "ACC8": false,
  "ACC9": false,
  "ACC10": false,
  "ACC11": false,
  "ACC12": false,
  "ACC13": false,
  "ACC14": false,
  "ACC15": true
}
```


ADF.USIM/EF.FPLMN (6F7B) — Forbidden PLMNs

Examples


```
// file: 22f860
[
  {
    "mcc": "228",
    "mnc": "06"
  }
]
```



```
// file: 330420
[
  {
    "mcc": "334",
    "mnc": "020"
  }
]
```


ADF.USIM/EF.LOCI (6F7E) — Location information

Examples


```
// file: 47d1264a62f21037211e00
{
  "tmsi": "47d1264a",
  "lai": {
    "mcc_mnc": "262-01",
    "lac": "3721"
  },
  "rfu": 30,
  "lu_status": 0
}
```

```
// file: ffffffff62f2200000ff01
{
  "tmsi": "ffffffff",
  "lai": {
    "mcc_mnc": "262-02",
    "lac": "0000"
  },
  "rfu": 255,
  "lu_status": 1
}
```

ADF.USIM/EF.AD (6FAD) — Administrative Data

Examples

```
// file: 00000002
{
  "ms_operation_mode": "normal",
  "additional_info": {
    "ciphering_indicator": false,
    "csg_display_control": false,
    "prose_services": false,
    "extended_drx": false
  },
  "rfu": 0,
  "mnc_len": 2,
  "extensions": ""
}
```

```
// file: 01000102
{
  "ms_operation_mode": "normal_and_specific_facilities",
  "additional_info": {
    "ciphering_indicator": true,
    "csg_display_control": false,
    "prose_services": false,
    "extended_drx": false
  },
  "rfu": 0,
  "mnc_len": 2,
  "extensions": ""
}
```

ADF.USIM/EF.CBMID (6F48) — Cell Broadcast Message Identifier for Data Download

ADF.USIM/EF.ECC (6FB7) — Emergency Call Codes

Examples

```
// file: 19f1ff01
{
  "call_code": "911",
  "service_category": {
```

(continues on next page)

(continued from previous page)


```
    "police": true,
    "ambulance": false,
    "fire_brigade": false,
    "marine_guard": false,
    "mountain_rescue": false,
    "manual_ecall": false,
    "automatic_ecall": false
  }
}
```

```
// file: 19f3ff02
{
  "call_code": "913",
  "service_category": {
    "police": false,
    "ambulance": true,
    "fire_brigade": false,
    "marine_guard": false,
    "mountain_rescue": false,
    "manual_ecall": false,
    "automatic_ecall": false
  }
}
```


ADF.USIM/EF.CBMIR (6F50) — Cell Broadcast message identifier range selection

ADF.USIM/EF.PSLOCI (6F73) — PS Location information

ADF.USIM/EF.FDN (6F3B) — Fixed Dialling Numbers

Examples


```
// file: 42204841203120536963ffffffffff068106285608 10ffffffffff
{
  "alpha_id": "B HA 1 Sic",
  "len_of_bcd": 6,
  "ton_npi": {
    "ext": true,
    "type_of_number": "unknown",
    "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "6082658001",
  "cap_conf_id": 255,
  "ext1_record_id": 255
}
```

```
// file: 4b756e64656e626574726575756e67ff07919471121227 21ffffffffff
{
  "alpha_id": "Kundenbetreuung",
  "len_of_bcd": 7,
  "ton_npi": {
    "ext": true,
```


(continues on next page)

(continued from previous page)

```
    "type_of_number": "international",
    "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "491721217212",
  "cap_conf_id": 255,
  "ext1_record_id": 255
}
```

ADF.USIM/EF.SMS (6F3C) — Short messages

ADF.USIM/EF.MSISDN (6F40) — MSISDN

Examples

```
// file: ffffffffffffffffffffffffffffffff04b12143f5ffffffffffffff
{
  "alpha_id": "",
  "len_of_bcd": 4,
  "ton_npi": {
    "ext": true,
    "type_of_number": "network_specific",
    "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "12345"
}
```

```
// file: 456967656e65205275666e756d6d6572ffffffff0891947172199181f3ffffffff
{
  "alpha_id": "Eigene Rufnummer",
  "len_of_bcd": 8,
  "ton_npi": {
    "ext": true,
    "type_of_number": "international",
    "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "4917279119183"
}
```

ADF.USIM/EF.SMSP (6F42) — Short message service parameters

Examples

```
// file: 534d5343ffffffffffffffffffffffe1ffffffffffffffffffffff0891945197109099f9fffff0000a9
{
  "alpha_id": "SMSC",
  "parameter_indicators": {
    "tp_dest_addr": false,
    "tp_sc_addr": true,
    "tp_pid": true,
    "tp_dcs": true,
    "tp_vp": true
  },
```

(continues on next page)

(continued from previous page)


```
  "tp_dest_addr": {
    "length": 255,
    "ton_npi": {
      "ext": true,
      "type_of_number": "reserved_for_extension",
      "numbering_plan_id": "reserved_for_extension"
    },
    "call_number": ""
  },
  "tp_sc_addr": {
    "length": 8,
    "ton_npi": {
      "ext": true,
      "type_of_number": "international",
      "numbering_plan_id": "isdn_e164"
    },
    "call_number": "4915790109999"
  },
  "tp_pid": "00",
  "tp_dcs": "00",
  "tp_vp_minutes": 4320
}
```

```
// file: e1ffffffffffffffffffff0891945197109099f9fffff0000a9
{
  "alpha_id": "",
  "parameter_indicators": {
    "tp_dest_addr": false,
    "tp_sc_addr": true,
    "tp_pid": true,
    "tp_dcs": true,
    "tp_vp": true
  },
  "tp_dest_addr": {
    "length": 255,
    "ton_npi": {
      "ext": true,
      "type_of_number": "reserved_for_extension",
      "numbering_plan_id": "reserved_for_extension"
    },
    "call_number": ""
  },
  "tp_sc_addr": {
    "length": 8,
    "ton_npi": {
      "ext": true,
      "type_of_number": "international",
      "numbering_plan_id": "isdn_e164"
    },
    "call_number": "4915790109999"
  },
  "tp_pid": "00",
```


(continues on next page)

(continued from previous page)

```
  "tp_dcs": "00",
  "tp_vp_minutes": 4320
}
```

```
// file: 454e6574776f726b73ffffffffffff1fffffffffffffffffffffffffffffffffffffffffffff0000a7
{
  "alpha_id": "ENetworks",
  "parameter_indicators": {
    "tp_dest_addr": false,
    "tp_sc_addr": true,
    "tp_pid": true,
    "tp_dcs": true,
    "tp_vp": false
  },
  "tp_dest_addr": {
    "length": 255,
    "ton_npi": {
      "ext": true,
      "type_of_number": "reserved_for_extension",
      "numbering_plan_id": "reserved_for_extension"
    },
    "call_number": ""
  },
  "tp_sc_addr": {
    "length": 255,
    "ton_npi": {
      "ext": true,
      "type_of_number": "reserved_for_extension",
      "numbering_plan_id": "reserved_for_extension"
    },
    "call_number": ""
  },
  "tp_pid": "00",
  "tp_dcs": "00",
  "tp_vp_minutes": 1440
}
```

```
// file: fffffffffffffffffffffffffffffffffffffffffffdfffffffffffffffffff07919403214365f7ffffffffff
{
  "alpha_id": "",
  "parameter_indicators": {
    "tp_dest_addr": false,
    "tp_sc_addr": true,
    "tp_pid": false,
    "tp_dcs": false,
    "tp_vp": false
  },
  "tp_dest_addr": {
    "length": 255,
    "ton_npi": {
      "ext": true,
```

(continues on next page)

(continued from previous page)


```
      "type_of_number": "reserved_for_extension",
      "numbering_plan_id": "reserved_for_extension"
    },
    "call_number": ""
  },
  "tp_sc_addr": {
    "length": 7,
    "ton_npi": {
      "ext": true,
      "type_of_number": "international",
      "numbering_plan_id": "isdn_e164"
    },
    "call_number": "49301234567"
  },
  "tp_pid": "ff",
  "tp_dcs": "ff",
  "tp_vp_minutes": 635040
}
```



```
// file: ffffffffffffffffffffffffffffffffffffffffffffffffc0b919403214365f7fffffff07919403214365f7ffffffffffff
{
  "alpha_id": "",
  "parameter_indicators": {
    "tp_dest_addr": true,
    "tp_sc_addr": true,
    "tp_pid": false,
    "tp_dcs": false,
    "tp_vp": false
  },
  "tp_dest_addr": {
    "length": 11,
    "ton_npi": {
      "ext": true,
      "type_of_number": "international",
      "numbering_plan_id": "isdn_e164"
    },
    "call_number": "49301234567"
  },
  "tp_sc_addr": {
    "length": 7,
    "ton_npi": {
      "ext": true,
      "type_of_number": "international",
      "numbering_plan_id": "isdn_e164"
    },
    "call_number": "49301234567"
  },
  "tp_pid": "ff",
  "tp_dcs": "ff",
  "tp_vp_minutes": 635040
}
```

ADF.USIM/EF.SMSS (6F43) — SMS status

ADF.USIM/EF.SDN (6F49) — Service Dialling Numbers

Examples

```
// file: 42204841203120536963ffffffffff06810628560810ffffffffffff
{
  "alpha_id": "B HA 1 Sic",
  "len_of_bcd": 6,
  "ton_npi": {
    "ext": true,
    "type_of_number": "unknown",
    "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "6082658001",
  "cap_conf_id": 255,
  "ext1_record_id": 255
}
```

```
// file: 4b756e64656e626574726575756e67ff07919471121227 21ffffffffff
{
  "alpha_id": "Kundenbetreuung",
  "len_of_bcd": 7,
  "ton_npi": {
    "ext": true,
    "type_of_number": "international",
    "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "491721217212",
  "cap_conf_id": 255,
  "ext1_record_id": 255
}
```

ADF.USIM/EF.EXT2 (6F4B) — Extension2 (FDN)

ADF.USIM/EF.EXT3 (6F4C) — Extension2 (SDN)

ADF.USIM/EF.SMSR (6F47) — SMS status reports

ADF.USIM/EF.ICI (6F80) — Incoming Call Information

ADF.USIM/EF.OCI (6F81) — Outgoing Call Information

ADF.USIM/EF.ICT (6F82) — Incoming Call Timer

ADF.USIM/EF.OCT (6F83) — Outgoing Call Timer

ADF.USIM/EF.EXT5 (6F4E) — Extension5 (ICI/OCI/MSISDN)

ADF.USIM/EF.CCP2 (6F4F) — Capability Configuration Parameters 2

ADF.USIM/EF.eMLPP (6FB5) — enhanced Multi Level Pre-emption and Priority

Examples

```
// file: 7c04
{
```

(continues on next page)

(continued from previous page)

```
  "levels": {
    "A": false,
    "B": false,
    "zero": true,
    "one": true,
    "two": true,
    "three": true,
    "four": true
  },
  "fast_call_setup_cond": {
    "A": false,
    "B": false,
    "zero": true,
    "one": false,
    "two": false,
    "three": false,
    "four": false
  }
}
```

ADF.USIM/EF.AAeM (6FB6) — Automatic Answer for eMLPP Service

Examples

```
// file: 3c
{
  "auto_answer_prio_levels": {
    "A": false,
    "B": false,
    "zero": true,
    "one": true,
    "two": true,
    "three": true,
    "four": false
  }
}
```

ADF.USIM/EF.BDN (6F4D) — Barred Dialling Numbers

Examples

```
// file: 42204841203120536963ffffffffff068106285608 10ffffffffff
{
  "alpha_id": "B HA 1 Sic",
  "len_of_bcd": 6,
  "ton_npi": {
    "ext": true,
    "type_of_number": "unknown",
    "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "6082658001",
  "cap_conf_id": 255,
```

(continues on next page)

(continued from previous page)

```
  "ext1_record_id": 255
}
```

```
// file: 4b756e64656e626574726575756e67ff0791947112122721ffffffffff
{
  "alpha_id": "Kundenbetreuung",
  "len_of_bcd": 7,
  "ton_npi": {
    "ext": true,
    "type_of_number": "international",
    "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "491721217212",
  "cap_conf_id": 255,
  "ext1_record_id": 255
}
```

ADF.USIM/EF.EXT4 (6F55) — Extension4 (BDN/SSC)

ADF.USIM/EF.CMI (6F58) — Comparison Method Information

ADF.USIM/EF.EST (6F56) — Enabled Services Table

ADF.USIM/EF.ACL (6F57) — Access Point Name Control List

ADF.USIM/EF.DCK (6F2C) — Depersonalisation Control Keys

ADF.USIM/EF.CNL (6F32) — Co-operative Network List

ADF.USIM/EF.START-HFN (6F5B) — Initialisation values for Hyperframe number

Examples

```
// file: f00000f00000
{
  "start_cs": 15728640,
  "start_ps": 15728640
}
```

ADF.USIM/EF.THRESHOLD (6F5C) — Maximum value of START

Examples

```
// file: f01000
{
  "max_start": 15732736
}
```

ADF.USIM/EF.OPLMNwAcT (6F61) — User controlled PLMN Selector with Access Technology

Examples

```
// file: 62f2104000
[
```

(continues on next page)

(continued from previous page)

```
  {
    "mcc": "262",
    "mnc": "01",
    "act": [
      "E-UTRAN NB-S1",
      "E-UTRAN WB-S1"
    ]
  }
]
```

```
// file: 62f2108000
[
  {
    "mcc": "262",
    "mnc": "01",
    "act": [
      "UTRAN"
    ]
  }
]
```

```
// file: 62f220488c
[
  {
    "mcc": "262",
    "mnc": "02",
    "act": [
      "E-UTRAN NB-S1",
      "E-UTRAN WB-S1",
      "EC-GSM-IoT",
      "GSM",
      "NG-RAN"
    ]
  }
]
```

ADF.USIM/EF.HPLMNwAcT (6F62) — HPLMN Selector with Access Technology

Examples

```
// file: 62f2104000
[
  {
    "mcc": "262",
    "mnc": "01",
    "act": [
      "E-UTRAN NB-S1",
      "E-UTRAN WB-S1"
    ]
  }
]
```

```
// file: 62f2108000
[
  {
    "mcc": "262",
    "mnc": "01",
    "act": [
      "UTRAN"
    ]
  }
]
```

```
// file: 62f220488c
[
  {
    "mcc": "262",
    "mnc": "02",
    "act": [
      "E-UTRAN NB-S1",
      "E-UTRAN WB-S1",
      "EC-GSM-IoT",
      "GSM",
      "NG-RAN"
    ]
  }
]
```

ADF.USIM/EF.RPLMNAcTD (6F65) — RPLMN Last used Access Technology

ADF.USIM/EF.NETPAR (6FC4) — Network Parameters

ADF.USIM/EF.PNN (6FC5) — PLMN Network Name

ADF.USIM/EF.OPL (6FC6) — Operator PLMN List

Examples

```
// file: 62f2100000fffe01
{
  "lai": {
    "mcc_mnc": "262-01",
    "lac_min": "0000",
    "lac_max": "fffe"
  },
  "pnn_record_id": 1
}
```

```
// file: 216354789abcde12
{
  "lai": {
    "mcc_mnc": "123-456",
    "lac_min": "789a",
    "lac_max": "bcde"
  },
```

(continues on next page)

(continued from previous page)


```
  "pnn_record_id": 18
}
```


ADF.USIM/EF.MBDN (6FC7) — Mailbox Dialling Numbers

Examples


```
// file: 42204841203120536963ffffffffff068106285608 10ffffffffff
{
  "alpha_id": "B HA 1 Sic",
  "len_of_bcd": 6,
  "ton_npi": {
    "ext": true,
    "type_of_number": "unknown",
    "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "6082658001",
  "cap_conf_id": 255,
  "ext1_record_id": 255
}
```



```
// file: 4b756e64656e626574726575756e67ff0791947112122721ffffffffff
{
  "alpha_id": "Kundenbetreuung",
  "len_of_bcd": 7,
  "ton_npi": {
    "ext": true,
    "type_of_number": "international",
    "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "491721217212",
  "cap_conf_id": 255,
  "ext1_record_id": 255
}
```


ADF.USIM/EF.EXT6 (6FC8) — Extension6 (MBDN)

ADF.USIM/EF.MBI (6FC9) — Mailbox Identifier

Examples


```
// file: 0100000000
{
  "mbi_voicemail": 1,
  "mbi_fax": 0,
  "mbi_email": 0,
  "mbi_other": 0,
  "mbi_videocall": 0
}
```

ADF.USIM/EF.MWIS (6FCA) — Message Waiting Indication Status

Examples

```
// file: 0000000000
{
  "mwi_status": {
    "voicemail": false,
    "fax": false,
    "email": false,
    "other": false,
    "videomail": false
  },
  "num_waiting_voicemail": 0,
  "num_waiting_fax": 0,
  "num_waiting_email": 0,
  "num_waiting_other": 0,
  "num_waiting_videomail": null
}
```

ADF.USIM/EF.CFIS (6FCB) — Call Forwarding Indication Status

Examples

```
// file: 0100ffffffffffffffffffffffff
{
  "msp_number": 1,
  "cfu_indicator_status": {
    "voice": false,
    "fax": false,
    "data": false,
    "rfu": 0
  },
  "len_of_bcd": 255,
  "ton_npi": {
    "ext": true,
    "type_of_number": "reserved_for_extension",
    "numbering_plan_id": "reserved_for_extension"
  },
  "dialing_nr": "",
  "cap_conf_id": 255,
  "ext7_record_id": 255
}
```

ADF.USIM/EF.EXT7 (6FCC) — Extension7 (CFIS)

ADF.USIM/EF.SPDI (6FCD) — Service Provider Display Information

ADF.USIM/EF.MMSN (6FCE) — MMS Notification

ADF.USIM/EF.EXT8 (6FCF) — Extension8 (MMSN)

ADF.USIM/EF.MMSICP (6FD0) — MMS Issuer Connectivity Parameters

ADF.USIM/EF.MMSUP (6FD1) — MMS User Preferences

ADF.USIM/EF.MMSUCP (6FD2) — MMS User Connectivity Parameters

ADF.USIM/EF.NIA (6FD3) — Network's Indication of Alerting

ADF.USIM/EF.VGCS (6FB1) — Voice Group Call Service

Examples

```
// file: 92f9ffff
[
  "299"
]
```

ADF.USIM/EF.VGCSS (6FB2) — Voice Group Call Service Status

Examples

```
// file: 010000004540fc
{
  "flags": [
    1,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    1,
    0,
    1,
    0,
```

(continues on next page)

(continued from previous page)

```
    0,
    0,
    1,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    1,
    0,
    0,
    0
  ]
}
```

ADF.USIM/EF.VBS (6FB3) — Voice Broadcast Service

Examples

```
// file: 92f9ffff
[
  "299"
]
```

ADF.USIM/EF.VBSS (6FB4) — Voice Broadcast Service Status

Examples

```
// file: 010000004540fc
{
  "flags": [
    1,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
```

(continues on next page)

(continued from previous page)

```
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    1,
    0,
    1,
    0,
    0,
    0,
    1,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    1,
    0,
    0,
    0
  ]
}
```

ADF.USIM/EF.VGCSCA (6FD4) — Voice Group Call Service Ciphering Algorithm

ADF.USIM/EF.VBSCA (6FD5) — Voice Broadcast Service Ciphering Algorithm

ADF.USIM/EF.GBABP (6FD6) — GBA Bootstrapping parameters

ADF.USIM/EF.MSK (6FD7) — MBMS Service Key List

ADF.USIM/EF.MUK (6FD8) — MBMS User Key

ADF.USIM/EF.GBANL (6FDA) — GBA NAF List

ADF.USIM/EF.EHPLMN (6FD9) — Equivalent HPLMN

Examples

```
// file: 22f860
[
  {
    "mcc": "228",
    "mnc": "06"
```

(continues on next page)

(continued from previous page)

```
  }
]
```

```
// file: 330420
[
  {
    "mcc": "334",
    "mnc": "020"
  }
]
```

ADF.USIM/EF.EHPLMNPI (6FDB) — Equivalent HPLMN Presentation Indication

Examples

```
// file: 00
{
  "presentation_ind": "no_preference"
}
```

```
// file: 02
{
  "presentation_ind": "display_all"
}
```

ADF.USIM/EF.NAFKCA (6FDD) — NAF Key Centre Address

ADF.USIM/EF.SPNI (6FDE) — Service Provider Name Icon

ADF.USIM/EF.PNNI (6FDF) — PLMN Network Name Icon

ADF.USIM/EF.NCP-IP (6FE2) — Network Connectivity Parameters for USIM IP connections

ADF.USIM/EF.EPSLOCI (6FE3) — EPS location information

ADF.USIM/EF.EPSNSC (6FE4) — EPS NAS Security Context

ADF.USIM/EF.UFC (6FE6) — USAT Facility Control

ADF.USIM/EF.NASCONFIG (6FE8) — Non Access Stratum Configuration

ADF.USIM/EF.PWS (6FEC) — Public Warning System

Examples

```
// file: 00
{
  "pws_configuration": {
    "ignore_pws_in_hplmn_and_equivalent": false,
    "ignore_pws_in_vplmn": false
  }
}
```

ADF.USIM/EF.FDNURI (6FED) — Fixed Dialling Numbers URI

ADF.USIM/EF.BDNURI (6FEE) — Barred Dialling Numbers URI

ADF.USIM/EF.SDNURI (6FEF) — Service Dialling Numbers URI

ADF.USIM/EF.IPS (6FF1) — IMEI(SV) Pairing Status

ADF.USIM/EF.ePDGId (6FF3) — Home ePDG Identifier

Examples

```
// file: 801100657064672e6f736d6f636f6d2e6f7267
{
  "e_pdg_id": {
    "type_of_ePDG_address": "FQDN",
    "ePDG_address": "epdg.osmocom.org"
  }
}
```

```
// file: 800501c0a8a001
{
  "e_pdg_id": {
    "type_of_ePDG_address": "IPv4",
    "ePDG_address": "192.168.160.1"
  }
}
```

```
// file: 80110220010db80000000000000000000023
{
  "e_pdg_id": {
    "type_of_ePDG_address": "IPv6",
    "ePDG_address": "2001:db8::23"
  }
}
```

ADF.USIM/EF.ePDGSelection (6FF4) — ePDG Selection Information

Examples

```
// file: 800600f110000100
{
  "e_pdg_selection": [
    {
      "plmn": "001-01",
      "epdg_priority": 1,
      "epdg_fqdn_format": "operator_identified"
    }
  ]
}
```

```
// file: 800600110000a001
{
  "e_pdg_selection": [
    {
```

(continues on next page)

(continued from previous page)

```
    "plmn": "001-001",
    "epdg_priority": 160,
    "epdg_fqdn_format": "location_based"
   }
  ]
}
```

```
// file: 800600011000a001
{
  "e_pdg_selection": [
   {
    "plmn": "001-010",
    "epdg_priority": 160,
    "epdg_fqdn_format": "location_based"
   }
  ]
}
```

ADF.USIM/EF.ePDGIdEm (6FF5) — Emergency ePDG Identifier

Examples

```
// file: 801100657064672e6f736d6f636f6d2e6f7267
{
  "e_pdg_id": {
   "type_of_ePDG_address": "FQDN",
   "ePDG_address": "epdg.osmocom.org"
  }
}
```

```
// file: 800501c0a8a001
{
  "e_pdg_id": {
   "type_of_ePDG_address": "IPv4",
   "ePDG_address": "192.168.160.1"
  }
}
```

```
// file: 80110220010db80000000000000000000023
{
  "e_pdg_id": {
   "type_of_ePDG_address": "IPv6",
   "ePDG_address": "2001:db8::23"
  }
}
```

ADF.USIM/EF.ePDGSelectionEm (6FF6) — ePDG Selection Information for Emergency Services

Examples

```
// file: 800600f110000100
{
  "e_pdg_selection": [
    {
      "plmn": "001-01",
      "epdg_priority": 1,
      "epdg_fqdn_format": "operator_identified"
    }
  ]
}
```

```
// file: 8006001100000a001
{
  "e_pdg_selection": [
    {
      "plmn": "001-001",
      "epdg_priority": 160,
      "epdg_fqdn_format": "location_based"
    }
  ]
}
```

```
// file: 8006000110000a001
{
  "e_pdg_selection": [
    {
      "plmn": "001-010",
      "epdg_priority": 160,
      "epdg_fqdn_format": "location_based"
    }
  ]
}
```

ADF.USIM/EF.FromPreferred (6FF7) — From Preferred

ADF.USIM/EF.IMSConfigData (6FF8) — IMS Configuration Data

ADF.USIM/EF.XCAPConfigData (6FFC) — XCAP Configuration Data

ADF.USIM/EF.EARFCNList (6FFD) — EARFCN list for MTC/NB-IOT UEs

Examples

```
// file: a01a8004000100008112000001100001000002100002000003100003
[
  {
    "earfcn_list_tlv": [
      {
        "earfcn": 65536
      },
      {
        "geographical_area": [
          {
```

(continues on next page)

(continued from previous page)


```
          "latitude": 1,
          "longitude": 1048577
        },
        {
          "latitude": 2,
          "longitude": 1048578
        },
        {
          "latitude": 3,
          "longitude": 1048579
        }
      ]
    }
  ]
}
]
```



```
// file:␣
↪a03480040001000081120000011000010000021000020000031000038118000001100001000002100002000003100003000004100001
[
  {
    "earfcn_list_tlv": [
      {
        "earfcn": 65536
      },
      {
        "geographical_area": [
          {
            "latitude": 1,
            "longitude": 1048577
          },
          {
            "latitude": 2,
            "longitude": 1048578
          },
          {
            "latitude": 3,
            "longitude": 1048579
          }
        ]
      },
      {
        "geographical_area": [
          {
            "latitude": 1,
            "longitude": 1048577
          },
          {
            "latitude": 2,
            "longitude": 1048578
          },
          {
```


(continues on next page)

(continued from previous page)

```
          "latitude": 3,
          "longitude": 1048579
        },
        {
          "latitude": 4,
          "longitude": 1048580
        }
      ]
    }
  ]
 }
]
```

```
// file:␣
↪a01a8004000100008112000001100001000002100002000003100003a01a8004000200008112000011100011000012100012000013
[
  {
    "earfcn_list_tlv": [
      {
        "earfcn": 65536
      },
      {
        "geographical_area": [
          {
            "latitude": 1,
            "longitude": 1048577
          },
          {
            "latitude": 2,
            "longitude": 1048578
          },
          {
            "latitude": 3,
            "longitude": 1048579
          }
        ]
      }
    ]
  },
  {
    "earfcn_list_tlv": [
      {
        "earfcn": 131072
      },
      {
        "geographical_area": [
          {
            "latitude": 17,
            "longitude": 1048593
          },
          {
            "latitude": 18,
```

(continues on next page)

(continued from previous page)


```
        "longitude": 1048594
      },
      {
        "latitude": 19,
        "longitude": 1048595
      }
    ]
  }
 ]
}
]
```


ADF.USIM/EF.MuDMiDConfigData (6FFE) — MuD and MiD Configuration Data

ADF.USIM/EF.eAKA (6F01) — enhanced AKA support

ADF.USIM/EF.OCST (6F02) — Operator controlled signal threshold per access technology

ADF.USIM/EF.GBAUAPI (6F0A) — Access Control to GBA_U_API

The use of this EF is eescribed in 3GPP TS 31.130

ADF.USIM/EF.IMSDCI (6F0B) — IMS Data Channel Indication

See Management object as defined in 3GPP TS 24.275.

ADF.USIM/DF.PHONEBOOK (5F3A) — Phonebook

ADF.USIM/DF.PHONEBOOK/EF.PBR (4F30) — Phone Book Reference

ADF.USIM/DF.PHONEBOOK/EF.PSC (4F22) — Phone Book Synchronization Counter

ADF.USIM/DF.PHONEBOOK/EF.CC (4F23) — Change Counter

ADF.USIM/DF.PHONEBOOK/EF.PUID (4F24) — Previous Unique Identifer

ADF.USIM/DF.GSM-ACCESS (5F3B) — GSM Access

ADF.USIM/DF.GSM-ACCESS/EF.Kc (4F20) — Ciphering key Kc

Examples


```
// file: 837d783609a3858f05
{
  "kc": "837d783609a3858f",
  "cksn": 5
}
```


ADF.USIM/DF.GSM-ACCESS/EF.KcGPRS (4F52) — GPRS Ciphering key KcGPRS

Examples


```
// file: 837d783609a3858f05
{
  "kc": "837d783609a3858f",
  "cksn": 5
}
```

ADF.USIM/DF.GSM-ACCESS/EF.CPBCCH (4F63) — CPBCCH Information

ADF.USIM/DF.GSM-ACCESS/EF.InvScan (4F64) — IOnvestigation Scan

ADF.USIM/DF.WLAN (5F40) — Files for WLAN purpose

ADF.USIM/DF.WLAN/EF.Pseudo (4F41) — Pseudonym

ADF.USIM/DF.WLAN/EF.UPLMNWLAN (4F42) — User controlled PLMN selector for I-WLAN Access

ADF.USIM/DF.WLAN/EF.OPLMNWLAN (4F43) — Operator controlled PLMN selector for I-WLAN Access

ADF.USIM/DF.WLAN/EF.UWSIDL (4F44) — User controlled WLAN Specific Identifier List

ADF.USIM/DF.WLAN/EF.OWSIDL (4F45) — Operator controlled WLAN Specific Identifier List

ADF.USIM/DF.WLAN/EF.WRI (4F46) — WLAN Reauthentication Identity

ADF.USIM/DF.WLAN/EF.HWSIDL (4F47) — Home I-WLAN Specific Identifier List

ADF.USIM/DF.WLAN/EF.WEHPLMNPI (4F48) — I-WLAN Equivalent HPLMN Presentation Indication

ADF.USIM/DF.WLAN/EF.WHPI (4F49) — I-WLAN HPLMN Priority Indication

ADF.USIM/DF.WLAN/EF.WLRPLMN (4F4A) — I-WLAN Last Registered PLMN

ADF.USIM/DF.WLAN/EF.HPLMNDAI (4F4B) — HPLMN Direct Access Indicator

ADF.USIM/DF.HNB (5F50) — Files for HomeNodeB purpose

ADF.USIM/DF.HNB/EF.ACSGL (4F81) — Allowed CSG Lists

Examples

```
// file: a00d800362f21081060000000002e0
{
  "csg_list": [
    {
      "plmn": "262-01"
    },
    {
      "csg_information": {
        "csg_type": "from_other_sources",
        "hnb_name_indication": "from_other_sources",
        "csg_id": {
          "id": 23
        }
      }
    }
  ]
}
```

ADF.USIM/DF.HNB/EF.CSGT (4F82) — CSG Types

Examples

```
// file: 8906810300666f6f
[
  {
    "text_csg_type": "foo"
```

(continues on next page)

(continued from previous page)

```
  }
]
```

```
// file: 8906810300666f6f801068747470733a2f2f666f6f2e6261722f
[
  {
    "text_csg_type": "foo"
  },
  {
    "graphics_csg_type_uri": "https://foo.bar/"
  }
]
```

ADF.USIM/DF.HNB/EF.HNBN (4F83) — Home NodeB Name

Examples

```
// file: 800b8108006d61686c7a656974
{
  "hnb_name": "mahlzeit"
}
```

ADF.USIM/DF.HNB/EF.OCSGL (4F84) — Operator CSG Lists

Examples

```
// file: a010800362f210810600000000002e0820100
{
  "operator_csg_list": [
    {
      "plmn": "262-01"
    },
    {
      "csg_information": {
        "csg_type": "from_other_sources",
        "hnb_name_indication": "from_other_sources",
        "csg_id": {
          "id": 23
        }
      }
    },
    {
      "csg_display_indicator": "all_available_csg_ids"
    }
  ]
}
```

ADF.USIM/DF.HNB/EF.OCSGT (4F85) — Operator CSG Type

Examples

```
// file: 8906810300666f6f
[
  {
    "text_csg_type": "foo"
  }
]
```

```
// file: 8906810300666f6f801068747470733a2f2f666f6f2e6261722f
[
  {
    "text_csg_type": "foo"
  },
  {
    "graphics_csg_type_uri": "https://foo.bar/"
  }
]
```

ADF.USIM/DF.HNB/EF.OHNBN (4F86) — Operator Home NodeB Name

Examples

```
// file: 800b8108006d61686c7a656974
{
  "hnb_name": "mahlzeit"
}
```

ADF.USIM/DF.ProSe (5F90) — Files for ProSe purpose

ADF.USIM/DF.ProSe/EF.PROSE_MON (4F01) — ProSe Monitoring Parameters

ADF.USIM/DF.ProSe/EF.PROSE_ANN (4F02) — ProSe Announcing Parameters

ADF.USIM/DF.ProSe/EF.PROSEFUNC (4F03) — HPLMN ProSe Function

ADF.USIM/DF.ProSe/EF.PROSE_RADIO_COM (4F04) — ProSe Direct Communication Radio Parameters

ADF.USIM/DF.ProSe/EF.PROSE_RADIO_MON (4F05) — ProSe Direct Discovery Monitoring Radio Parameters

ADF.USIM/DF.ProSe/EF.PROSE_RADIO_ANN (4F06) — ProSe Direct Discovery Announcing Radio Parameters

ADF.USIM/DF.ProSe/EF.PROSE_POLICY (4F07) — ProSe Policy Parameters

ADF.USIM/DF.ProSe/EF.PROSE_PLMN (4F08) — ProSe PLMN Parameters

ADF.USIM/DF.ProSe/EF.PROSE_GC (4F09) — ProSe Group Counter

ADF.USIM/DF.ProSe/EF.PST (4F10) — ProSe Service Table

ADF.USIM/DF.ProSe/EF.UIRC (4F11) — ProSe UsageInformationReportingConfiguration

ADF.USIM/DF.ProSe/EF.PROSE_GM_DISCOVERY (4F12) — ProSe Group Member Discovery Parameters

ADF.USIM/DF.ProSe/EF.PROSE_RELAY (4F13) — ProSe Relay Parameters

ADF.USIM/DF.ProSe/EF.PROSE_RELAY_DISCOVERY (4F14) — ProSe Relay Discovery Parameters

ADF.USIM/DF.5GS (5FC0) — 5GS related files

ADF.USIM/DF.5GS/EF.5GS3GPPLOCI (4F01) — 5GS 3GPP location information

ADF.USIM/DF.5GS/EF.5GSN3GPPLOCI (4F02) — 5GS non-3GPP location information

ADF.USIM/DF.5GS/EF.5GS3GPPNSC (4F03) — 5GS 3GPP Access NAS Security Context

ADF.USIM/DF.5GS/EF.5GSN3GPPNSC (4F04) — 5GS non-3GPP Access NAS Security Context

ADF.USIM/DF.5GS/EF.5GAUTHKEYS (4F05) — 5G authentication keys

ADF.USIM/DF.5GS/EF.UAC_AIC (4F06) — UAC Access Identities Configuration

Examples

```
// file: 03
{
  "uac_access_id_config": {
    "multimedia_priority_service": true,
    "mission_critical_service": true
  }
}
```

ADF.USIM/DF.5GS/EF.SUCI_Calc_Info (4F07) — SUCI Calc Info

Examples

```
// file: a00401010000a14a80010a81204e858c4d49d1343e6181284c47ca721730c98742cb7c6182d2e8126e08088d3680010b8120d1bc365
{
  "prot_scheme_id_list": [
    {
      "priority": 0,
      "identifier": 1,
      "key_index": 1
    },
    {
      "priority": 1,
      "identifier": 0,
      "key_index": 0
    }
  ],
  "hnet_pubkey_list": [
    {
      "hnet_pubkey_identifier": 10,
      "hnet_pubkey": "4e858c4d49d1343e6181284c47ca721730c98742cb7c6182d2e8126e08088d36"
    },
    {
      "hnet_pubkey_identifier": 11,
      "hnet_pubkey": "d1bc365f4997d17ce4374e72181431cbfeba9e1b98d7618f79d48561b144672a"
    }
  ]
}
```

```
// file: a000ffffffffffffffffffffffffffffff
{
  "prot_scheme_id_list": [],
  "hnet_pubkey_list": []
}
```

```
// file: a000a100ffffffffffffffffffffffffff
{
  "prot_scheme_id_list": [],
  "hnet_pubkey_list": []
}
```

ADF.USIM/DF.5GS/EF.OPL5G (4F08) — 5GS Operator PLMN List

ADF.USIM/DF.5GS/EF.SUPI_NAI (4F09) — SUPI as Network Access Identifier

ADF.USIM/DF.5GS/EF.Routing_Indicator (4F0A) — Routing Indicator

ADF.USIM/DF.5GS/EF.URSP (4F0B) — UE Route Selector Policies per PLMN

ADF.USIM/DF.5GS/EF.TN3GPPSNN (4F0C) — Trusted non-3GPP Serving network names list

ADF.USIM/DF.5GS/EF.CAG (4F0D) — Pre-configured CAG information list EF

ADF.USIM/DF.5GS/EF.SOR-CMCI (4F0E) — Steering Of Roaming - Connected Mode Control Information

ADF.USIM/DF.5GS/EF.DRI (4F0F) — Disaster roaming information EF

ADF.USIM/DF.5GS/EF.5GSEDRX (4F10) — 5GS eDRX Parameters

ADF.USIM/DF.5GS/EF.5GNSWO_CONF (4F11) — 5G Non-Seamless WLAN Offload configuration

ADF.USIM/DF.5GS/EF.MCHPPLMN (4F15) — Multiplier Coefficient for Higher Priority PLMN search

ADF.USIM/DF.5GS/EF.KAUSF_DERIVATION (4F16) — K_AUSF derivation configuration

ADF.USIM/DF.SNPN (5FE0) — Files for SNPN purpose

ADF.USIM/DF.SNPN/EF.PWS_SNPN (4F01) — Public Warning System in SNPNs

ADF.USIM/DF.SNPN/EF.NID (4F02) — Network Identifier for SNPN

ADF.USIM/DF.5G_ProSe (5FF0) — Files for 5G ProSe purpose

ADF.USIM/DF.5G_ProSe/EF.5G_PROSE_ST (4F01) — 5G ProSe Service Table

ADF.USIM/DF.5G_ProSe/EF.5G_PROSE_DD (4F02) — 5G ProSe configuration data for direct discovery

ADF.USIM/DF.5G_ProSe/EF.5G_PROSE_DC (4F03) — 5G ProSe configuration data for direct communication

ADF.USIM/DF.5G_ProSe/EF.5G_PROSE_U2NRU (4F04) — 5G ProSe configuration data for UE-to-network relay UE

ADF.USIM/DF.5G_ProSe/EF.5G_PROSE_RU (4F05) — 5G ProSe configuration data for remote UE

ADF.USIM/DF.5G_ProSe/EF.5G_PROSE_UIR (4F06) — 5G ProSe configuration data for usage information reporting

ADF.USIM/DF.5G_ProSe/EF.5G_PROSE_U2URU (4F07) — 5G ProSe configuration data for UE-to-UE relay UE

ADF.USIM/DF.5G_ProSe/EF.5G_PROSE_EU (4F08) — 5G ProSe configuration data for end UE

ADF.USIM/DF.SAIP (5FD0) — SIMalliance Interoperable Profile

This is not really TS 31.102 but part of the eUICC Profile Package: Interoperable Format Technical Specification as released by TCA (formerly SIMalliance)

ADF.USIM/DF.SAIP/EF.SUCI_Calc_Info (4F01) — SUCI Calc Info

Examples

```
// file: a00401010000a14a80010a81204e858c4d49d1343e6181284c47ca721730c98742cb7c6182d2e8126e08088d3680010b8120d1bc365
{
  "prot_scheme_id_list": [
    {
      "priority": 0,
      "identifier": 1,
      "key_index": 1
    },
    {
      "priority": 1,
      "identifier": 0,
      "key_index": 0
    }
  ],
  "hnet_pubkey_list": [
    {
      "hnet_pubkey_identifier": 10,
      "hnet_pubkey": "4e858c4d49d1343e6181284c47ca721730c98742cb7c6182d2e8126e08088d36"
    },
    {
      "hnet_pubkey_identifier": 11,
      "hnet_pubkey": "d1bc365f4997d17ce4374e72181431cbfeba9e1b98d7618f79d48561b144672a"
    }
  ]
}
```

```
// file: a000ffffffffffffffffffffffffffff
{
  "prot_scheme_id_list": [],
  "hnet_pubkey_list": []
}
```

```
// file: a000a100ffffffffffffffffffffffffff
{
  "prot_scheme_id_list": [],
  "hnet_pubkey_list": []
}
```

ADF.USIM/DF.5MBSUECONFIG (5FF1)

ADF.USIM/DF.5MBSUECONFIG/EF.5MBSUECONFIG (4F01) — 5MBS UE pre-configuration

ADF.USIM/EF.IMSI (6F07) — IMSI

Examples

```
// file: 082982608200002080
{
  "imsi": "228062800000208"
}
```

```
// file: 082926101160845740
{
  "imsi": "262011106487504"
}
```

1.2.3 ADF.ISIM / TS 31.103

ADF.ISIM/EF.IMPI (6F02) — IMS private user identity

Examples

```
// file:␣
↪803137333830303630303030303031303140696d732e6d6e633030302e6d63633733382e336770706e6574776f726b2e6f7267
{
  "nai": "738006000000101@ims.mnc000.mcc738.3gppnetwork.org"
}
```

ADF.ISIM/EF.DOMAIN (6F03) — Home Network Domain Name

Examples

```
// file: 8021696d732e6d6e633030302e6d63633733382e336770706e6574776f726b2e6f7267
{
  "domain": "ims.mnc000.mcc738.3gppnetwork.org"
}
```

ADF.ISIM/EF.IMPU (6F04) — IMS public user identity

Examples

```
// file:␣
↪80357369703a373338303036303030303030313031303140696d732e6d6e633030302e6d63633733382e336770706e6574776f726b2e6f72
{
  "impu": "sip:738006000000101@ims.mnc000.mcc738.3gppnetwork.org"
}
```

ADF.ISIM/EF.AD (6FAD) — Administrative Data

Examples

```
// file: 00ffff
{
  "ms_operation_mode": "normal",
  "rfu1": 255,
  "rfu2": 127,
  "ofm": true,
  "extensions": null
}
```

ADF.ISIM/EF.IST (6F07) — ISIM Service Table

ADF.ISIM/EF.P-CSCF (6F09) — P-CSCF Address

Examples

```
// file:␣
↪802c0070637363662e696d732e6d6e633030302e6d63633733382e7075622e336770706e6574776f726b2e6f7267
{
  "pcscf_address": {
    "address": "pcscf.ims.mnc000.mcc738.pub.3gppnetwork.org",
    "type_of_address": "FQDN"
  }
}
```

```
// file: 800501c0a80c22
{
  "pcscf_address": {
    "address": "192.168.12.34",
    "type_of_address": "IPv4"
  }
}
```

```
// file: 801102fe80000000000000042d7fffe530335
{
  "pcscf_address": {
    "address": "fe80::42:d7ff:fe53:335",
    "type_of_address": "IPv6"
  }
}
```

ADF.ISIM/EF.GBABP (6FD5) — GBA Bootstrapping

ADF.ISIM/EF.GBANL (6FD7) — GBA NAF List

ADF.ISIM/EF.NAFKCA (6FDD) — NAF Key Centre Address

Examples

```
// file: 80296273662e696d732e6d6e633030302e6d63633733382e7075622e336770706e6574776f726b2e6f7267
{
  "naf_key_centre_address": "bsf.ims.mnc000.mcc738.pub.3gppnetwork.org"
}
```

```
// file:␣
↪8030656e65746e61667830312e696d732e6d6e633030302e6d63633733382e7075622e336770706e6574776f726b2e6f7267
{
  "naf_key_centre_address": "enetnafx01.ims.mnc000.mcc738.pub.3gppnetwork.org"
}
```

ADF.ISIM/EF.UICCIARI (6FE7) — UICC IARI

ADF.ISIM/EF.WebRTCURI (6FFA) — WebRTC URI

1.2.4 DF.GSM + DF.TELECOM / TS 51.011 (SIM)

DF.TELECOM (7F10)

DF.TELECOM/EF.ADN (6F3A) — Abbreviated Dialing Numbers

Examples

```
// file: 42204841203120536963ffffffffff068106285608 10ffffffffff
{
  "alpha_id": "B HA 1 Sic",
  "len_of_bcd": 6,
  "ton_npi": {
    "ext": true,
    "type_of_number": "unknown",
    "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "6082658001",
  "cap_conf_id": 255,
  "ext1_record_id": 255
}
```

```
// file: 4b756e64656e626574726575756e67ff0791947112122721ffffffffff
{
  "alpha_id": "Kundenbetreuung",
  "len_of_bcd": 7,
  "ton_npi": {
    "ext": true,
    "type_of_number": "international",
    "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "491721217212",
  "cap_conf_id": 255,
  "ext1_record_id": 255
}
```

DF.TELECOM/EF.FDN (6F3B) — Fixed dialling numbers

Examples

```
// file: 42204841203120536963ffffffffff068106285608 10ffffffffff
{
  "alpha_id": "B HA 1 Sic",
  "len_of_bcd": 6,
```

(continues on next page)

(continued from previous page)


```
  "ton_npi": {
    "ext": true,
    "type_of_number": "unknown",
    "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "6082658001",
  "cap_conf_id": 255,
  "ext1_record_id": 255
}
```



```
// file: 4b756e64656e626574726575756e67ff07919471121227 21ffffffffff
{
  "alpha_id": "Kundenbetreuung",
  "len_of_bcd": 7,
  "ton_npi": {
    "ext": true,
    "type_of_number": "international",
    "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "491721217212",
  "cap_conf_id": 255,
  "ext1_record_id": 255
}
```


DF.TELECOM/EF.SMS (6F3C) — Short messages

DF.TELECOM/EF.CCP (6F3D) — Capability Configuration Parameters

DF.TELECOM/EF.ECCP (6F4F) — Extended Capability Configuration Parameters

DF.TELECOM/EF.MSISDN (6F40) — MSISDN

Examples


```
// file: ffffffffffffffffffffffffffffffff04b12143f5ffffffffffffff
{
  "alpha_id": "",
  "len_of_bcd": 4,
  "ton_npi": {
    "ext": true,
    "type_of_number": "network_specific",
    "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "12345"
}
```



```
// file: 456967656e65205275666e756d6d6572ffffffff0891947172199181f3ffffffff
{
  "alpha_id": "Eigene Rufnummer",
  "len_of_bcd": 8,
  "ton_npi": {
    "ext": true,
    "type_of_number": "international",
```


(continues on next page)

(continued from previous page)


```
    "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "4917279119183"
}
```


DF.TELECOM/EF.SMSP (6F42) — Short message service parameters

Examples


```
// file: 534d5343ffffffffffffffffffffffe1ffffffffffffffffffff089194519710909 9f9fffff0000a9
{
  "alpha_id": "SMSC",
  "parameter_indicators": {
    "tp_dest_addr": false,
    "tp_sc_addr": true,
    "tp_pid": true,
    "tp_dcs": true,
    "tp_vp": true
  },
  "tp_dest_addr": {
    "length": 255,
    "ton_npi": {
      "ext": true,
      "type_of_number": "reserved_for_extension",
      "numbering_plan_id": "reserved_for_extension"
    },
    "call_number": ""
  },
  "tp_sc_addr": {
    "length": 8,
    "ton_npi": {
      "ext": true,
      "type_of_number": "international",
      "numbering_plan_id": "isdn_e164"
    },
    "call_number": "4915790109999"
  },
  "tp_pid": "00",
  "tp_dcs": "00",
  "tp_vp_minutes": 4320
}
```



```
// file: e1ffffffffffffffffffff089194519710909 9f9fffff0000a9
{
  "alpha_id": "",
  "parameter_indicators": {
    "tp_dest_addr": false,
    "tp_sc_addr": true,
    "tp_pid": true,
    "tp_dcs": true,
    "tp_vp": true
  },
```


(continues on next page)

(continued from previous page)


```
  "tp_dest_addr": {
    "length": 255,
    "ton_npi": {
      "ext": true,
      "type_of_number": "reserved_for_extension",
      "numbering_plan_id": "reserved_for_extension"
    },
    "call_number": ""
  },
  "tp_sc_addr": {
    "length": 8,
    "ton_npi": {
      "ext": true,
      "type_of_number": "international",
      "numbering_plan_id": "isdn_e164"
    },
    "call_number": "4915790109999"
  },
  "tp_pid": "00",
  "tp_dcs": "00",
  "tp_vp_minutes": 4320
}
```



```
// file: 454e6574776f726b73ffffffffffff1fffffffffffffffffffffffffffffffffffffffffffffffffffff0000a7
{
  "alpha_id": "ENetworks",
  "parameter_indicators": {
    "tp_dest_addr": false,
    "tp_sc_addr": true,
    "tp_pid": true,
    "tp_dcs": true,
    "tp_vp": false
  },
  "tp_dest_addr": {
    "length": 255,
    "ton_npi": {
      "ext": true,
      "type_of_number": "reserved_for_extension",
      "numbering_plan_id": "reserved_for_extension"
    },
    "call_number": ""
  },
  "tp_sc_addr": {
    "length": 255,
    "ton_npi": {
      "ext": true,
      "type_of_number": "reserved_for_extension",
      "numbering_plan_id": "reserved_for_extension"
    },
    "call_number": ""
  },
  "tp_pid": "00",
```


(continues on next page)

(continued from previous page)


```
  "tp_dcs": "00",
  "tp_vp_minutes": 1440
}
```



```
// file: ffffffffffffffffffffffffffffffffffffffffffffffffdffffffffffffffffff07919403214365f7ffffffffff
{
  "alpha_id": "",
  "parameter_indicators": {
    "tp_dest_addr": false,
    "tp_sc_addr": true,
    "tp_pid": false,
    "tp_dcs": false,
    "tp_vp": false
  },
  "tp_dest_addr": {
    "length": 255,
    "ton_npi": {
      "ext": true,
      "type_of_number": "reserved_for_extension",
      "numbering_plan_id": "reserved_for_extension"
    },
    "call_number": ""
  },
  "tp_sc_addr": {
    "length": 7,
    "ton_npi": {
      "ext": true,
      "type_of_number": "international",
      "numbering_plan_id": "isdn_e164"
    },
    "call_number": "49301234567"
  },
  "tp_pid": "ff",
  "tp_dcs": "ff",
  "tp_vp_minutes": 635040
}
```



```
// file: ffffffffffffffffffffffffffffffffffffffffffffffffc0b919403214365f7fffffff07919403214365f7ffffffffff
{
  "alpha_id": "",
  "parameter_indicators": {
    "tp_dest_addr": true,
    "tp_sc_addr": true,
    "tp_pid": false,
    "tp_dcs": false,
    "tp_vp": false
  },
  "tp_dest_addr": {
    "length": 11,
    "ton_npi": {
      "ext": true,
```


(continues on next page)

(continued from previous page)


```
    "type_of_number": "international",
    "numbering_plan_id": "isdn_e164"
   },
   "call_number": "49301234567"
  },
  "tp_sc_addr": {
   "length": 7,
   "ton_npi": {
    "ext": true,
    "type_of_number": "international",
    "numbering_plan_id": "isdn_e164"
   },
   "call_number": "49301234567"
  },
  "tp_pid": "ff",
  "tp_dcs": "ff",
  "tp_vp_minutes": 635040
}
```


DF.TELECOM/EF.SMSS (6F43) — SMS status

DF.TELECOM/EF.LND (6F44) — Last Number Dialled

Examples


```
// file: 42204841203120536963ffffffffff068106285608 10ffffffffff
{
  "alpha_id": "B HA 1 Sic",
  "len_of_bcd": 6,
  "ton_npi": {
   "ext": true,
   "type_of_number": "unknown",
   "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "6082658001",
  "cap_conf_id": 255,
  "ext1_record_id": 255
}
```



```
// file: 4b756e64656e626574726575756e67ff0791947112122721ffffffffff
{
  "alpha_id": "Kundenbetreuung",
  "len_of_bcd": 7,
  "ton_npi": {
   "ext": true,
   "type_of_number": "international",
   "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "491721217212",
  "cap_conf_id": 255,
  "ext1_record_id": 255
}
```

DF.TELECOM/EF.SDN (6F49) — Service Dialling Numbers

Examples

```
// file: 42204841203120536963ffffffffff06810628560810ffffffffff
{
  "alpha_id": "B HA 1 Sic",
  "len_of_bcd": 6,
  "ton_npi": {
    "ext": true,
    "type_of_number": "unknown",
    "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "6082658001",
  "cap_conf_id": 255,
  "ext1_record_id": 255
}
```

```
// file: 4b756e64656e626574726575756e67ff07919471121227 21ffffffffff
{
  "alpha_id": "Kundenbetreuung",
  "len_of_bcd": 7,
  "ton_npi": {
    "ext": true,
    "type_of_number": "international",
    "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "491721217212",
  "cap_conf_id": 255,
  "ext1_record_id": 255
}
```

DF.TELECOM/EF.EXT1 (6F4A) — Extension1 (ADN/SSC)

DF.TELECOM/EF.EXT2 (6F4B) — Extension2 (FDN/SSC)

DF.TELECOM/EF.EXT3 (6F4C) — Extension3 (SDN)

DF.TELECOM/EF.BDN (6F4D) — Barred Dialling Numbers

Examples

```
// file: 42204841203120536963ffffffffff06810628560810ffffffffff
{
  "alpha_id": "B HA 1 Sic",
  "len_of_bcd": 6,
  "ton_npi": {
    "ext": true,
    "type_of_number": "unknown",
    "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "6082658001",
  "cap_conf_id": 255,
  "ext1_record_id": 255
}
```

```
// file: 4b756e64656e626574726575756e67ff0791947112122721ffffffffff
{
  "alpha_id": "Kundenbetreuung",
  "len_of_bcd": 7,
  "ton_npi": {
    "ext": true,
    "type_of_number": "international",
    "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "491721217212",
  "cap_conf_id": 255,
  "ext1_record_id": 255
}
```

DF.TELECOM/EF.EXT4 (6F4E) — Extension4 (BDN/SSC)

DF.TELECOM/EF.SMSR (6F47) — SMS status reports

DF.TELECOM/EF.CMI (6F58) — Comparison Method Information

DF.TELECOM/DF.PHONEBOOK (5F3A) — Phonebook

DF.TELECOM/DF.PHONEBOOK/EF.PBR (4F30) — Phone Book Reference

DF.TELECOM/DF.PHONEBOOK/EF.PSC (4F22) — Phone Book Synchronization Counter

DF.TELECOM/DF.PHONEBOOK/EF.CC (4F23) — Change Counter

DF.TELECOM/DF.PHONEBOOK/EF.PUID (4F24) — Previous Unique Identifer

DF.TELECOM/DF.MULTIMEDIA (5F3B) — Multimedia

DF.TELECOM/DF.MULTIMEDIA/EF.MML (4F47) — Multimedia Messages List

DF.TELECOM/DF.MULTIMEDIA/EF.MMDF (4F48) — Multimedia Messages Data File

DF.TELECOM/DF.MCS (5F3D) — Mission Critical Services

DF.TELECOM/DF.MCS/EF.MST (4F01) — MCS Service Table

DF.TELECOM/DF.MCS/EF.MCS_CONFIG (4F02) — MCS configuration data

DF.TELECOM/DF.V2X (5F3E) — Vehicle to X

DF.TELECOM/DF.V2X/EF.VST (4F01) — V2X Service Table

DF.TELECOM/DF.V2X/EF.V2X_CONFIG (4F02) — V2X configuration data

DF.GSM (7F20) — GSM Network related files

DF.GSM/EF.LP (6F05) — Language Preference

Examples

```
// file: 24
[
  "24"
]
```

DF.GSM/EF.IMSI (6F07) — IMSI

Examples

```
// file: 082982608200002080
{
  "imsi": "228062800000208"
}
```

```
// file: 082926101160845740
{
  "imsi": "262011106487504"
}
```

DF.GSM/EF.Kc (6F20) — Ciphering key Kc

Examples

```
// file: 837d783609a3858f05
{
  "kc": "837d783609a3858f",
  "cksn": 5
}
```

DF.GSM/EF.PLMNsel (6F30) — PLMN selector

Examples

```
// file: 22f860
[
  {
    "mcc": "228",
    "mnc": "06"
  }
]
```

```
// file: 330420
[
  {
    "mcc": "334",
    "mnc": "020"
  }
]
```

DF.GSM/EF.HPPLMN (6F31) — Higher Priority PLMN search period

DF.GSM/EF.ACMmax (6F37) — ACM maximum value

Examples

```
// file: 000000
{
  "acm_max": 0
}
```

DF.GSM/EF.SST (6F38) — SIM service table

DF.GSM/EF.ACM (6F39) — Accumulated call meter

DF.GSM/EF.GID1 (6F3E) — Group Identifier Level 1

DF.GSM/EF.GID2 (6F3F) — Group Identifier Level 2

DF.GSM/EF.SPN (6F46) — Service Provider Name

Examples

```
// file: 0147534d2d52204348ffffffffffffff
{
  "rfu": 0,
  "hide_in_oplmn": false,
  "show_in_hplmn": true,
  "spn": "GSM-R CH"
}
```

DF.GSM/EF.PUCT (6F41) — Price per unit and currency table

DF.GSM/EF.CBMI (6F45) — Cell Broadcast message identifier selection

DF.GSM/EF.BCCH (6F74) — Broadcast control channels

DF.GSM/EF.ACC (6F78) — Access Control Class

Examples

```
// file: 0000
{
  "ACC0": false,
  "ACC1": false,
  "ACC2": false,
  "ACC3": false,
  "ACC4": false,
  "ACC5": false,
  "ACC6": false,
  "ACC7": false,
  "ACC8": false,
  "ACC9": false,
  "ACC10": false,
  "ACC11": false,
  "ACC12": false,
  "ACC13": false,
  "ACC14": false,
  "ACC15": false
}
```

```
// file: 0001
{
  "ACC0": true,
  "ACC1": false,
  "ACC2": false,
  "ACC3": false,
  "ACC4": false,
```

(continues on next page)

(continued from previous page)

```
  "ACC5": false,
  "ACC6": false,
  "ACC7": false,
  "ACC8": false,
  "ACC9": false,
  "ACC10": false,
  "ACC11": false,
  "ACC12": false,
  "ACC13": false,
  "ACC14": false,
  "ACC15": false
}
```

```
// file: 0002
{
  "ACC0": false,
  "ACC1": true,
  "ACC2": false,
  "ACC3": false,
  "ACC4": false,
  "ACC5": false,
  "ACC6": false,
  "ACC7": false,
  "ACC8": false,
  "ACC9": false,
  "ACC10": false,
  "ACC11": false,
  "ACC12": false,
  "ACC13": false,
  "ACC14": false,
  "ACC15": false
}
```

```
// file: 0100
{
  "ACC0": false,
  "ACC1": false,
  "ACC2": false,
  "ACC3": false,
  "ACC4": false,
  "ACC5": false,
  "ACC6": false,
  "ACC7": false,
  "ACC8": true,
  "ACC9": false,
  "ACC10": false,
  "ACC11": false,
  "ACC12": false,
  "ACC13": false,
  "ACC14": false,
  "ACC15": false
}
```

```
// file: 8000
{
  "ACC0": false,
  "ACC1": false,
  "ACC2": false,
  "ACC3": false,
  "ACC4": false,
  "ACC5": false,
  "ACC6": false,
  "ACC7": false,
  "ACC8": false,
  "ACC9": false,
  "ACC10": false,
  "ACC11": false,
  "ACC12": false,
  "ACC13": false,
  "ACC14": false,
  "ACC15": true
}
```

```
// file: 802b
{
  "ACC0": true,
  "ACC1": true,
  "ACC2": false,
  "ACC3": true,
  "ACC4": false,
  "ACC5": true,
  "ACC6": false,
  "ACC7": false,
  "ACC8": false,
  "ACC9": false,
  "ACC10": false,
  "ACC11": false,
  "ACC12": false,
  "ACC13": false,
  "ACC14": false,
  "ACC15": true
}
```

DF.GSM/EF.FPLMN (6F7B) — Forbidden PLMNs

Examples

```
// file: 22f860
[
  {
    "mcc": "228",
    "mnc": "06"
  }
]
```

```
// file: 330420
[
  {
    "mcc": "334",
    "mnc": "020"
  }
]
```

DF.GSM/EF.LOCI (6F7E) — Location Information

Examples

```
// file: 7802570222f81009780000
{
  "tmsi": "78025702",
  "lai": "22f8100978",
  "tmsi_time": 0,
  "lu_status": "updated"
}
```

DF.GSM/EF.AD (6FAD) — Administrative Data

Examples

```
// file: 00ffff
{
  "ms_operation_mode": "normal",
  "rfu1": 255,
  "rfu2": 127,
  "ofm": true,
  "extensions": null
}
```

DF.GSM/EF.Phase (6FAE) — Phase identification

DF.GSM/EF.VGCS (6FB1) — Voice Group Call Service

Examples

```
// file: 92f9ffff
[
  "299"
]
```

DF.GSM/EF.VGCSS (6FB2) — Voice Group Call Service Status

Examples

```
// file: 010000004540fc
{
  "flags": [
    1,
    0,
```

(continues on next page)

(continued from previous page)


```
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    1,
    0,
    1,
    0,
    0,
    0,
    1,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    1,
    0,
    0,
    0
  ]
}
```

DF.GSM/EF.VBS (6FB3) — Voice Broadcast Service

Examples

```
// file: 92f9ffff
[
  "299"
]
```

DF.GSM/EF.VBSS (6FB4) — Voice Broadcast Service Status

Examples

```
// file: 010000004540fc
{
  "flags": [
    1,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    1,
    0,
    1,
    0,
    0,
    0,
```

(continues on next page)

(continued from previous page)

```
    1,
    0,
    0,
    0,
    0,
    0,
    0,
    0,
    1,
    0,
    0,
    0
  ]
}
```

DF.GSM/EF.eMLPP (6FB5) — enhanced Multi Level Pre-emption and Priority

Examples

```
// file: 7c04
{
  "levels": {
    "A": false,
    "B": false,
    "zero": true,
    "one": true,
    "two": true,
    "three": true,
    "four": true
  },
  "fast_call_setup_cond": {
    "A": false,
    "B": false,
    "zero": true,
    "one": false,
    "two": false,
    "three": false,
    "four": false
  }
}
```

DF.GSM/EF.AAeM (6FB6) — Automatic Answer for eMLPP Service

Examples

```
// file: 3c
{
  "auto_answer_prio_levels": {
    "A": false,
    "B": false,
    "zero": true,
    "one": true,
```

(continues on next page)

(continued from previous page)

```
    "two": true,
    "three": true,
    "four": false
  }
}
```

DF.GSM/EF.CBMID (6F48) — Cell Broadcast Message Identifier for Data Download

DF.GSM/EF.ECC (6FB7) — Emergency Call Codes

DF.GSM/EF.CBMIR (6F50) — Cell Broadcast message identifier range selection

DF.GSM/EF.DCK (6F2C) — Depersonalisation Control Keys

DF.GSM/EF.CNL (6F32) — Co-operative Network List

DF.GSM/EF.NIA (6F51) — Network's Indication of Alerting

DF.GSM/EF.KcGPRS (6F52) — GPRS Ciphering key KcGPRS

Examples

```
// file: 837d783609a3858f05
{
  "kc": "837d783609a3858f",
  "cksn": 5
}
```

DF.GSM/EF.LOCIGPRS (6F53) — GPRS Location Information

Examples

```
// file: ffffffffffff22f8990000ff01
{
  "ptmsi": "fffffff",
  "ptmsi_sig": "fffff",
  "rai": "22f8990000ff",
  "rau_status": "not_updated"
}
```

DF.GSM/EF.SUME (6F54) — SetUpMenu Elements

DF.GSM/EF.PLMNwAcT (6F60) — User controlled PLMN Selector with Access Technology

Examples

```
// file: 62f2104000
[
  {
    "mcc": "262",
    "mnc": "01",
    "act": [
      "E-UTRAN NB-S1",
      "E-UTRAN WB-S1"
    ]
```

(continues on next page)

(continued from previous page)

```
  }
]
```

```
// file: 62f2108000
[
  {
    "mcc": "262",
    "mnc": "01",
    "act": [
      "UTRAN"
    ]
  }
]
```

```
// file: 62f220488c
[
  {
    "mcc": "262",
    "mnc": "02",
    "act": [
      "E-UTRAN NB-S1",
      "E-UTRAN WB-S1",
      "EC-GSM-IoT",
      "GSM",
      "NG-RAN"
    ]
  }
]
```

DF.GSM/EF.OPLMNwAcT (6F61) — Operator controlled PLMN Selector with Access Technology

Examples

```
// file: 62f2104000
[
  {
    "mcc": "262",
    "mnc": "01",
    "act": [
      "E-UTRAN NB-S1",
      "E-UTRAN WB-S1"
    ]
  }
]
```

```
// file: 62f2108000
[
  {
    "mcc": "262",
    "mnc": "01",
    "act": [
```

(continues on next page)

(continued from previous page)

```
    "UTRAN"
  ]
 }
]
```

```
// file: 62f220488c
[
 {
  "mcc": "262",
  "mnc": "02",
  "act": [
   "E-UTRAN NB-S1",
   "E-UTRAN WB-S1",
   "EC-GSM-IoT",
   "GSM",
   "NG-RAN"
  ]
 }
]
```

DF.GSM/EF.HPLMNwAcT (6F62) — HPLMN Selector with Access Technology

Examples

```
// file: 62f2104000
[
 {
  "mcc": "262",
  "mnc": "01",
  "act": [
   "E-UTRAN NB-S1",
   "E-UTRAN WB-S1"
  ]
 }
]
```

```
// file: 62f2108000
[
 {
  "mcc": "262",
  "mnc": "01",
  "act": [
   "UTRAN"
  ]
 }
]
```

```
// file: 62f220488c
[
 {
  "mcc": "262",
```

(continues on next page)

(continued from previous page)


```
    "mnc": "02",
    "act": [
      "E-UTRAN NB-S1",
      "E-UTRAN WB-S1",
      "EC-GSM-IoT",
      "GSM",
      "NG-RAN"
    ]
  }
]
```


DF.GSM/EF.CPBCCH (6F63) — CPBCCH Information

DF.GSM/EF.InvScan (6F64) — IOnvestigation Scan

DF.GSM/EF.PNN (6FC5) — PLMN Network Name

DF.GSM/EF.OPL (6FC6) — Operator PLMN List

Examples


```
// file: 62f2100000fffe01
{
  "lai": {
    "mcc_mnc": "262-01",
    "lac_min": "0000",
    "lac_max": "fffe"
  },
  "pnn_record_id": 1
}
```



```
// file: 216354789abcde12
{
  "lai": {
    "mcc_mnc": "123-456",
    "lac_min": "789a",
    "lac_max": "bcde"
  },
  "pnn_record_id": 18
}
```


DF.GSM/EF.MBDN (6FC7) — Mailbox Dialling Numbers

Examples


```
// file: 42204841203120536963ffffffffff068106285608 10ffffffffff
{
  "alpha_id": "B HA 1 Sic",
  "len_of_bcd": 6,
  "ton_npi": {
    "ext": true,
    "type_of_number": "unknown",
    "numbering_plan_id": "isdn_e164"
```


(continues on next page)

(continued from previous page)


```
  },
  "dialing_nr": "6082658001",
  "cap_conf_id": 255,
  "ext1_record_id": 255
}
```

```
// file: 4b756e64656e626574726575756e67ff07919471121227 21ffffffffff
{
  "alpha_id": "Kundenbetreuung",
  "len_of_bcd": 7,
  "ton_npi": {
    "ext": true,
    "type_of_number": "international",
    "numbering_plan_id": "isdn_e164"
  },
  "dialing_nr": "491721217212",
  "cap_conf_id": 255,
  "ext1_record_id": 255
}
```


DF.GSM/EF.MBI (6FC9) — Mailbox Identifier

Examples


```
// file: 0100000000
{
  "mbi_voicemail": 1,
  "mbi_fax": 0,
  "mbi_email": 0,
  "mbi_other": 0,
  "mbi_videocall": 0
}
```


DF.GSM/EF.MWIS (6FCA) — Message Waiting Indication Status

Examples


```
// file: 0000000000
{
  "mwi_status": {
    "voicemail": false,
    "fax": false,
    "email": false,
    "other": false,
    "videomail": false
  },
  "num_waiting_voicemail": 0,
  "num_waiting_fax": 0,
  "num_waiting_email": 0,
  "num_waiting_other": 0,
  "num_waiting_videomail": null
}
```

DF.GSM/EF.CFIS (6FCB) — Call Forwarding Indication Status

Examples

```
// file: 0100ffffffffffffffffffffffff
{
  "msp_number": 1,
  "cfu_indicator_status": {
    "voice": false,
    "fax": false,
    "data": false,
    "rfu": 0
  },
  "len_of_bcd": 255,
  "ton_npi": {
    "ext": true,
    "type_of_number": "reserved_for_extension",
    "numbering_plan_id": "reserved_for_extension"
  },
  "dialing_nr": "",
  "cap_conf_id": 255,
  "ext7_record_id": 255
}
```

DF.GSM/EF.EXT6 (6FC8) — Extension6 (MBDN)

DF.GSM/EF.EXT7 (6FCC) — Extension7 (CFIS)

DF.GSM/EF.SPDI (6FCD) — Service Provider Display Information

DF.GSM/EF.MMSN (6FCE) — MMS Notification

DF.GSM/EF.EXT8 (6FCF) — Extension8 (MMSN)

DF.GSM/EF.MMSICP (6FD0) — MMS Issuer Connectivity Parameters

DF.GSM/EF.MMSUP (6FD1) — MMS User Preferences

DF.GSM/EF.MMSUCP (6FD2) — MMS User Connectivity Parameters

1.2.5 CDMA / IS-820 (RUIM)

DF.CDMA (7F25) — CDMA related files (3GPP2 C.S0023-D)

DF.CDMA/EF.CST (6F32) — CDMA Service Table

DF.CDMA/EF.SPN (6F41) — Service Provider Name

3.4.31 CDMA Home Service Provider Name

Examples

```
// file: 010801536b796c696e6b204e57ffffffffffffffffffffffffffffffffffffffff
{
  "rfu1": 0,
  "show_in_hsa": true,
  "rfu2": 0,
  "char_encoding": 8,
  "lang_ind": 1,
```

(continues on next page)

(continued from previous page)

```
  "spn": "Skylink NW"
}
```

DF.CDMA/EF.AD (6F43) — Administrative Data

3.4.33 Administrative Data

Examples

```
// file: 000000
{
  "ms_operation_mode": "normal",
  "additional_info": "0000",
  "rfu": ""
}
```

DF.CDMA/EF.SMS (6F3C) — Short messages

3.4.27 Short Messages

1.2.6 DF.EIRENE / GSM-R

DF.EIRENE/EF.FN (6FF1) — Functional numbers

Section 7.2

Examples

```
// file: 40315801000010ff01
{
  "functional_number_and_type": {
    "functional_number": "04138510000001f",
    "presentation_of_only_this_fn": true,
    "permanent_fn": true
  },
  "list_number": 1
}
```

DF.EIRENE/EF.CallconfC (6FF2) — Call Configuration of emergency calls Configuration

Section 7.3

Examples

```
// file: 026121ffffffffff1e000a040a01025360079579242 6f0
{
  "pl_conf": 3,
  "conf_nr": "1612fffffffff",
  "max_rand": 30,
  "n_ack_max": 10,
  "pl_ack": 1,
  "n_nested_max": 10,
  "train_emergency_gid": 1,
  "shunting_emergency_gid": 2,
```

(continues on next page)

(continued from previous page)

```
  "imei": "350670599742620f"
}
```

DF.EIRENE/EF.CallconfI (6FF3) — Call Configuration of emergency calls Information

Section 7.5

DF.EIRENE/EF.Shunting (6FF4) — Shunting

Section 7.6

Examples

```
// file: 03f8ffffff000000
{
  "common_gid": 3,
  "shunting_gid": "f8ffffff000000"
}
```

DF.EIRENE/EF.GsmrPLMN (6FF5) — GSM-R network selection

Section 7.7

Examples

```
// file: 22f860f86f8d6f8e01
{
  "plmn": "228-06",
  "class_of_network": {
    "supported": {
      "vbs": true,
      "vgcs": true,
      "emlpp": true,
      "fn": true,
      "eirene": true
    },
    "preference": 0
  },
  "ic_incoming_ref_tbl": "6f8d",
  "outgoing_ref_tbl": "6f8e",
  "ic_table_ref": "01"
}
```

```
// file: 22f810416f8d6f8e02
{
  "plmn": "228-01",
  "class_of_network": {
    "supported": {
      "vbs": false,
      "vgcs": false,
      "emlpp": false,
      "fn": true,
      "eirene": false
```

(continues on next page)

(continued from previous page)


```
    },
    "preference": 1
  },
  "ic_incoming_ref_tbl": "6f8d",
  "outgoing_ref_tbl": "6f8e",
  "ic_table_ref": "02"
}
```


DF.EIRENE/EF.IC (6F8D) — International Code

Section 7.8

Examples


```
// file: f06f8e40f10001
{
  "next_table_type": "decision",
  "id_of_next_table": "6f8e",
  "ic_decision_value": "041f",
  "network_string_table_index": 1
}
```

```
// file: ffffffffffff
{
  "next_table_type": "empty",
  "id_of_next_table": "ffff",
  "ic_decision_value": "ffff",
  "network_string_table_index": 65535
}
```


DF.EIRENE/EF.NW (6F80) — Network Name

Section 7.9

Examples


```
// file: 47534d2d52204348
"GSM-R CH"
```

```
// file: 537769737347534d
"SwissGSM"
```

```
// file: 47534d2d52204442
"GSM-R DB"
```

```
// file: 47534d2d52524649
"GSM-RRFI"
```

DF.EIRENE/EF.CT (6F8E) — Call Type

Section 8.4

Examples

```
// file: f26f87f0ff00
{
  "next_table_type": "num_dial_digits",
  "id_of_next_table": "6f87",
  "decision_value": "0fff",
  "string_table_index": 0
}
```

```
// file: f06f8ff1ff01
{
  "next_table_type": "decision",
  "id_of_next_table": "6f8f",
  "decision_value": "1fff",
  "string_table_index": 1
}
```

```
// file: f16f89f5ff05
{
  "next_table_type": "predefined",
  "id_of_next_table": "6f89",
  "decision_value": "5fff",
  "string_table_index": 5
}
```

DF.EIRENE/EF.SC (6F8F) — Short Code

Section 8.4

Examples

```
// file: f26f87f0ff00
{
  "next_table_type": "num_dial_digits",
  "id_of_next_table": "6f87",
  "decision_value": "0fff",
  "string_table_index": 0
}
```

```
// file: f06f8ff1ff01
{
  "next_table_type": "decision",
  "id_of_next_table": "6f8f",
  "decision_value": "1fff",
  "string_table_index": 1
}
```

```
// file: f16f89f5ff05
{
```

(continues on next page)

(continued from previous page)


```
  "next_table_type": "predefined",
  "id_of_next_table": "6f89",
  "decision_value": "5fff",
  "string_table_index": 5
}
```


DF.EIRENE/EF.FC (6F88) — Function Code

Section 8.5

Examples


```
// record 1: f26f85
{
  "next_table_type": "num_dial_digits",
  "id_of_next_table": "6f85"
}
```

```
// record 2: f0ffc8
{
  "predefined_value1": "0fff",
  "string_table_index1": 200
}
```


DF.EIRENE/EF.Service (6F89) — VGCS/VBS Service Code

Section 8.5

Examples


```
// record 1: f26f85
{
  "next_table_type": "num_dial_digits",
  "id_of_next_table": "6f85"
}
```

```
// record 2: f0ffc8
{
  "predefined_value1": "0fff",
  "string_table_index1": 200
}
```


DF.EIRENE/EF.Call (6F8A) — First digit of the group ID

Section 8.5

Examples


```
// record 1: f26f85
{
  "next_table_type": "num_dial_digits",
  "id_of_next_table": "6f85"
}
```

```
// record 2: f0ffc8
{
  "predefined_value1": "0fff",
  "string_table_index1": 200
}
```

DF.EIRENE/EF.FctTeam (6F8B) — Call Type 6 Team Type + Team member function

Section 8.5

Examples

```
// record 1: f26f85
{
  "next_table_type": "num_dial_digits",
  "id_of_next_table": "6f85"
}
```

```
// record 2: f0ffc8
{
  "predefined_value1": "0fff",
  "string_table_index1": 200
}
```

DF.EIRENE/EF.Controller (6F92) — Call Type 7 Controller function code

Section 8.5

Examples

```
// record 1: f26f85
{
  "next_table_type": "num_dial_digits",
  "id_of_next_table": "6f85"
}
```

```
// record 2: f0ffc8
{
  "predefined_value1": "0fff",
  "string_table_index1": 200
}
```

DF.EIRENE/EF.Gateway (6F8C) — Access to external networks

Section 8.5

Examples

```
// record 1: f26f85
{
  "next_table_type": "num_dial_digits",
  "id_of_next_table": "6f85"
}
```

```
// record 2: f0ffc8
{
  "predefined_value1": "0fff",
  "string_table_index1": 200
}
```

DF.EIRENE/EF.5to8digits (6F81) — Call Type 2 User Identity Number length

Section 8.6

Examples

```
// file: fffff22
{
  "next_table_type": "empty",
  "id_of_next_table": "ffff",
  "dialed_digits": "22"
}
```

```
// file: f16f8885
{
  "next_table_type": "predefined",
  "id_of_next_table": "6f88",
  "dialed_digits": "58"
}
```

DF.EIRENE/EF.2digits (6F82) — 2 digits input

Section 8.6

Examples

```
// file: fffff22
{
  "next_table_type": "empty",
  "id_of_next_table": "ffff",
  "dialed_digits": "22"
}
```

```
// file: f16f8885
{
  "next_table_type": "predefined",
  "id_of_next_table": "6f88",
  "dialed_digits": "58"
}
```

DF.EIRENE/EF.8digits (6F83) — 8 digits input

Section 8.6

Examples

```
// file: fffff22
{
```

(continues on next page)

(continued from previous page)

```
  "next_table_type": "empty",
  "id_of_next_table": "ffff",
  "dialed_digits": "22"
}
```

```
// file: f16f8885
{
  "next_table_type": "predefined",
  "id_of_next_table": "6f88",
  "dialed_digits": "58"
}
```

DF.EIRENE/EF.9digits (6F84) — 9 digits input

Section 8.6

Examples

```
// file: ffffff22
{
  "next_table_type": "empty",
  "id_of_next_table": "ffff",
  "dialed_digits": "22"
}
```

```
// file: f16f8885
{
  "next_table_type": "predefined",
  "id_of_next_table": "6f88",
  "dialed_digits": "58"
}
```

DF.EIRENE/EF.SSSSS (6F85) — Group call area input

Section 8.6

Examples

```
// file: ffffff22
{
  "next_table_type": "empty",
  "id_of_next_table": "ffff",
  "dialed_digits": "22"
}
```

```
// file: f16f8885
{
  "next_table_type": "predefined",
  "id_of_next_table": "6f88",
  "dialed_digits": "58"
}
```

DF.EIRENE/EF.LLLLL (6F86) — Location number Call Type 6

Section 8.6

Examples

```
// file: fffff22
{
  "next_table_type": "empty",
  "id_of_next_table": "ffff",
  "dialed_digits": "22"
}
```

```
// file: f16f8885
{
  "next_table_type": "predefined",
  "id_of_next_table": "6f88",
  "dialed_digits": "58"
}
```

DF.EIRENE/EF.Location (6F91) — Location number Call Type 7

Section 8.6

Examples

```
// file: fffff22
{
  "next_table_type": "empty",
  "id_of_next_table": "ffff",
  "dialed_digits": "22"
}
```

```
// file: f16f8885
{
  "next_table_type": "predefined",
  "id_of_next_table": "6f88",
  "dialed_digits": "58"
}
```

DF.EIRENE/EF.FreeNumber (6F87) — Free Number Call Type 0 and 8

Section 8.6

Examples

```
// file: fffff22
{
  "next_table_type": "empty",
  "id_of_next_table": "ffff",
  "dialed_digits": "22"
}
```

```
// file: f16f8885
{
```

(continues on next page)

(continued from previous page)


```
  "next_table_type": "predefined",
  "id_of_next_table": "6f88",
  "dialed_digits": "58"
}
```


1.2.7 DF.SYSTEM / sysmocom SJA2+SJA5

DF.SYSTEM/EF.CHV1 (6F01) — EF.CHV1 PIN file

Examples


```
// file: f10303313233346fffffff0a0a3132333435363738
{
  "state": {
    "valid": true,
    "change_able": true,
    "unblock_able": true,
    "disable_able": true,
    "not_initialized": false,
    "disabled": true
  },
  "attempts_remaining": 3,
  "maximum_attempts": 3,
  "pin": "31323334",
  "puk": {
    "attempts_remaining": 10,
    "maximum_attempts": 10,
    "puk": "3132333435363738"
  }
}
```



```
// file: f0030399999999999999
{
  "state": {
    "valid": true,
    "change_able": true,
    "unblock_able": true,
    "disable_able": true,
    "not_initialized": false,
    "disabled": false
  },
  "attempts_remaining": 3,
  "maximum_attempts": 3,
  "pin": "9999999999999999",
  "puk": null
}
```


DF.SYSTEM/EF.CHV2 (6F81) — EF.CHV2 PIN file

Examples

```
// file: f10303313233334fffffff0a0a3132333435363738
{
  "state": {
    "valid": true,
    "change_able": true,
    "unblock_able": true,
    "disable_able": true,
    "not_initialized": false,
    "disabled": true
  },
  "attempts_remaining": 3,
  "maximum_attempts": 3,
  "pin": "31323334",
  "puk": {
    "attempts_remaining": 10,
    "maximum_attempts": 10,
    "puk": "3132333435363738"
  }
}
```

```
// file: f003039999999999999999
{
  "state": {
    "valid": true,
    "change_able": true,
    "unblock_able": true,
    "disable_able": true,
    "not_initialized": false,
    "disabled": false
  },
  "attempts_remaining": 3,
  "maximum_attempts": 3,
  "pin": "9999999999999999",
  "puk": null
}
```

DF.SYSTEM/EF.ADM1 (6F0A) — EF.ADM1 PIN file

Examples

```
// file: f10303313233334fffffff0a0a3132333435363738
{
  "state": {
    "valid": true,
    "change_able": true,
    "unblock_able": true,
    "disable_able": true,
    "not_initialized": false,
    "disabled": true
  },
  "attempts_remaining": 3,
  "maximum_attempts": 3,
```

(continues on next page)

(continued from previous page)


```
  "pin": "31323334",
  "puk": {
    "attempts_remaining": 10,
    "maximum_attempts": 10,
    "puk": "3132333435363738"
  }
}
```



```
// file: f003039999999999999999
{
  "state": {
    "valid": true,
    "change_able": true,
    "unblock_able": true,
    "disable_able": true,
    "not_initialized": false,
    "disabled": false
  },
  "attempts_remaining": 3,
  "maximum_attempts": 3,
  "pin": "9999999999999999",
  "puk": null
}
```


DF.SYSTEM/EF.ADM2 (6F0B) — EF.ADM2 PIN file

Examples


```
// file: f10303313233334ffffffff0a0a3132333435363738
{
  "state": {
    "valid": true,
    "change_able": true,
    "unblock_able": true,
    "disable_able": true,
    "not_initialized": false,
    "disabled": true
  },
  "attempts_remaining": 3,
  "maximum_attempts": 3,
  "pin": "31323334",
  "puk": {
    "attempts_remaining": 10,
    "maximum_attempts": 10,
    "puk": "3132333435363738"
  }
}
```



```
// file: f003039999999999999999
{
  "state": {
```


(continues on next page)

(continued from previous page)


```
    "valid": true,
    "change_able": true,
    "unblock_able": true,
    "disable_able": true,
    "not_initialized": false,
    "disabled": false
  },
  "attempts_remaining": 3,
  "maximum_attempts": 3,
  "pin": "9999999999999999",
  "puk": null
}
```


DF.SYSTEM/EF.ADM3 (6F0C) — EF.ADM3 PIN file

Examples


```
// file: f10303313233334ffffffff0a0a3132333435363738
{
  "state": {
    "valid": true,
    "change_able": true,
    "unblock_able": true,
    "disable_able": true,
    "not_initialized": false,
    "disabled": true
  },
  "attempts_remaining": 3,
  "maximum_attempts": 3,
  "pin": "31323334",
  "puk": {
    "attempts_remaining": 10,
    "maximum_attempts": 10,
    "puk": "3132333435363738"
  }
}
```



```
// file: f0030399999999999999999
{
  "state": {
    "valid": true,
    "change_able": true,
    "unblock_able": true,
    "disable_able": true,
    "not_initialized": false,
    "disabled": false
  },
  "attempts_remaining": 3,
  "maximum_attempts": 3,
  "pin": "9999999999999999",
  "puk": null
}
```

DF.SYSTEM/EF.ADM4 (6F0D) — EF.ADM4 PIN file

Examples

```
// file: f10303313233346fffffff0a0a3132333435363738
{
  "state": {
    "valid": true,
    "change_able": true,
    "unblock_able": true,
    "disable_able": true,
    "not_initialized": false,
    "disabled": true
  },
  "attempts_remaining": 3,
  "maximum_attempts": 3,
  "pin": "31323334",
  "puk": {
    "attempts_remaining": 10,
    "maximum_attempts": 10,
    "puk": "3132333435363738"
  }
}
```

```
// file: f003039999999999999999
{
  "state": {
    "valid": true,
    "change_able": true,
    "unblock_able": true,
    "disable_able": true,
    "not_initialized": false,
    "disabled": false
  },
  "attempts_remaining": 3,
  "maximum_attempts": 3,
  "pin": "9999999999999999",
  "puk": null
}
```

DF.SYSTEM/EF.MILENAGE_CFG (6F21) — Milenage connfiguration

Examples

```
// file:␣
↪400020406000000000000000000000000000000000000000000000000000000000000000000000000000000000000000100000000000000000000000000000000020
{
  "r1": 64,
  "r2": 0,
  "r3": 32,
  "r4": 64,
  "r5": 96,
  "c1": "00000000000000000000000000000000",
  "c2": "00000000000000000000000000000001",
```

(continues on next page)

(continued from previous page)

```
  "c3": "00000000000000000000000000000002",
  "c4": "00000000000000000000000000000004",
  "c5": "00000000000000000000000000000008"
}
```

DF.SYSTEM/EF.0348_KEY (6F22) — TS 03.48 OTA Keys

DF.SYSTEM/EF.SIM_AUTH_COUNTER (AF24) — Number of remaining RUN GSM ALGORITHM executions

DF.SYSTEM/EF.SIM_AUTH_KEY (6F20) — USIM authentication key

Examples

```
// file: 140001020304050607080 90a0b0c0d0e0f101112131415161718191a1b1c1d1e1f
{
  "cfg": {
    "sres_deriv_func": 1,
    "use_opc_instead_of_op": true,
    "algorithm": "milenage"
  },
  "key": "00010203040506070 8090a0b0c0d0e0f",
  "op_opc": "101112131415161718191a1b1c1d1e1f"
}
```

DF.SYSTEM/EF.0348_COUNT (6F23) — TS 03.48 OTA Counters

Examples

```
// file: fe010000000000
{
  "sec_domain": 254,
  "key_set_version": 1,
  "counter": "0000000000"
}
```

DF.SYSTEM/EF.GP_COUNT (6F26) — GP SCP02 Counters

Examples

```
// file: 0070000000
{
  "sec_domain": 0,
  "key_set_version": 112,
  "counter": 0,
  "rfu": 0
}
```

DF.SYSTEM/EF.GP_DIV_DATA (6F27) — GP SCP02 key diversification data

1.3 pySim-trace

pySim-trace is a utility for high-level decode of APDU protocol traces such as those obtained with Osmocom SIMtrace2 or osmo-qcdiag.

pySim-trace leverages the existing knowledge of pySim-shell on anything related to SIM cards, including the structure/encoding of the various files on SIM/USIM/ISIM/HPSIM cards, and applies this to decoding protocol traces. This means that it shows not only the name of the command (like READ BINARY), but actually understands what the currently selected file is, and how to decode the contents of that file.

pySim-trace also understands the parameters passed to commands and how to decode them, for example of the AUTHENTICATE command within the USIM/ISIM/HPSIM application.

1.3.1 Demo

To get an idea how pySim-trace usage looks like, you can watch the relevant part of the 11/2022 SIMtrace2 tutorial whose recording is freely accessible.

1.3.2 Running pySim-trace

Running pySim-trace requires you to specify the *source* of the to-be-decoded APDUs. There are several supported options, each with their own respective parameters (like a file name for PCAP decoding).

See the detailed command line reference below for details.

A typical execution of pySim-trace for doing live decodes of *GSMTAP (SIM APDU)* e.g. from SIMtrace2 or osmo-qcdiag would look like this:

```
./pySim-trace.py gsmtap-udp
```

This binds to the default UDP port 4729 (GSMTAP) on localhost (127.0.0.1), and decodes any APDUs received there.

1.3.3 pySim-trace command line reference

Osmocom pySim high-level SIM card trace decoder

```
usage: pySim-trace.py [-h] [--no-suppress-select] [--no-suppress-status]
                      [--show-raw-apdu]
                      {gsmtap-udp,gsmtap-pyshark-pcap,rspro-pyshark-pcap,rspro-pyshark-live,tca-loader-log,
↪stdin-hex} ...
```

Positional Arguments

source Possible choices: gsmtap-udp, gsmtap-pyshark-pcap, rspro-pyshark-pcap, rspro-pyshark-live, tca-loader-log, stdin-hex

APDU Source

General Options

--no-suppress-select Don't suppress displaying SELECT APDUs. We normally suppress them as they just clutter up the output without giving any useful information. Any subsequent READ/UPDATE/... operations on the selected file will log the file name most recently SELECTed.

Default: True

--no-suppress-status Don't suppress displaying STATUS APDUs. We normally suppress them as they don't provide any information that was not already received in response to the most recent SEELCT.

Default: True

--show-raw-apdu Show the raw APDU in addition to its parsed form.

Default: False

Sub-commands

gsmtap-udp

Read APDUs from live capture by receiving GSMTAP-SIM packets on specified UDP port. Use this for live capture from SIMtrace2 or osmo-qcdiag.

```
pySim-trace.py gsmtap-udp [-h] [-i BIND_IP] [-p BIND_PORT]
```

Named Arguments

-i, --bind-ip Local IP address to which to bind the UDP port

Default: "127.0.0.1"

-p, --bind-port Local UDP port

Default: 4729

gsmtap-pyshark-pcap

Read APDUs from PCAP file containing GSMTAP (SIM APDU) communication; processed via pyshark. Use this if you have recorded a PCAP file containing GSMTAP (SIM APDU) e.g. via tcpdump or wireshark/tshark.

```
pySim-trace.py gsmtap-pyshark-pcap [-h] -f PCAP_FILE
```

Named Arguments

-f, --pcap-file Name of the PCAP[ng] file to be read

rspro-pyshark-pcap

Read APDUs from PCAP file containing RSPRO (osmo-remsim) communication; processed via pyshark. REQUIRES OSMOCOM PATCHED WIRESHARK!

```
pySim-trace.py rspro-pyshark-pcap [-h] -f PCAP_FILE
```

Named Arguments

-f, --pcap-file Name of the PCAP[ng] file to be read

rspro-pyshark-live

Read APDUs from live capture of RSPRO (osmo-remsim) communication; processed via pyshark. REQUIRES OSMOCOM PATCHED WIRESHARK!

```
pySim-trace.py rspro-pyshark-live [-h] -i INTERFACE
```

Named Arguments

-i, --interface Name of the network interface to capture on

tca-loader-log

Read APDUs from a TCA Loader log file.

```
pySim-trace.py tca-loader-log [-h] -f LOG_FILE
```

Named Arguments

-f, --log-file Name of the log file to be read

stdin-hex

Read APDUs as hex-string from stdin.

```
pySim-trace.py stdin-hex [-h]
```

1.3.4 Constraints

- In order to properly track the current location in the filesystem tree and other state, it is important that the trace you're decoding includes all of the communication with the SIM, ideally from the very start (power up).
- pySim-trace currently only supports ETSI UICC (USIM/ISIM/HPSIM) and doesn't yet support legacy GSM SIM. This is not a fundamental technical constraint, it's just simply that nobody got around developing and testing that part. Contributions are most welcome.

1.4 Legacy tools

legacy tools are the classic pySim-prog and pySim-read programs that existed long before pySim-shell.

These days, it is highly recommended to use pySim-shell instead of these legacy tools.

1.4.1 pySim-prog

pySim-prog was the first part of the pySim software suite. It started as a tool to write ICCID, IMSI, MSISDN and Ki to very simplistic SIM cards, and was later extended to a variety of other cards. As the number of features supported became no longer bearable to express with command-line arguments, *pySim-shell* was created.

Basic use cases can still use *pySim-prog*.

Program customizable SIMs

Two modes are possible:

- one where the user specifies every parameter manually:

  This is the most common way to use pySim-prog. The user will specify all relevant parameters directly via the commandline. A typical commandline would look like this:

  pySim-prog.py -p <pcsc_reader> --ki <ki_value> --opc <opc_value> --mcc <mcc_value> --mnc <mnc_value> --country <country_code> --imsi <imsi_value> --iccid <iccid_value> --pin-adm <adm_pin>

  Please note, that this already lengthy commandline still only contains the most common card parameters. For a full list of all possible parameters, use the --help option of pySim-prog. It is also important to mention that not all parameters are supported by all card types. In particular, very simple programmable SIM cards will only support a very basic set of parameters, such as MCC, MNC, IMSI and KI values.

- one where the parameters are generated from a minimal set:

  It is also possible to leave the generation of certain parameters to pySim-prog. This is in particular helpful when a large number of cards should be initialized with randomly generated key material.

  pySim-prog.py -p <pcsc_reader> --mcc <mcc_value> --mnc <mnc_value> --secret <random_secret> --num <card_number> --pin-adm <adm_pin>

  The parameter --secret specifies a random seed that is used to generate the card individual parameters. (IMSI). The secret should contain enough randomness to avoid conflicts. It is also recommended to store the secret safely, in case cards have to be re-generated or the current card batch has to be extended later. For security reasons, the key material, which is also card individual, will not be derived from the random seed. Instead a new random set of Ki and OPc will be generated during each programming cycle. This means fresh keys are generated, even when the --num remains unchanged.

  The parameter --num specifies a card individual number. This number will be managed into the random seed so that it serves as an identifier for a particular set of randomly generated parameters.

  In the example above the parameters --mcc, and --mnc are specified as well, since they identify the GSM network where the cards should operate in, it is absolutely required to keep them static. pySim-prog will use those parameters to generate a valid IMSI that thas the specified MCC/MNC at the beginning and a random tail.

Specifying the card type:

pySim-prog usually autodetects the card type. In case auto detection does not work, it is possible to specify the parameter --type. The following card types are supported:

- Fairwaves-SIM
- fakemagicsim
- gialersim
- grcardsim
- magicsim
- OpenCells-SIM
- supersim
- sysmoISIM-SJA2
- sysmoISIM-SJA5
- sysmosim-gr1
- sysmoSIM-GR2

- sysmoUSIM-SJS1
- Wavemobile-SIM

Specifying the card reader:

It is most common to use pySim-prog together with a PCSC reader. The PCSC reader number is specified via the --pcsc-device or -p option. However, other reader types (such as serial readers and modems) are supported. Use the --help option of pySim-prog for more information.

Card programming using CSV files

To simplify the card programming process, pySim-prog also allows to read the card parameters from a CSV file. When a CSV file is used as input, the user does not have to craft an individual commandline for each card. Instead all card related parameters are automatically drawn from the CSV file.

A CSV files may hold rows for multiple (hundreds or even thousands) of cards. pySim-prog is able to identify the rows either by ICCID (recommended as ICCIDs are normally not changed) or IMSI.

The CSV file format is a flexible format with mandatory and optional columns, here the same rules as for the commandline parameters apply. The column names match the command line options. The CSV file may also contain columns that are unknown to pySim-prog, such as inventory numbers, nicknames or parameters that are unrelated to the card programming process. pySim-prog will silently ignore all unknown columns.

A CSV file may contain the following columns:

- name
- iccid (typically used as key)
- mcc
- mnc
- imsi (may be used as key, but not recommended)
- smsp
- ki
- opc
- acc
- pin_adm, adm1 or pin_adm_hex (must be present)
- msisdn
- epdgid
- epdgSelection
- pcscf
- ims_hdomain
- impi
- impu
- opmode
- fplmn

Due to historical reasons, and to maintain the compatibility between multiple different CSV file formats, the ADM pin may be stored in three different columns. Only one of the three columns must be available.

- adm1: This column contains the ADM pin in numeric ASCII digit format. This format is the most common.

- pin_adm: Same as adm1, only the column name is different
- pin_adm_hex: If the ADM pin consists of raw HEX digits, rather then of numerical ASCII digits, then the ADM pin can also be provided as HEX string using this column.

The following example shows a typical minimal example

```
"imsi","iccid","acc","ki","opc","adm1"
"999700000053010","89882110000000530108","0001","51ACE8BD6313C230F0BFE1A458928DF0",
↪"E5A00E8DE427E21B206526B5D1B902DF","65942330"
"999700000053011","89882110000000530116","0002","746AAFD7F13CFED3AE626B770E53E860",
↪"38F7CE8322D2A7417E0BBD1D7B1190EC","13445792"
"999700123053012","89882110000000530124","0004","D0DA4B7B150026ADC966DC637B26429C",
↪"144FD3AEAC208DFFF4E2140859BAE8EC","53540383"
"999700000053013","89882110000000530132","0008","52E59240ABAC6F53FF5778715C5CE70E",
↪"D9C988550DC70B95F40342298EB84C5E","26151368"
"999700000053014","89882110000000530140","0010","3B4B83CB9C5F3A0B41EBD17E7D96F324",
↪"D61DCC160E3B91F284979552CC5B4D9F","64088605"
"999700000053015","89882110000000530157","0020","D673DAB320D81039B025263610C2BBB3",
↪"4BCE1458936B338067989A06E5327139","94108841"
"999700000053016","89882110000000530165","0040","89DE5ACB76E06D14B0F5D5CD3594E2B1",
↪"411C4B8273FD7607E1885E59F0831906","55184287"
"999700000053017","89882110000000530173","0080","977852F7CEE83233F02E69E211626DE1",
↪"2EC35D48DBF2A99C07D4361F19EF338F","70284674"
```

The following commandline will instruct pySim-prog to use the provided CSV file as parameter source and the ICCID (read from the card before programming) as a key to identify the card. To use the IMSI as a key, the parameter --read-imsi can be used instead of --read-iccid. However, this option is only recommended to be used in very specific corner cases.

pySim-prog.py -p <pcsc_reader> --read-csv <path_to_csv_file> --source csv --read-iccid

It is also possible to pick a row from the CSV file by manually providing an ICCID (option --iccid) or an IMSI (option --imsi) that is then used as a key to find the matching row in the CSV file.

pySim-prog.py -p <pcsc_reader> --read-csv <path_to_csv_file> --source csv --iccid <iccid_value>

Writing CSV files

pySim-prog is also able to generate CSV files that contain a subset of the parameters it has generated or received from some other source (commandline, CSV-File). The generated file will be header-less and contain the following columns:

- name
- iccid
- mcc
- mnc
- imsi
- smsp
- ki
- opc

A commandline that makes use of the CSV write feature would look like this:

pySim-prog.py -p <pcsc_reader> --read-csv <path_to_input_csv_file> --read-iccid --source csv --write-csv <path_to_output_csv_file>

Batch programming

In case larger card batches need to be programmed, it is possible to use the --batch parameter to run pySim-prog in batch mode.

The batch mode will prompt the user to insert a card. Once a card is detected in the reader, the programming is carried out. The user may then remove the card again and the process starts over. This allows for a quick and efficient card programming without permanent commandline interaction.

1.4.2 pySim-read

pySim-read allows to read some of the most important data items from a SIM card. This means it will only read some files of the card, and will only read files accessible to a normal user (without any special authentication)

These days, it is recommended to use the export command of pySim-shell instead. It performs a much more comprehensive export of all of the [standard] files that can be found on the card. To get a human-readable decode instead of the raw hex export, you can use export --json.

Specifically, pySim-read will dump the following:

- MF
- EF.ICCID
- DF.GSM
- EF.IMSI
- EF.GID1
- EF.GID2
- EF.SMSP
- EF.SPN
- EF.PLMNsel
- EF.PLMNwAcT
- EF.OPLMNwAcT
- EF.HPLMNAcT
- EF.ACC
- EF.MSISDN
- EF.AD
- EF.SST
- ADF.USIM
- EF.EHPLMN
- EF.UST
- EF.ePDGId
- EF.ePDGSelection
- ADF.ISIM
- EF.PCSCF
- EF.DOMAIN

- EF.IMPI
- EF.IMPU
- EF.UICCIARI
- EF.IST

pySim-read usage

Legacy tool for reading some parts of a SIM card

```
usage: pySim-read.py [-h] [--verbose] [-d DEV] [-b BAUD] [--pcsc-shared]
                     [-p PCSC | --pcsc-regex REGEX] [--modem-device DEV]
                     [--modem-baud BAUD] [--osmocon PATH] [--apdu-trace]
```

Named Arguments

--verbose Enable verbose logging

Default: False

--apdu-trace Trace the command/response APDUs exchanged with the card

Default: False

Serial Reader

Use a simple/ultra-low-cost serial reader attached to a (physical or USB/virtual) RS232 port. This doesn't work with all RS232-attached smart card readers, only with the very primitive readers following the ancient *Phoenix* or *Smart Mouse* design.

-d, --device Serial Device for SIM access

Default: "/dev/ttyUSB0"

-b, --baud Baud rate used for SIM access

Default: 9600

PC/SC Reader

Use a PC/SC card reader to talk to the SIM card. PC/SC is a standard API for how applications access smart card readers, and is available on a variety of operating systems, such as Microsoft Windows, MacOS X and Linux. Most vendors of smart card readers provide drivers that offer a PC/SC interface, if not even a generic USB CCID driver is used. You can use a tool like `pcsc_scan -r` to obtain a list of readers available on your system.

--pcsc-shared Open PC/SC reaer in SHARED access (default: EXCLUSIVE)

Default: False

-p, --pcsc-device Number of PC/SC reader to use for SIM access

--pcsc-regex Regex matching PC/SC reader to use for SIM access

AT Command Modem Reader

Talk to a SIM Card inside a mobile phone or cellular modem which is attached to this computer and offers an AT command interface including the AT+CSIM interface for Generic SIM access as specified in 3GPP TS 27.007.

--modem-device Serial port of modem for Generic SIM Access (3GPP TS 27.007)

--modem-baud Baud rate used for modem port

Default: 115200

OsmocomBB Reader

Use an OsmocomBB compatible phone to access the SIM inserted to the phone SIM slot. This will require you to run the OsmocomBB firmware inside the phone (can be ram-loaded). It also requires that you run the osmocon program, which provides a unix domain socket to which this reader driver can attach.

--osmocon Socket path for Calypso (e.g. Motorola C1XX) based reader (via OsmocomBB)

1.5 pySim-smpp2sim

This is a program to emulate the entire communication path SMSC-CN-RAN-ME that is usually between an OTA backend and the SIM card. This allows to play with SIM OTA technology without using a mobile network or even a mobile phone.

An external application can act as SMPP ESME and must encode (and encrypt/sign) the OTA SMS and submit them via SMPP to this program, just like it would submit it normally to a SMSC (SMS Service Centre). The program then re-formats the SMPP-SUBMIT into a SMS DELIVER TPDU and passes it via an ENVELOPE APDU to the SIM card that is locally inserted into a smart card reader.

The path from SIM to external OTA application works the opposite way.

The default SMPP system_id is *test*. Likewise, the default SMPP password is *test*

1.5.1 Running pySim-smpp2sim

The command accepts the same command line arguments for smart card interface device selection as pySim-shell, as well as a few SMPP specific arguments:

```
usage: pySim-smpp2sim.py [-h] [-d DEV] [-b BAUD] [--pcsc-shared] [-p PCSC |
                         --pcsc-regex REGEX] [--modem-device DEV]
                         [--modem-baud BAUD] [--osmocon PATH] [--apdu-trace]
                         [--smpp-bind-port SMPP_BIND_PORT]
                         [--smpp-bind-ip SMPP_BIND_IP]
                         [--smpp-system-id SMPP_SYSTEM_ID]
                         [--smpp-password SMPP_PASSWORD]
```

Named Arguments

--apdu-trace Trace the command/response APDUs exchanged with the card

Default: False

Serial Reader

Use a simple/ultra-low-cost serial reader attached to a (physical or USB/virtual) RS232 port. This doesn't work with all RS232-attached smart card readers, only with the very primitive readers following the ancient *Phoenix* or *Smart Mouse* design.

-d, --device Serial Device for SIM access

Default: "/dev/ttyUSB0"

-b, --baud Baud rate used for SIM access

Default: 9600

PC/SC Reader

Use a PC/SC card reader to talk to the SIM card. PC/SC is a standard API for how applications access smart card readers, and is available on a variety of operating systems, such as Microsoft Windows, MacOS X and Linux. Most vendors of smart card readers provide drivers that offer a PC/SC interface, if not even a generic USB CCID driver is used. You can use a tool like `pcsc_scan -r` to obtain a list of readers available on your system.

--pcsc-shared Open PC/SC reaer in SHARED access (default: EXCLUSIVE)

Default: False

-p, --pcsc-device Number of PC/SC reader to use for SIM access

--pcsc-regex Regex matching PC/SC reader to use for SIM access

AT Command Modem Reader

Talk to a SIM Card inside a mobile phone or cellular modem which is attached to this computer and offers an AT command interface including the AT+CSIM interface for Generic SIM access as specified in 3GPP TS 27.007.

--modem-device Serial port of modem for Generic SIM Access (3GPP TS 27.007)

--modem-baud Baud rate used for modem port

Default: 115200

OsmocomBB Reader

Use an OsmocomBB compatible phone to access the SIM inserted to the phone SIM slot. This will require you to run the OsmocomBB firmware inside the phone (can be ram-loaded). It also requires that you run the `osmocon` program, which provides a unix domain socket to which this reader driver can attach.

--osmocon Socket path for Calypso (e.g. Motorola C1XX) based reader (via OsmocomBB)

SMPP Options

--smpp-bind-port TCP Port to bind the SMPP socket to

Default: 2775

--smpp-bind-ip IPv4/IPv6 address to bind the SMPP socket to

Default: "::"

--smpp-system-id SMPP System-ID used by ESME to bind

Default: "test"

--smpp-password SMPP Password used by ESME to bind

Default: "test"

1.5.2 Example execution with sample output

So for a simple system with a single PC/SC device, you would typically use something like *./pySim-smpp2sim.py -p0* to start the program. You will see output like this at start-up

```
Using reader PCSC[HID Global OMNIKEY 3x21 Smart Card Reader [OMNIKEY 3x21 Smart Card␣
↪Reader] 00 00]
INFO     root: Binding Virtual SMSC to TCP Port 2775 at ::
```

The application has hence bound to local TCP port 2775 and expects your SMS-sending applications to send their SMS there. Once you do, you will see log output like below:

```
WARNING  smpp.twisted.protocol: SMPP connection established from ::ffff:127.0.0.1 to port 2775
INFO     smpp.twisted.server: Added CommandId.bind_transceiver bind for 'test'. Active binds:␣
↪CommandId.bind_transceiver: 1, CommandId.bind_transmitter: 0, CommandId.bind_receiver: 0. Max␣
↪binds: 2
INFO     smpp.twisted.protocol: Bind request succeeded for test. 1 active binds
```

And once your external program is sending SMS to the simulated SMSC, it will log something like

```
INFO     root: SMS_DELIVER(MTI=0, MMS=False, LP=False, RP=False, UDHI=True, SRI=False,␣
↪OA=AddressField(TON=international, NPI=unknown, 12), PID=7f, DCS=f6, SCTS=bytearray(b'
↪"pR\x00\x00\x00\x00'), UDL=45, UD=b"\x02p\x00\x00(\x15\x16\x19\x12\x12\xb0\x00\x01'\xfa(\
↪xa5\xba\xc6\x9d<^\x9d\xf2\xc7\x15]\xfd\xdeD\x9c\x82k#b\x15Ve0x{0\xe8\xbe]")
SMSPPDownload(DeviceIdentities({'source_dev_id': 'network', 'dest_dev_id': 'uicc'}),Address({'ton_npi
↪': 0, 'call_number': '0123456'}),SMS_TPDU({'tpdu':
↪'400290217ff6227052000000002d0270000028151619121 2b0000127fa28a5bac69d3c5e9df2c7155dfdde449c826b2362155665307
↪'}))
INFO     root: ENVELOPE:␣
↪d1478202838186040010325 48b3b400290217ff6227052000000002d0270000028151619121 2b0000127fa28a5bac69d3c5e9df2c715
INFO     root: SW 9000:␣
↪027100002412b000019a551bb7c28183652de0ace6170d0e563c5e949a3ba56747fe4c1dbbef16642c
```

Note

for sending OTA SMS messages *smpp-ota-tool* may be used.

1.6 pySim library

1.6.1 pySim filesystem abstraction

Representation of the ISO7816-4 filesystem model.

The File (and its derived classes) represent the structure / hierarchy of the ISO7816-4 smart card file system with the MF, DF, EF and ADF entries, further sub-divided into the EF sub-types Transparent, Linear Fixed, etc.

The classes are intended to represent the *specification* of the filesystem, not the actual contents / runtime state of interacting with a given smart card.

class pySim.filesystem.BerTlvEF(*fid: str*, *sfid: str = None*, *name: str = None*, *desc: str = None*, *parent:* CardDF *= None*, *size: Tuple[int, int | None] = (1, None)*, ***kwargs*)

BER-TLV EF (Entry File) in the smart card filesystem. A BER-TLV EF is a binary file with a BER (Basic Encoding Rules) TLV structure

NOTE: We currently don't really support those, this class is simply a wrapper around TransparentEF as a placeholder, so we can already define EFs of BER-TLV type without fully supporting them.

Parameters

- fid – File Identifier (4 hex digits)
- sfid – Short File Identifier (2 hex digits, optional)
- name – Brief name of the file, like EF_ICCID
- desc – Description of the file
- parent – Parent CardFile object within filesystem hierarchy
- size – tuple of (minimum_size, recommended_size)

class ShellCommands

Shell commands specific for BER-TLV EFs.

Private reference to the CLI instance in which this CommandSet running.

This will be set when the CommandSet is registered and it should be accessed by child classes using the self._cmd property.

do_delete_all(*opts*)

Delete all data from a BER-TLV EF

do_delete_data(*opts*)

Delete data for a given tag in a BER-TLV EF

do_retrieve_data(*opts*)

Retrieve (Read) data from a BER-TLV EF

do_retrieve_tags(*_opts*)

List tags available in a given BER-TLV EF

do_set_data(*opts*)

Set (Write) data for a given tag in a BER-TLV EF

static export(*as_json: bool*, *lchan*)

Export the file contents of a BerTlvEF. This method returns a shell command string (See also ShellCommand definition in this class) that can be used to write the file contents back.

class pySim.filesystem.CardADF(*aid: str*, *has_fs: bool = False*, ***kwargs*)

ADF (Application Dedicated File) in the smart card filesystem

Parameters

- fid – File Identifier (4 hex digits)
- sfid – Short File Identifier (2 hex digits, optional)
- name – Brief name of the file, like EF_ICCID
- desc – Description of the file
- parent – Parent CardFile object within filesystem hierarchy
- profile – Card profile that this file should be part of
- service – Service (SST/UST/IST) associated with the file

static export(*as_json: bool*, *lchan*)

Export application specific parameters that are not part of the UICC filesystem.

class pySim.filesystem.CardApplication(*name*, *adf: CardADF | None = None*, *aid: str = None*, *sw: dict = None*)

A card application is represented by an ADF (with contained hierarchy) and optionally some SW definitions.

Parameters

- adf – ADF name
- sw – Dict of status word conversions

static export(*as_json: bool*, *lchan*)

Export application specific parameters, in the form of commandline script. (see also comment in the export method of class "CardFile")

interpret_sw(*sw*)

Interpret a given status word within the application.

Parameters
sw – Status word as string of 4 hex digits

Returns
Tuple of two strings

class pySim.filesystem.CardDF(***kwargs*)

DF (Dedicated File) in the smart card filesystem. Those are basically sub-directories.

Parameters

- fid – File Identifier (4 hex digits)
- sfid – Short File Identifier (2 hex digits, optional)
- name – Brief name of the file, like EF_ICCID
- desc – Description of the file
- parent – Parent CardFile object within filesystem hierarchy
- profile – Card profile that this file should be part of
- service – Service (SST/UST/IST) associated with the file

class ShellCommands

Private reference to the CLI instance in which this CommandSet running.

This will be set when the CommandSet is registered and it should be accessed by child classes using the self._cmd property.

add_file(*child: CardFile*, *ignore_existing: bool = False*)

Add a child (DF/EF) to this DF. :param child: The new DF/EF to be added :param ignore_existing: Ignore, if file with given FID already exists. Old one will be kept.

add_files(*children: Iterable[CardFile]*, *ignore_existing: bool = False*)

Add a list of child (DF/EF) to this DF

Parameters

- children – List of new DF/EFs to be added
- ignore_existing – Ignore, if file[s] with given FID already exists. Old one[s] will be kept.

get_selectables(*flags=[]*) → dict

Return a dict of {'identifier': File} that is selectable from the current DF.

Parameters
flags – Specify which selectables to return 'FIDS' and/or 'NAMES'; If not specified, all selectables will be returned.

Returns
dict containing all selectable items. Key is identifier (string), value a reference to a CardFile (or derived class) instance.

lookup_file_by_fid(*fid: str*) → *CardFile* | None

Find a file with given file ID within current DF.

lookup_file_by_name(*name: str | None*) → *CardFile* | None

Find a file with given name within current DF.

lookup_file_by_sfid(*sfid: str | None*) → *CardFile* | None

Find a file with given short file ID within current DF.

class pySim.filesystem.CardEF(** (Keyword-only parameters separator (PEP 3102))*, *fid*, ***kwargs*)

EF (Entry File) in the smart card filesystem

Parameters

- fid – File Identifier (4 hex digits)
- sfid – Short File Identifier (2 hex digits, optional)
- name – Brief name of the file, like EF_ICCID
- desc – Description of the file
- parent – Parent CardFile object within filesystem hierarchy
- profile – Card profile that this file should be part of
- service – Service (SST/UST/IST) associated with the file

get_selectables(*flags=[]*) → dict

Return a dict of {'identifier': File} that is selectable from the current DF.

Parameters
flags – Specify which selectables to return 'FIDS' and/or 'NAMES'; If not specified, all selectables will be returned.

Returns
dict containing all selectable items. Key is identifier (string), value a reference to a CardFile (or derived class) instance.

class pySim.filesystem.CardFile(*fid: str = None*, *sfid: str = None*, *name: str = None*, *desc: str = None*, *parent: CardDF | None = None*, *profile: CardProfile | None = None*, *service: int | List[int] | Tuple[int, ...] | None = None*)

Base class for all objects in the smart card filesystem. Serve as a common ancestor to all other file types; rarely used directly.

Parameters

- fid – File Identifier (4 hex digits)
- sfid – Short File Identifier (2 hex digits, optional)
- name – Brief name of the file, like EF_ICCID

- desc – Description of the file
- parent – Parent CardFile object within filesystem hierarchy
- profile – Card profile that this file should be part of
- service – Service (SST/UST/IST) associated with the file

build_select_path_to(*target: CardFile*) → List[*CardFile*] | None

Build the relative sequence of files we need to traverse to get from us to 'target'.

decode_select_response(*data_hex: str*)

Decode the response to a SELECT command.

Parameters
data_hex – Hex string of the select response

static export(*as_json: bool*, *lchan*)

Export file contents in the form of commandline script. This method is meant to be overloaded by a subclass in case any exportable contents are present. The generated script may contain multiple command lines separated by line breaks ("n"), where the last commandline shall have no line break at the end (e.g. "update_record 1 112233nupdate_record 1 445566"). Naturally this export method will always refer to the currently selected file of the presented lchan.

fully_qualified_path(*prefer_name: bool = True*) → List[str]

Return fully qualified path to file as list of FID or name strings.

Parameters
prefer_name – Preferably build path of names; fall-back to FIDs as required

fully_qualified_path_fobj() → List[*CardFile*]

Return fully qualified path to file as list of CardFile instance references.

fully_qualified_path_str(*prefer_name: bool = True*) → str

Return fully qualified path to file as string.

Parameters
prefer_name – Preferably build path of names; fall-back to FIDs as required

get_mf() → *CardMF* | None

Return the MF (root) of the file system.

get_profile()

Get the profile associated with this file. If this file does not have any profile assigned, try to find a file above (usually the MF) in the filesystem hierarchy that has a profile assigned

get_selectable_names(*flags=[]*) → List[str]

Return a dict of {'identifier': File} that is selectable from the current file.

Parameters
flags – Specify which selectables to return 'FIDS' and/or 'NAMES'; If not specified, all selectables will be returned.

Returns
list containing all selectable names.

get_selectables(*flags=[]*) → Dict[str, *CardFile*]

Return a dict of {'identifier': File} that is selectable from the current file.

Parameters
flags – Specify which selectables to return 'FIDS' and/or 'NAMES'; If not specified, all selectables will be returned.

Returns
dict containing all selectable items. Key is identifier (string), value a reference to a CardFile (or derived class) instance.

should_exist_for_services(*services: List[int]*)

Assuming the provided list of activated services, should this file exist and be activated?.

class pySim.filesystem.CardMF(***kwargs*)

MF (Master File) in the smart card filesystem

Parameters

- fid – File Identifier (4 hex digits)
- sfid – Short File Identifier (2 hex digits, optional)
- name – Brief name of the file, like EF_ICCID
- desc – Description of the file
- parent – Parent CardFile object within filesystem hierarchy
- profile – Card profile that this file should be part of
- service – Service (SST/UST/IST) associated with the file

add_application_df(*app: CardADF*)

Add an Application to the MF

decode_select_response(*data_hex: str | None*) → object

Decode the response to a SELECT command.

This is the fall-back method which automatically defers to the standard decoding method defined by the card profile. When no profile is set, then no decoding is performed. Specific derived classes (usually ADF) can overload this method to install specific decoding.

get_app_names()

Get list of completions (AID names)

get_app_selectables(*flags=[]*) → dict

Get applications by AID + name

get_selectables(*flags=[]*) → dict

Return a dict of {'identifier': File} that is selectable from the current DF.

Parameters
flags – Specify which selectables to return 'FIDS' and/or 'NAMES'; If not specified, all selectables will be returned.

Returns
dict containing all selectable items. Key is identifier (string), value a reference to a CardFile (or derived class) instance.

class pySim.filesystem.CardModel

A specific card model, typically having some additional vendor-specific files. All you need to do is to define a sub-class with a list of ATRs or an overridden match method.

abstractmethod classmethod add_files(*rs: RuntimeState*)

Add model specific files to given RuntimeState.

static apply_matching_models(*scc:* SimCardCommands, *rs: RuntimeState*)

Check if any of the CardModel sub-classes 'match' the currently inserted card (by ATR or overriding the 'match' method). If so, call their 'add_files' method.

classmethod match(*scc:* SimCardCommands) → bool

Test if given card matches this model.

class pySim.filesystem.CyclicEF(*fid: str*, *sfid: str = None*, *name: str = None*, *desc: str = None*, *parent:* CardDF *= None*, *rec_len: Tuple[int, int | None] = (1, None)*, ***kwargs*)

Cyclic EF (Entry File) in the smart card filesystem

Parameters

- fid – File Identifier (4 hex digits)
- sfid – Short File Identifier (2 hex digits, optional)
- name – Brief name of the file, like EF_ICCID
- desc – Description of the file
- parent – Parent CardFile object within filesystem hierarchy
- rec_len – Tuple of (minimum_length, recommended_length)
- leftpad – On write, data must be padded from the left to fit physical record length

class pySim.filesystem.JsonEditor(*cmd*, *orig_json*, *ef*)

Context manager for editing a JSON-encoded EF value in an external editor.

Writes the current JSON value (plus encode/decode examples as //-comments) to a temporary file, opens the user's editor, then reads the result back (stripping comment lines) and returns it as the context variable:

```
with JsonEditor(self._cmd, orig_json, ef) as edited_json:
    if edited_json != orig_json:
        ...write back...
```

class pySim.filesystem.LinFixedEF(*fid: str*, *sfid: str = None*, *name: str = None*, *desc: str = None*, *parent:* CardDF *| None = None*, *rec_len: Tuple[int, int | None] = (1, None)*, *leftpad: bool = False*, ***kwargs*)

Linear Fixed EF (Entry File) in the smart card filesystem.

Linear Fixed EFs are record oriented files. They consist of a number of fixed-size records. The records can be individually read/updated.

Parameters

- fid – File Identifier (4 hex digits)
- sfid – Short File Identifier (2 hex digits, optional)
- name – Brief name of the file, like EF_ICCID
- desc – Description of the file
- parent – Parent CardFile object within filesystem hierarchy
- rec_len – Tuple of (minimum_length, recommended_length)
- leftpad – On write, data must be padded from the left to fit physical record length

class ShellCommands

Shell commands specific for Linear Fixed EFs.

Private reference to the CLI instance in which this CommandSet running.

This will be set when the CommandSet is registered and it should be accessed by child classes using the self._cmd property.

do_decode_hex(*opts*)

Decode command-line provided hex-string as if it was read from the file.

do_edit_record_decoded(*opts*)

Edit the JSON representation of one record in an editor.

do_read_record(*opts*)

Read one or multiple records from a record-oriented EF

do_read_record_decoded(*opts*)

Read + decode a record from a record-oriented EF

do_read_records(*_opts*)

Read all records from a record-oriented EF

do_read_records_decoded(*opts*)

Read + decode all records from a record-oriented EF

do_update_record(*opts*)

Update (write) data to a record-oriented EF

do_update_record_decoded(*opts*)

Encode + Update (write) data to a record-oriented EF

decode_record_bin(*raw_bin_data: bytearray*, *record_nr: int*) → dict

Decode raw (binary) data into abstract representation.

A derived class would typically provide a _decode_record_bin() or _decode_record_hex() method for implementing this specifically for the given file. This function checks which of the method exists, add calls them (with conversion, as needed).

Parameters

- raw_bin_data – binary encoded data
- record_nr – record number (1 for first record, ...)

Returns

abstract_data; dict representing the decoded data

decode_record_hex(*raw_hex_data: str*, *record_nr: int = 1*) → dict

Decode raw (hex string) data into abstract representation.

A derived class would typically provide a _decode_record_bin() or _decode_record_hex() method for implementing this specifically for the given file. This function checks which of the method exists, add calls them (with conversion, as needed).

Parameters

- raw_hex_data – hex-encoded data
- record_nr – record number (1 for first record, ...)

Returns

abstract_data; dict representing the decoded data

encode_record_bin(*abstract_data: dict*, *record_nr: int*, *total_len: int | None = None*) → bytearray

Encode abstract representation into raw (binary) data.

A derived class would typically provide an _encode_record_bin() or _encode_record_hex() method for implementing this specifically for the given file. This function checks which of the method exists, add calls them (with conversion, as needed).

Parameters

- abstract_data – dict representing the decoded data
- record_nr – record number (1 for first record, ...)
- total_len – expected total length of the encoded data (record length)

Returns
binary encoded data

encode_record_hex(*abstract_data: dict*, *record_nr: int*, *total_len: int | None = None*) → str

Encode abstract representation into raw (hex string) data.

A derived class would typically provide an _encode_record_bin() or _encode_record_hex() method for implementing this specifically for the given file. This function checks which of the method exists, add calls them (with conversion, as needed).

Parameters

- abstract_data – dict representing the decoded data
- record_nr – record number (1 for first record, ...)
- total_len – expected total length of the encoded data (record length)

Returns
hex string encoded data

static export(*as_json: bool*, *lchan*)

Export the file contents of a LinFixedEF (or a CyclicEF). This method returns a shell command string (See also ShellCommand definition in this class) that can be used to write the file contents back.

class pySim.filesystem.Path(*p: str | List[str] | List[int]*)

Representation of a file-system path.

is_parent(*other:* Path) → bool

Is this instance a parent of the given other instance?

relative_to_mf() → *Path*

Return a path relative to MF, i.e. without initial explicit MF.

class pySim.filesystem.TransRecEF(*fid: str*, *rec_len: int*, *sfid: str = None*, *name: str = None*, *desc: str = None*, *parent:* CardDF *| None = None*, *size: Tuple[int, int | None] = (1, None)*, ***kwargs*)

Transparent EF (Entry File) containing fixed-size records.

These are the real odd-balls and mostly look like mistakes in the specification: Specified as 'transparent' EF, but actually containing several fixed-length records inside. We add a special class for those, so the user only has to provide encoder/decoder functions for a record, while this class takes care of split / merge of records.

Parameters

- fid – File Identifier (4 hex digits)
- sfid – Short File Identifier (2 hex digits, optional)

- name – Brief name of the file, like EF_ICCID
- desc – Description of the file
- parent – Parent CardFile object within filesystem hierarchy
- rec_len – Length of the fixed-length records within transparent EF
- size – tuple of (minimum_size, recommended_size)

decode_record_bin(*raw_bin_data: bytearray*) → dict

Decode raw (binary) data into abstract representation.

A derived class would typically provide a _decode_record_bin() or _decode_record_hex() method for implementing this specifically for the given file. This function checks which of the method exists, add calls them (with conversion, as needed).

Parameters
raw_bin_data – binary encoded data

Returns
abstract_data; dict representing the decoded data

decode_record_hex(*raw_hex_data: str*) → dict

Decode raw (hex string) data into abstract representation.

A derived class would typically provide a _decode_record_bin() or _decode_record_hex() method for implementing this specifically for the given file. This function checks which of the method exists, add calls them (with conversion, as needed).

Parameters
raw_hex_data – hex-encoded data

Returns
abstract_data; dict representing the decoded data

encode_record_bin(*abstract_data: dict*, *total_len: int | None = None*) → bytearray

Encode abstract representation into raw (binary) data.

A derived class would typically provide an _encode_record_bin() or _encode_record_hex() method for implementing this specifically for the given file. This function checks which of the method exists, add calls them (with conversion, as needed).

Parameters

- abstract_data – dict representing the decoded data
- total_len – expected total length of the encoded data (record length)

Returns
binary encoded data

encode_record_hex(*abstract_data: dict*, *total_len: int | None = None*) → str

Encode abstract representation into raw (hex string) data.

A derived class would typically provide an _encode_record_bin() or _encode_record_hex() method for implementing this specifically for the given file. This function checks which of the method exists, add calls them (with conversion, as needed).

Parameters

- abstract_data – dict representing the decoded data
- total_len – expected total length of the encoded data (record length)

Returns
hex string encoded data

class pySim.filesystem.TransparentEF(*fid: str*, *sfid: str = None*, *name: str = None*, *desc: str = None*, *parent:* CardDF *= None*, *size: Tuple[int, int | None] = (1, None)*, ***kwargs*)

Transparent EF (Entry File) in the smart card filesystem.

A Transparent EF is a binary file with no formal structure. This is contrary to Record based EFs which have [fixed size] records that can be individually read/updated.

Parameters

- fid – File Identifier (4 hex digits)
- sfid – Short File Identifier (2 hex digits, optional)
- name – Brief name of the file, like EF_ICCID
- desc – Description of the file
- parent – Parent CardFile object within filesystem hierarchy
- size – tuple of (minimum_size, recommended_size)

class ShellCommands

Shell commands specific for transparent EFs.

Private reference to the CLI instance in which this CommandSet running.

This will be set when the CommandSet is registered and it should be accessed by child classes using the self._cmd property.

do_decode_hex(*opts*)

Decode command-line provided hex-string as if it was read from the file.

do_edit_binary_decoded(*_opts*)

Edit the JSON representation of the EF contents in an editor.

do_read_binary(*opts*)

Read binary data from a transparent EF

do_read_binary_decoded(*opts*)

Read + decode data from a transparent EF

do_update_binary(*opts*)

Update (Write) data of a transparent EF

do_update_binary_decoded(*opts*)

Encode + Update (Write) data of a transparent EF

decode_bin(*raw_bin_data: bytearray*) → dict

Decode raw (binary) data into abstract representation.

A derived class would typically provide a _decode_bin() or _decode_hex() method for implementing this specifically for the given file. This function checks which of the method exists, add calls them (with conversion, as needed).

Parameters
raw_bin_data – binary encoded data

Returns
abstract_data; dict representing the decoded data

decode_hex(*raw_hex_data: str*) → dict

Decode raw (hex string) data into abstract representation.

A derived class would typically provide a _decode_bin() or _decode_hex() method for implementing this specifically for the given file. This function checks which of the method exists, add calls them (with conversion, as needed).

Parameters
raw_hex_data – hex-encoded data

Returns
abstract_data; dict representing the decoded data

encode_bin(*abstract_data: dict*, *total_len: int | None = None*) → bytearray

Encode abstract representation into raw (binary) data.

A derived class would typically provide an _encode_bin() or _encode_hex() method for implementing this specifically for the given file. This function checks which of the method exists, add calls them (with conversion, as needed).

Parameters

- abstract_data – dict representing the decoded data
- total_len – expected total length of the encoded data (file size)

Returns
binary encoded data

encode_hex(*abstract_data: dict*, *total_len: int | None = None*) → str

Encode abstract representation into raw (hex string) data.

A derived class would typically provide an _encode_bin() or _encode_hex() method for implementing this specifically for the given file. This function checks which of the method exists, add calls them (with conversion, as needed).

Parameters

- abstract_data – dict representing the decoded data
- total_len – expected total length of the encoded data (file size)

Returns
hex string encoded data

static export(*as_json: bool*, *lchan*)

Export the file contents of a TransparentEF. This method returns a shell command string (See also ShellCommand definition in this class) that can be used to write the file contents back.

pySim.filesystem.interpret_sw(*sw_data: dict*, *sw: str*)

Interpret a given status word.

Parameters

- sw_data – Hierarchical dict of status word matches
- sw – status word to match (string of 4 hex digits)

Returns
tuple of two strings (class_string, description)

1.6.2 pySim commands abstraction

pySim: SIM Card commands according to ISO 7816-4 and TS 11.11

class pySim.commands.SimCardCommands(*transport: LinkBase*, *lchan_nr: int = 0*)

Class providing methods for various card-specific commands such as SELECT, READ BINARY, etc. Historically one instance exists below CardBase, but with the introduction of multiple logical channels there can be multiple instances. The lchan number will then be patched into the CLA byte by the respective instance.

activate_file(*fid: Hexstr*) → Tuple[Hexstr, SwHexstr]

Execute ACTIVATE FILE command as per TS 102 221 Section 11.1.15.

Parameters
fid – file identifier as hex string

authenticate(*rand: Hexstr*, *autn: Hexstr*, *context: str = '3g'*) → Tuple[Hexstr, SwHexstr]

Execute AUTHENTICATE (USIM/ISIM).

Parameters

- rand – 16 byte random data as hex string (RAND)
- autn – 8 byte Authentication Token (AUTN)
- context – 16 byte random data ('3g' or 'gsm')

binary_size(*ef: Hexstr | List[Hexstr]*) → int

Determine the size of given transparent file.

Parameters
ef – string or list of strings indicating name or path of transparent EF

change_chv(*chv_no: int*, *pin_code: Hexstr*, *new_pin_code: Hexstr*) → Tuple[Hexstr, SwHexstr]

Change a given CHV (Card Holder Verification == PIN)

Parameters

- chv_no – chv number (1=CHV1, 2=CHV2, ...)
- pin_code – current chv code as hex string
- new_pin_code – new chv code as hex string

create_file(*payload: Hexstr*) → Tuple[Hexstr, SwHexstr]

Execute CREATE FILE command as per TS 102 222 Section 6.3

deactivate_file() → Tuple[Hexstr, SwHexstr]

Execute DECATIVATE FILE command as per TS 102 221 Section 11.1.14.

delete_file(*fid: Hexstr*) → Tuple[Hexstr, SwHexstr]

Execute DELETE FILE command as per TS 102 222 Section 6.4

disable_chv(*chv_no: int*, *pin_code: Hexstr*) → Tuple[Hexstr, SwHexstr]

Disable a given CHV (Card Holder Verification == PIN)

Parameters

- chv_no – chv number (1=CHV1, 2=CHV2, ...)
- pin_code – current chv code as hex string
- new_pin_code – new chv code as hex string

enable_chv(*chv_no: int*, *pin_code: Hexstr*) → Tuple[Hexstr, SwHexstr]

Enable a given CHV (Card Holder Verification == PIN)

Parameters

- chv_no – chv number (1=CHV1, 2=CHV2, ...)
- pin_code – chv code as hex string

envelope(*payload: Hexstr*) → Tuple[Hexstr, SwHexstr]

Send one ENVELOPE command to the SIM

Parameters

payload – payload as hex string

fork_lchan(*lchan_nr: int*) → *SimCardCommands*

Fork a per-lchan specific SimCardCommands instance off the current instance.

get_atr() → Hexstr

Return the ATR of the currently inserted card.

manage_channel(*mode: str = 'open'*, *lchan_nr: int = 0*) → Tuple[Hexstr, SwHexstr]

Execute MANAGE CHANNEL command as per TS 102 221 Section 11.1.17.

Parameters

- mode – logical channel operation code ('open' or 'close')
- lchan_nr – logical channel number (1-19, 0=assigned by UICC)

property max_cmd_len: int

Maximum length of the command apdu data section. Depends on secure channel protocol used.

read_binary(*ef: Hexstr | List[Hexstr]*, *length: int = None*, *offset: int = 0*) → Tuple[Hexstr, SwHexstr]

Execute READD BINARY.

Parameters

- ef – string or list of strings indicating name or path of transparent EF
- length – number of bytes to read
- offset – byte offset in file from which to start reading

read_record(*ef: Hexstr | List[Hexstr]*, *rec_no: int*) → Tuple[Hexstr, SwHexstr]

Execute READ RECORD.

Parameters

- ef – string or list of strings indicating name or path of linear fixed EF
- rec_no – record number to read

record_count(*ef: Hexstr | List[Hexstr]*) → int

Determine the number of records in given file.

Parameters

ef – string or list of strings indicating name or path of linear fixed EF

record_size(*ef: Hexstr | List[Hexstr]*) → int

Determine the record size of given file.

Parameters

ef – string or list of strings indicating name or path of linear fixed EF

reset_card() → Hexstr

Physically reset the card

resize_file(*payload: Hexstr*) → Tuple[Hexstr, SwHexstr]

Execute RESIZE FILE command as per TS 102 222 Section 6.10

resume_uicc(*token: Hexstr*) → Tuple[Hexstr, SwHexstr]

Send SUSPEND UICC (resume) to the card.

retrieve_data(*ef: Hexstr | List[Hexstr]*, *tag: int*) → Tuple[Hexstr, SwHexstr]

Execute RETRIEVE DATA, see also TS 102 221 Section 11.3.1.

Args

ef : string or list of strings indicating name or path of transparent EF tag : BER-TLV Tag of value to be retrieved

run_gsm(*rand: Hexstr*) → Tuple[Hexstr, SwHexstr]

Execute RUN GSM ALGORITHM.

Parameters

rand – 16 byte random data as hex string (RAND)

select_adf(*aid: Hexstr*) → Tuple[Hexstr, SwHexstr]

Execute SELECT a given Application ADF.

Parameters

aid – application identifier as hex string

select_file(*fid: Hexstr*) → Tuple[Hexstr, SwHexstr]

Execute SELECT a given file by FID.

Parameters

fid – file identifier as hex string

select_parent_df() → Tuple[Hexstr, SwHexstr]

Execute SELECT to switch to the parent DF

select_path(*dir_list: Hexstr | List[Hexstr]*) → List[Hexstr]

Execute SELECT for an entire list/path of FIDs.

Parameters

dir_list – list of FIDs representing the path to select

Returns

list of return values (FCP in hex encoding) for each element of the path

send_apdu(*pdu: Hexstr*, *apply_lchan: bool = True*) → Tuple[Hexstr, SwHexstr]

Sends an APDU and auto fetch response data

Parameters

- pdu – string of hexadecimal characters (ex. “A0A40000023F00”)
- apply_lchan – apply the currently selected lchan to the CLA byte before sending

Returns

tuple(data, sw), where

data : string (in hex) of returned data (ex. “074F4EFFFF”) sw : string (in hex) of status word (ex. “9000”)

send_apdu_checksw(*pdu: Hexstr*, *sw: SwMatchstr = '9000'*, *apply_lchan: bool = True*) → Tuple[Hexstr, SwHexstr]

Sends an APDU and check returned SW

Parameters

- pdu – string of hexadecimal characters (ex. "A0A40000023F00")
- sw – string of 4 hexadecimal characters (ex. "9000"). The user may mask out certain digits using a '?' to add some ambiguity if needed.
- apply_lchan – apply the currently selected lchan to the CLA byte before sending

Returns

tuple(data, sw), where
data : string (in hex) of returned data (ex. "074F4EFFFF") sw : string (in hex) of status word (ex. "9000")

send_apdu_constr(*cla: Hexstr*, *ins: Hexstr*, *p1: Hexstr*, *p2: Hexstr*, *cmd_constr: Construct*, *cmd_data: Hexstr*, *resp_constr: Construct*, *apply_lchan: bool = True*) → Tuple[dict, SwHexstr]

Build and sends an APDU using a 'construct' definition; parses response.

Parameters

- cla – string (in hex) ISO 7816 class byte
- ins – string (in hex) ISO 7816 instruction byte
- p1 – string (in hex) ISO 7116 Parameter 1 byte
- p2 – string (in hex) ISO 7116 Parameter 2 byte
- cmd_cosntr – defining how to generate binary APDU command data
- cmd_data – command data passed to cmd_constr
- resp_cosntr – defining how to decode binary APDU response data
- apply_lchan – apply the currently selected lchan to the CLA byte before sending

Returns
Tuple of (decoded_data, sw)

send_apdu_constr_checksw(*cla: Hexstr*, *ins: Hexstr*, *p1: Hexstr*, *p2: Hexstr*, *cmd_constr: Construct*, *cmd_data: Hexstr*, *resp_constr: Construct*, *sw_exp: SwMatchstr = '9000'*, *apply_lchan: bool = True*) → Tuple[dict, SwHexstr]

Build and sends an APDU using a 'construct' definition; parses response.

Parameters

- cla – string (in hex) ISO 7816 class byte
- ins – string (in hex) ISO 7816 instruction byte
- p1 – string (in hex) ISO 7116 Parameter 1 byte
- p2 – string (in hex) ISO 7116 Parameter 2 byte
- cmd_cosntr – defining how to generate binary APDU command data
- cmd_data – command data passed to cmd_constr
- resp_cosntr – defining how to decode binary APDU response data
- exp_sw – string (in hex) of status word (ex. "9000")

Returns
Tuple of (decoded_data, sw)

set_data(*ef*, *tag: int*, *value: str*, *verify: bool = False*, *conserve: bool = False*) → Tuple[Hexstr, SwHexstr]

Execute SET DATA.

Args
ef : string or list of strings indicating name or path of transparent EF tag : BER-TLV Tag of value to be stored value : BER-TLV value to be stored

status() → Tuple[Hexstr, SwHexstr]

Execute a STATUS command as per TS 102 221 Section 11.1.2.

suspend_uicc(*min_len_secs: int = 60*, *max_len_secs: int = 43200*) → Tuple[int, Hexstr, SwHexstr]

Send SUSPEND UICC to the card.

Parameters
- min_len_secs – mimumum suspend time seconds
- max_len_secs – maximum suspend time seconds

terminal_profile(*payload: Hexstr*) → Tuple[Hexstr, SwHexstr]

Send TERMINAL PROFILE to card

Parameters
payload – payload as hex string

terminate_card_usage() → Tuple[Hexstr, SwHexstr]

Execute TERMINATE CARD USAGE command as per TS 102 222 Section 6.9

terminate_df(*fid: Hexstr*) → Tuple[Hexstr, SwHexstr]

Execute TERMINATE DF command as per TS 102 222 Section 6.7

terminate_ef(*fid: Hexstr*) → Tuple[Hexstr, SwHexstr]

Execute TERMINATE EF command as per TS 102 222 Section 6.8

try_select_path(*dir_list: List[Hexstr]*) → List[Tuple[Hexstr, SwHexstr]]

Try to select a specified path

Parameters
dir_list – list of hex-string FIDs

unblock_chv(*chv_no: int*, *puk_code: str*, *pin_code: str*)

Unblock a given CHV (Card Holder Verification == PIN)

Parameters
- chv_no – chv number (1=CHV1, 2=CHV2, ...)
- puk_code – puk code as hex string
- pin_code – new chv code as hex string

update_binary(*ef: Hexstr | List[Hexstr]*, *data: Hexstr*, *offset: int = 0*, *verify: bool = False*, *conserve: bool = False*) → Tuple[Hexstr, SwHexstr]

Execute UPDATE BINARY.

Parameters
- ef – string or list of strings indicating name or path of transparent EF
- data – hex string of data to be written

- offset – byte offset in file from which to start writing
- verify – Whether or not to verify data after write

update_record(*ef: Hexstr | List[Hexstr]*, *rec_no: int*, *data: Hexstr*, *force_len: bool = False*, *verify: bool = False*, *conserve: bool = False*, *leftpad: bool = False*) → Tuple[Hexstr, SwHexstr]

Execute UPDATE RECORD.

Parameters

- ef – string or list of strings indicating name or path of linear fixed EF
- rec_no – record number to read
- data – hex string of data to be written
- force_len – enforce record length by using the actual data length
- verify – verify data by re-reading the record
- conserve – read record and compare it with data, skip write on match
- leftpad – apply 0xff padding from the left instead from the right side.

verify_chv(*chv_no: int*, *code: Hexstr*) → Tuple[Hexstr, SwHexstr]

Verify a given CHV (Card Holder Verification == PIN)

Parameters

- chv_no – chv number (1=CHV1, 2=CHV2, ...)
- code – chv code as hex string

pySim.commands.cla_with_lchan(*cla_byte: Hexstr*, *lchan_nr: int*) → Hexstr

Embed a logical channel number into the hex-string encoded CLA value.

pySim.commands.lchan_nr_to_cla(*cla: int*, *lchan_nr: int*) → int

Embed a logical channel number into the CLA byte.

1.6.3 pySim Transport

The pySim.transport classes implement specific ways how to communicate with a SIM card. A "transport" provides ways to transceive APDUs with the card.

The most commonly used transport uses the PC/SC interface to utilize a variety of smart card interfaces ("readers").

Transport base class

pySim: PCSC reader transport link base

class pySim.transport.LinkBase(*sw_interpreter=None*, *apdu_tracer: ApduTracer | None = None*, *proactive_handler:* ProactiveHandler *| None = None*)

Base class for link/transport to card.

abstractmethod connect()

Connect to a card immediately

abstractmethod disconnect()

Disconnect from card

abstractmethod get_atr() → Hexstr

Retrieve card ATR

reset_card()

Resets the card (power down/up)

send_apdu(*apdu: Hexstr*) → Tuple[Hexstr, SwHexstr]

Sends an APDU with minimal processing

Parameters

apdu – string of hexadecimal characters (ex. "A0A40000023F00", must comply to ISO/IEC 7816-3, section 12.1)

Returns

tuple(data, sw), where

data : string (in hex) of returned data (ex. "074F4EFFFF") sw : string (in hex) of status word (ex. "9000")

send_apdu_checksw(*apdu: Hexstr*, *sw: SwMatchstr = '9000'*) → Tuple[Hexstr, SwHexstr]

Sends an APDU and check returned SW

Parameters

- apdu – string of hexadecimal characters (ex. "A0A40000023F00", must comply to ISO/IEC 7816-3, section 12.1)
- sw – string of 4 hexadecimal characters (ex. "9000"). The user may mask out certain digits using a '?' to add some ambiguity if needed.

Returns

tuple(data, sw), where

data : string (in hex) of returned data (ex. "074F4EFFFF") sw : string (in hex) of status word (ex. "9000")

set_sw_interpreter(*interp*)

Set an (optional) status word interpreter.

abstractmethod wait_for_card(*timeout: int | None = None*, *newcardonly: bool = False*)

Wait for a card and connect to it

Parameters

- timeout – Maximum wait time in seconds (None=no timeout)
- newcardonly – Should we wait for a new card, or an already inserted one ?

class pySim.transport.LinkBaseTpdu(*sw_interpreter=None*, *apdu_tracer: ApduTracer | None = None*, *proactive_handler:* ProactiveHandler *| None = None*)

abstractmethod send_tpdu(*tpdu: Hexstr*) → Tuple[Hexstr, SwHexstr]

Implementation specific method for sending the resulting TPDU. This method must accept TPDUs as defined in ETSI TS 102 221, section 7.3.1 and 7.3.2, depending on the protocol selected.

set_tpdu_format(*protocol: int*)

Set TPDU format. Each transport protocol has its specific TPDU format. This method allows the concrete transport layer implementation to set the TPDU format it expects. (This method must not be called by higher layers. Switching the TPDU format does not switch the transport protocol that the reader uses on the wire)

Parameters

protocol – number of the transport protocol used. (0 => T=0, 1 => T=1)

class pySim.transport.ProactiveHandler

Abstract base class representing the interface of some code that handles the proactive commands, as returned by the card in responses to the FETCH command.

receive_fetch(*pcmd: ProactiveCommand*)

Default handler for not otherwise handled proactive commands.

class pySim.transport.StdoutApduTracer

Minimalistic APDU tracer, printing commands to stdout.

pySim.transport.argparse_add_reader_args(*arg_parser: ArgumentParser*)

Add all reader related arguments to the given argparse.Argumentparser instance.

pySim.transport.init_reader(*opts*, ***kwargs*) → *LinkBase*

Init card reader driver

calypso / OsmocomBB transport

This allows the use of the SIM slot of an OsmocomBB compatible phone with the TI Calypso chipset, using the L1CTL interface to talk to the layer1.bin firmware on the phone.

class pySim.transport.calypso.CalypsoSimLink(*opts: Namespace = Namespace(osmocon_sock='/tmp/osmocom_l2')*, ***kwargs*)

Transport Link for Calypso based phones.

connect()

Connect to a card immediately

disconnect()

Disconnect from card

get_atr() → Hexstr

Retrieve card ATR

send_tpdu(*tpdu: Hexstr*) → Tuple[Hexstr, SwHexstr]

Implementation specific method for sending the resulting TPDU. This method must accept TPDUs as defined in ETSI TS 102 221, section 7.3.1 and 7.3.2, depending on the protocol selected.

wait_for_card(*timeout: int | None = None*, *newcardonly: bool = False*)

Wait for a card and connect to it

Parameters

- timeout – Maximum wait time in seconds (None=no timeout)
- newcardonly – Should we wait for a new card, or an already inserted one ?

AT-command Modem transport

This transport uses AT commands of a cellular modem in order to get access to the SIM card inserted in such a modem.

class pySim.transport.modem_atcmd.ModemATCommandLink(*opts: Namespace = Namespace(modem_dev='/dev/ttyUSB0', modem_baud=115200)*, ***kwargs*)

Transport Link for 3GPP TS 27.007 compliant modems.

connect()

Connect to a card immediately

disconnect()

Disconnect from card

get_atr() → Hexstr

Retrieve card ATR

send_tpdu(*tpdu: Hexstr*) → Tuple[Hexstr, SwHexstr]

Implementation specific method for sending the resulting TPDU. This method must accept TPDUs as defined in ETSI TS 102 221, section 7.3.1 and 7.3.2, depending on the protocol selected.

wait_for_card(*timeout: int | None = None*, *newcardonly: bool = False*)

Wait for a card and connect to it

Parameters

- timeout – Maximum wait time in seconds (None=no timeout)
- newcardonly – Should we wait for a new card, or an already inserted one ?

PC/SC transport

PC/SC is the standard API for accessing smart card interfaces on all major operating systems, including the MS Windows Family, OS X as well as Linux / Unix OSs.

class pySim.transport.pcsc.PcscSimLink(*opts: Namespace = Namespace(pcsc_dev=0)*, ***kwargs*)

pySim: PCSC reader transport link.

connect()

Connect to a card immediately

disconnect()

Disconnect from card

get_atr() → Hexstr

Retrieve card ATR

send_tpdu(*tpdu: Hexstr*) → Tuple[Hexstr, SwHexstr]

Implementation specific method for sending the resulting TPDU. This method must accept TPDUs as defined in ETSI TS 102 221, section 7.3.1 and 7.3.2, depending on the protocol selected.

wait_for_card(*timeout: int | None = None*, *newcardonly: bool = False*)

Wait for a card and connect to it

Parameters

- timeout – Maximum wait time in seconds (None=no timeout)
- newcardonly – Should we wait for a new card, or an already inserted one ?

Serial/UART transport

This transport implements interfacing smart cards via very simplistic UART readers. These readers basically wire together the Rx+Tx pins of a RS232 UART, provide a fixed crystal oscillator for clock, and operate the UART at 9600 bps. These readers are sometimes called *Phoenix*.

class pySim.transport.serial.SerialSimLink(*opts=Namespace(device='/dev/ttyUSB0', baudrate=9600)*, *rst: str = '-rts'*, *debug: bool = False*, ***kwargs*)

pySim: Transport Link for serial (RS232) based readers included with simcard

connect()

Connect to a card immediately

disconnect()

Disconnect from card

get_atr() → Hexstr

Retrieve card ATR

send_tpdu(*tpdu: Hexstr*) → Tuple[Hexstr, SwHexstr]

Implementation specific method for sending the resulting TPDU. This method must accept TPDUs as defined in ETSI TS 102 221, section 7.3.1 and 7.3.2, depending on the protocol selected.

wait_for_card(*timeout: int | None = None*, *newcardonly: bool = False*)

Wait for a card and connect to it

Parameters

- timeout – Maximum wait time in seconds (None=no timeout)
- newcardonly – Should we wait for a new card, or an already inserted one ?

1.6.4 pySim utility functions

pySim: various utilities

class pySim.utils.CardCommand(*name*, *ins*, *cla_list=None*, *desc=None*)

A single card command / instruction.

match_cla(*cla*)

Does the given CLA match the CLA list of the command?.

class pySim.utils.CardCommandSet(*name*, *cmds=[]*)

A set of card instructions, typically specified within one spec.

lookup(*ins*, *cla=None*)

look-up the command within the CommandSet.

class pySim.utils.DataObject(*name: str*, *desc: str | None = None*, *tag: int | None = None*)

A DataObject (DO) in the sense of ISO 7816-4. Contrary to 'normal' TLVs where one simply has any number of different TLVs that may occur in any order at any point, ISO 7816 has the habit of specifying TLV data but with very specific ordering, or specific choices of tags at specific points in a stream. This class tries to represent this.

Parameters

- name – A brief, all-lowercase, underscore separated string identifier
- desc – A human-readable description of what this DO represents
- tag – The tag associated with this DO

decode(*binary: bytes*) → Tuple[dict, bytes]

Decode a single DOs from the input data. :param binary: binary bytes of encoded data

Returns

tuple of (decoded_result, binary_remainder)

abstractmethod from_bytes(*do: bytes*)

Parse the value part of the DO into the internal state of this instance. :param do: binary encoded bytes

from_tlv(*do: bytes*) → bytes

Parse binary TLV representation into internal state. The resulting decoded representation is _not_ returned, but just internalized in the object instance! :param do: input bytes containing TLV-encoded representation

Returns

bytes remaining at end of 'do' after parsing one TLV/DO.

abstractmethod to_bytes() → bytes

Encode the internal state of this instance into the TLV value part. :returns: binary bytes encoding the internal state

to_dict() → dict

Return a dict in form "name: decoded_value"

to_tlv() → bytes

Encode internal representation to binary TLV. :returns: bytes encoded in TLV format.

class pySim.utils.DataObjectChoice(*name: str*, *desc: str | None = None*, *members=None*)

One Data Object from within a choice, identified by its tag. This means that exactly one member of the choice must occur, and which one occurs depends on the tag.

decode(*binary: bytes*) → Tuple[dict, bytes]

Decode a single DOs from the choice based on the tag. :param binary: binary bytes of encoded data

Returns

tuple of (decoded_result, binary_remainder)

class pySim.utils.DataObjectCollection(*name: str*, *desc: str | None = None*, *members=None*)

A DataObjectCollection consists of multiple Data Objects identified by their tags. A given encoded DO may contain any of them in any order, and may contain multiple instances of each DO.

decode(*binary: bytes*) → Tuple[List, bytes]

Decode any number of DOs from the collection until the end of the input data, or uninitialized memory (0xFF) is found. :param binary: binary bytes of encoded data

Returns

tuple of (decoded_result, binary_remainder)

class pySim.utils.DataObjectSequence(*name: str*, *desc: str | None = None*, *sequence=None*)

A sequence of DataObjects or DataObjectChoices. This allows us to express a certain ordered sequence of DOs or choices of DOs that have to appear as per the specification. By wrapping them into this formal DataObject-Sequence, we can offer convenience methods for encoding or decoding an entire sequence.

decode(*binary: bytes*) → Tuple[list, bytes]

Decode a sequence by calling the decoder of each element in the sequence. :param binary: binary bytes of encoded data

Returns

tuple of (decoded_result, binary_remainder)

decode_multi(*do: bytes*) → Tuple[list, bytes]

Decode multiple occurrences of the sequence from the binary input data. :param do: binary input data to be decoded

Returns

list of results of the decoder of this sequences

encode(*decoded*) → bytes

Encode a sequence by calling the encoder of each element in the sequence.

encode_multi(*decoded*) → bytes

Encode multiple occurrences of the sequence from the decoded input data. :param decoded: list of json-serializable input data; one sequence per list item

Returns
binary encoded output data

class pySim.utils.TL0_DataObject(*name: str*, *desc: str*, *tag: int*, *val=None*)

Data Object that has Tag, Len=0 and no Value part.

Parameters

- name – A brief, all-lowercase, underscore separated string identifier
- desc – A human-readable description of what this DO represents
- tag – The tag associated with this DO

from_bytes(*binary: bytes*)

Parse the value part of the DO into the internal state of this instance. :param do: binary encoded bytes

to_bytes() → bytes

Encode the internal state of this instance into the TLV value part. :returns: binary bytes encoding the internal state

pySim.utils.boxed_heading_str(*heading*, *width=80*)

Generate a string that contains a boxed heading.

pySim.utils.bytes_for_nibbles(*num_nibbles: int*) → int

compute the number of bytes needed to store the given number of nibbles.

pySim.utils.calculate_luhn(*cc*) → int

Calculate Luhn checksum used in e.g. ICCID and IMEI

pySim.utils.dec_imsi(*ef: Hexstr*) → str | None

Converts an EF value to the IMSI string representation

pySim.utils.decomposeATR(*atr_txt*)

Decompose the ATR in elementary fields

Parameters
atr_txt – ATR as a hex bytes string

Returns
dictionary of field and values

Example:

```
>>> decomposeATR("3B A7 00 40 18 80 65 A2 08 01 01 52")
 { 'T0': {'value': 167},
   'TB': {1: {'value': 0}},
   'TC': {2: {'value': 24}},
   'TD': {1: {'value': 64}},
   'TS': {'value': 59},
   'atr': [59, 167, 0, 64, 24, 128, 101, 162, 8, 1, 1, 82],
   'hb': {'value': [128, 101, 162, 8, 1, 1, 82]},
   'hbn': 7}
```

pySim.utils.derive_mcc(*digit1: int*, *digit2: int*, *digit3: int*) → int

Derive decimal representation of the MCC (Mobile Country Code) from three given digits.

pySim.utils.derive_milenage_opc(*ki_hex: Hexstr*, *op_hex: Hexstr*) → Hexstr

Run the milenage algorithm to calculate OPC from Ki and OP

pySim.utils.derive_mnc(*digit1: int*, *digit2: int*, *digit3: int = 15*) → int

Derive decimal representation of the MNC (Mobile Network Code) from two or (optionally) three given digits.

pySim.utils.enc_imsi(*imsi: str*)

Converts a string IMSI into the encoded value of the EF

pySim.utils.enc_plmn(*mcc: Hexstr*, *mnc: Hexstr*) → Hexstr

Converts integer MCC/MNC into 3 bytes for EF

pySim.utils.expand_hex(*hexstring*, *length*)

Expand a given hexstring to a specified length by replacing "." or ".."
with a filler that is derived from the neighboring nibbles respective bytes. Usually this will be the nibble respective byte before "." or "..", except when the string begins with "." or "..", then the nibble respective byte after "." or ".." is used.". In case the string cannot be expanded for some reason, the input string is returned unmodified.

Parameters

- hexstring – hexstring to expand
- length – desired length of the resulting hexstring.

Returns
expanded hexstring

pySim.utils.get_addr_type(*addr*)

Validates the given address and returns it's type (FQDN or IPv4 or IPv6) Return: 0x00 (FQDN), 0x01 (IPv4), 0x02 (IPv6), None (Bad address argument given)

TODO: Handle IPv6

pySim.utils.mcc_from_imsi(*imsi: str*) → str | None

Derive the MCC (Mobile Country Code) from the first three digits of an IMSI

pySim.utils.mnc_from_imsi(*imsi: str*, *long: bool = False*) → str | None

Derive the MNC (Mobile Country Code) from the 4th to 6th digit of an IMSI

pySim.utils.normalizeATR(*atr*)

Transform an ATR in list of integers. valid input formats are "3B A7 00 40 18 80 65 A2 08 01 01 52" "3B:A7:00:40:18:80:65:A2:08:01:01:52"

Parameters
atr – string

Returns
list of bytes

```
>>> normalize("3B:A7:00:40:18:80:65:A2:08:01:01:52")
[59, 167, 0, 64, 24, 128, 101, 162, 8, 1, 1, 82]
```

pySim.utils.parse_command_apdu(*apdu: bytes*) → int

Parse a given command APDU and return case (see also ISO/IEC 7816-3, Table 12 and Figure 26), lc, le and the data field.

Parameters
apdu – hexstring that contains the command APDU

Returns
tuple containing case, lc and le values of the APDU (case, lc, le, data)

pySim.utils.sanitize_pin_adm(*pin_adm*, *pin_adm_hex=None*) → Hexstr

The ADM pin can be supplied either in its hexadecimal form or as ascii string. This function checks the supplied opts parameter and returns the pin_adm as hex encoded string, regardless in which form it was originally supplied by the user

pySim.utils.sw_match(*sw: str*, *pattern: str*) → bool

Match given SW against given pattern.

pySim.utils.tabulate_str_list(*str_list*, *width: int = 79*, *hspace: int = 2*, *lspace: int = 1*, *align_left: bool = True*) → str

Pretty print a list of strings into a tabulated form.

Parameters
- width – total width in characters per line
- space – horizontal space between cells
- lspace – number of spaces before row
- align_lef – Align text to the left side

Returns
multi-line string containing formatted table

pySim.utils.verify_luhn(*digits: str*)

Verify the Luhn check digit; raises ValueError if it is incorrect.

1.6.5 pySim exceptions

pySim: Exceptions

exception pySim.exceptions.NoCardError

No card was found in the reader.

exception pySim.exceptions.ProtocolError

Some kind of protocol level error interfacing with the card.

exception pySim.exceptions.ReaderError

Some kind of general error with the card reader.

exception pySim.exceptions.SwMatchError(*sw_actual: str*, *sw_expected*, *rs=None*)

Raised when an operation specifies an expected SW but the actual SW from the card doesn't match.

Parameters
- sw_actual – the SW we actually received from the card (4 hex digits)
- sw_expected – the SW we expected to receive from the card (4 hex digits), or a list of acceptable ones
- rs – interpreter class to convert SW to string

1.6.6 pySim card_handler

pySim: card handler utilities. A 'card handler' is some method by which cards can be inserted/removed into the card reader. For normal smart card readers, this has to be done manually. However, there are also automatic card feeders.

class pySim.card_handler.CardHandler(*sl:* LinkBase)

Manual card handler: User is prompted to insert/remove card from the reader.

class pySim.card_handler.CardHandlerAuto(*sl:* LinkBase, *config_file: str*)

Automatic card handler: A machine is used to handle the cards.

class pySim.card_handler.CardHandlerBase(*sl:* LinkBase)

Abstract base class representing a mechanism for card insertion/removal.

done()

Method called when pySim failed to program a card. Move card to 'good' batch.

error()

Method called when pySim failed to program a card. Move card to 'bad' batch.

get(*first: bool = False*)

Method called when pySim needs a new card to be inserted.

Parameters

first – set to true when the get method is called the first time. This is required to prevent blocking when a card is already inserted into the reader. The reader API would not recognize that card as "new card" until it would be removed and re-inserted again.

1.6.7 pySim card_key_provider

Obtaining card parameters (mostly key data) from external source.

This module contains a base class and a concrete implementation of obtaining card key material (or other card-individual parameters) from an external data source.

This is used e.g. to keep PIN/PUK data in some file on disk, avoiding the need of manually entering the related card-individual data on every operation with pySim-shell.

class pySim.card_key_provider.CardKeyFieldCryptor(*transport_keys: dict*)

A Card key field encryption class that may be used by Card key provider implementations to add support for a column-based encryption to protect sensitive material (cryptographic key material, ADM keys, etc.). The sensitive material is encrypted using a "key-encryption key", occasionally also known as "transport key" before it is stored into a file or database (see also GSMA FS.28). The "transport key" is then used to decrypt the key material on demand.

Create new field encryptor/decryptor object and set transport keys, usually one for each column. In some cases it is also possible to use a single key for multiple columns (see also __CRYPT_GROUPS)

Parameters

transport_keys – a dict indexed by field name, whose values are hex-encoded AES keys for the respective field (column) of the CSV. This is done so that different fields (columns) can use different transport keys, which is strongly recommended by GSMA FS.28

decrypt_field(*field_name: str*, *encrypted_val: str*) → str

Decrypt a single field. The decryption is only applied if we have a transport key is known under the provided field name, otherwise the field is treated as plaintext and passed through as it is.

Parameters

- field_name – name of the field to decrypt (used to identify which key to use)

- encrypted_val – encrypted field value

Returns
plaintext field value

encrypt_field(*field_name: str*, *plaintext_val: str*) → str

Encrypt a single field. The encryption is only applied if we have a transport key is known under the provided field name, otherwise the field is treated as non sensitive and passed through as it is.

Parameters

- field_name – name of the field to decrypt (used to identify which key to use)
- encrypted_val – encrypted field value

Returns
plaintext field value

static transport_keys_from_opts(*opts: Namespace*) → dict

Transport keys are passed via the commandline using the '–column-key' option. Each column requires a dedicated transport key. This method can be used to extract the column keys parameters from the commandline options into a dict that can be directly passed to the construtor with the transport_keys argument.

Parameters
opts – parsed commandline options (Namespace)

class pySim.card_key_provider.CardKeyProvider

Base class, not containing any concrete implementation.

static argparse_add_args(*arg_parser: ArgumentParser*)

Add the commandline arguments relevant for this card key provider.

Parameters
arg_parser – argument parser group

abstractmethod get(*fields: List[str]*, *key: str*, *value: str*) → Dict[str, str]

Get multiple card-individual fields for identified card. This method should not fail with an exception in case the entry, columns or even the key column itsself is not found.

Parameters

- fields – list of valid field names such as 'ADM1', 'PIN1', … which are to be obtained
- key – look-up key to identify card data, such as 'ICCID'
- value – value for look-up key to identify card data

Returns
value, … } strings for each requested field from 'fields'. In case nothing is fond None shall be returned.

Return type
dictionary of {field

class pySim.card_key_provider.CardKeyProviderCsv(*csv_filename: str*, *field_cryptor:* CardKeyFieldCryptor)

Card key provider implementation that allows to query against a specified CSV file.

Parameters

- csv_filename – file name (path) of CSV file containing card-individual key/data
- field_cryptor – (see class CardKeyFieldCryptor)

static argparse_add_args(*arg_parser: ArgumentParser*)

Add the commandline arguments relevant for this card key provider.

Parameters
arg_parser – argument parser group

get(*fields: List[str]*, *key: str*, *value: str*) → Dict[str, str]

Get multiple card-individual fields for identified card. This method should not fail with an exception in case the entry, columns or even the key column itsself is not found.

Parameters

- fields – list of valid field names such as 'ADM1', 'PIN1', … which are to be obtained
- key – look-up key to identify card data, such as 'ICCID'
- value – value for look-up key to identify card data

Returns
value, … } strings for each requested field from 'fields'. In case nothing is fond None shall be returned.

Return type
dictionary of {field

class pySim.card_key_provider.CardKeyProviderPgsql(*config_filename: str*, *field_cryptor:* CardKeyFieldCryptor)

Card key provider implementation that allows to query against a specified PostgreSQL database table.

Parameters

- config_filename – file name (path) of CSV file containing card-individual key/data
- field_cryptor – (see class CardKeyFieldCryptor)

static argparse_add_args(*arg_parser: ArgumentParser*)

Add the commandline arguments relevant for this card key provider.

Parameters
arg_parser – argument parser group

get(*fields: List[str]*, *key: str*, *value: str*) → Dict[str, str]

Get multiple card-individual fields for identified card. This method should not fail with an exception in case the entry, columns or even the key column itsself is not found.

Parameters

- fields – list of valid field names such as 'ADM1', 'PIN1', … which are to be obtained
- key – look-up key to identify card data, such as 'ICCID'
- value – value for look-up key to identify card data

Returns
value, … } strings for each requested field from 'fields'. In case nothing is fond None shall be returned.

Return type
dictionary of {field

pySim.card_key_provider.card_key_provider_argparse_add_args(*arg_parser: ArgumentParser*)

Add card key provider commandline options to the given argument parser

pySim.card_key_provider.card_key_provider_get(*fields: list[str]*, *key: str*, *value: str*, *provider_list=[]*) → Dict[str, str]

Query all registered card data providers for card-individual [key] data.

Parameters

- fields – list of valid field names such as 'ADM1', 'PIN1', … which are to be obtained
- key – look-up key to identify card data, such as 'ICCID'
- value – value for look-up key to identify card data
- provider_list – override the list of providers from the global default

Returns

dictionary of {field, value} strings for each requested field from 'fields'

pySim.card_key_provider.card_key_provider_get_field(*field: str*, *key: str*, *value: str*, *provider_list=[]*) → str

Query all registered card data providers for a single field.

Parameters

- field – name valid field such as 'ADM1', 'PIN1', … which is to be obtained
- key – look-up key to identify card data, such as 'ICCID'
- value – value for look-up key to identify card data
- provider_list – override the list of providers from the global default

Returns

dictionary of {field, value} strings for the requested field

pySim.card_key_provider.card_key_provider_init(*opts: Namespace*)

Initialize card key provider depending on the user provided commandline options

pySim.card_key_provider.card_key_provider_register(*provider:* CardKeyProvider, *provider_list=[]*)

Register a new card key provider.

Parameters

- provider – the to-be-registered provider
- provider_list – override the list of providers from the global default

1.7 pySim eSIM libraries

The pySim eSIM libraries implement a variety of functionality related to the GSMA eSIM universe, including the various interfaces of SGP.21 + SGP.22, as well as Interoperable Profile decioding, validation, personalization and encoding.

class pySim.esim.ActivationCode(*hostname: str*, *token: str*, *oid: str | None = None*, *cc_required: bool | None = False*)

SGP.22 section 4.1 Activation Code

static decode_str(*ac: str*) → dict

decode an activation code from its string representation.

classmethod from_string(*ac: str*) → *ActivationCode*

Create new instance from SGP.22 section 4.1 string representation.

to_qrcode()

Encode internal representation to QR code.

to_string(*for_qrcode: bool = False*) → str

Convert from internal representation to SGP.22 section 4.1 string representation.

class pySim.esim.PMO(*op: str*)

Convenience conversion class for ProfileManagementOperation as used in ES9+ notifications.

classmethod from_bitstring(*bstr: Tuple[bytes, int]*) → *PMO*

Parse a asn1tools BITSTRING representation.

classmethod from_int(*i: int*) → *PMO*

Parse an integer representation.

to_bitstring() → Tuple[bytes, int]

return value in a format as used by asn1tools for BITSTRING.

pySim.esim.compile_asn1_subdir(*subdir_name: str*, *codec='der'*)

Helper function that compiles ASN.1 syntax from all files within given subdir

1.7.1 GSMA SGP.21/22 Remote SIM Provisioning (RSP) - High Level

pySim.esim.rsp

Implementation of GSMA eSIM RSP (Remote SIM Provisioning) as per SGP22 v3.0

class pySim.esim.rsp.RspSessionState(*transactionId: str*, *serverChallenge: bytes*, *ci_cert_id: bytes*)

Encapsulates the state of a RSP session. It is created during the initiateAuthentication and subsequently used by further API calls using the same transactionId. The session state is removed either after cancelSession or after notification. TODO: add some kind of time based expiration / garbage collection.

class pySim.esim.rsp.RspSessionStore(*filename: str | None = None*, *in_memory: bool = False*)

A wrapper around the database-backed storage 'shelve' for storing RspSessionState objects. Can be configured to use either file-based storage or in-memory storage. We use it to store RspSessionState objects indexed by transactionId.

close()

Close the session store.

sync()

Synchronize the cache with the underlying storage.

pySim.esim.rsp.extract_euiccSigned1(*authenticateServerResponse: bytes*) → bytes

Extract the raw, DER-encoded binary euiccSigned1 field from the given AuthenticateServerResponse. This is needed due to the very peculiar SGP.22 notion of signing sections of DER-encoded ASN.1 objects.

pySim.esim.rsp.extract_euiccSigned2(*prepareDownloadResponse: bytes*) → bytes

Extract the raw, DER-encoded binary euiccSigned2 field from the given prepareDownloadrResponse. This is needed due to the very peculiar SGP.22 notion of signing sections of DER-encoded ASN.1 objects.

pySim.esim.es2p

GSMA eSIM RSP ES2+ interface according to SGP.22 v2.5

class pySim.esim.es2p.CancelOrder(**args*, *role='legacy_client'*, ***kwargs*)

Parameters

- args – (see JsonHttpApiClient and JsonHttpApiServer)
- role – role ('server' or 'client') in which the JsonHttpApiFunction should be created.
- kwargs – (see JsonHttpApiClient and JsonHttpApiServer)

class pySim.esim.es2p.ConfirmOrder(**args*, *role='legacy_client'*, ***kwargs*)

Parameters

- args – (see JsonHttpApiClient and JsonHttpApiServer)
- role – role ('server' or 'client') in which the JsonHttpApiFunction should be created.
- kwargs – (see JsonHttpApiClient and JsonHttpApiServer)

class pySim.esim.es2p.DownloadOrder(**args*, *role='legacy_client'*, ***kwargs*)

Parameters

- args – (see JsonHttpApiClient and JsonHttpApiServer)
- role – role ('server' or 'client') in which the JsonHttpApiFunction should be created.
- kwargs – (see JsonHttpApiClient and JsonHttpApiServer)

class pySim.esim.es2p.Es2PlusApiFunction(**args*, *role='legacy_client'*, ***kwargs*)

Base class for representing an ES2+ API Function.

Parameters

- args – (see JsonHttpApiClient and JsonHttpApiServer)
- role – role ('server' or 'client') in which the JsonHttpApiFunction should be created.
- kwargs – (see JsonHttpApiClient and JsonHttpApiServer)

class pySim.esim.es2p.Es2pApiClient(*url_prefix: str*, *func_req_id: str*, *server_cert_verify: str = None*, *client_cert: str = None*)

Main class representing a full ES2+ API client. Has one method for each API function.

call_cancelOrder(*data: dict*) → dict

Perform ES2+ CancelOrder function (SGP.22 section 5.3.3).

call_confirmOrder(*data: dict*) → dict

Perform ES2+ ConfirmOrder function (SGP.22 section 5.3.2).

call_downloadOrder(*data: dict*) → dict

Perform ES2+ DownloadOrder function (SGP.22 section 5.3.1).

call_handleDownloadProgressInfo(*data: dict*) → dict

Perform ES2+ HandleDownloadProgressInfo function (SGP.22 section 5.3.5).

call_releaseProfile(*data: dict*) → dict

Perform ES2+ CancelOrder function (SGP.22 section 5.3.4).

class pySim.esim.es2p.Es2pApiServer(*port: int*, *interface: str*, *server_cert: str = None*, *client_cert_verify: str = None*)

Main class representing a full ES2+ API server. Has one method for each API function.

class pySim.esim.es2p.Es2pApiServerHandlerMno

ES2+ (MNO side) API Server handler class. The API user is expected to override the contained methods.

abstractmethod call_handleDownloadProgressInfo(*data: dict) -> (<class 'dict'>, <class 'str'>)*

Perform ES2+ HandleDownloadProgressInfo function (SGP.22 section 5.3.5).

class pySim.esim.es2p.Es2pApiServerHandlerSmdpp

ES2+ (SMDP+ side) API Server handler class. The API user is expected to override the contained methods.

abstractmethod call_cancelOrder(*data: dict) -> (<class 'dict'>, <class 'str'>)*

Perform ES2+ CancelOrder function (SGP.22 section 5.3.3).

abstractmethod call_confirmOrder(*data: dict) -> (<class 'dict'>, <class 'str'>)*

Perform ES2+ ConfirmOrder function (SGP.22 section 5.3.2).

abstractmethod call_downloadOrder(*data: dict) -> (<class 'dict'>, <class 'str'>)*

Perform ES2+ DownloadOrder function (SGP.22 section 5.3.1).

abstractmethod call_releaseProfile(*data: dict) -> (<class 'dict'>, <class 'str'>)*

Perform ES2+ CancelOrder function (SGP.22 section 5.3.4).

class pySim.esim.es2p.Es2pApiServerMno(*port: int*, *interface: str*, *handler:* Es2pApiServerHandlerMno, *server_cert: str = None*, *client_cert_verify: str = None*)

ES2+ (MNO side) API Server.

call_handleDownloadProgressInfo(*request: twisted.web.server.Request*) → dict

Perform ES2+ HandleDownloadProgressInfo function (SGP.22 section 5.3.5).

class pySim.esim.es2p.Es2pApiServerSmdpp(*port: int*, *interface: str*, *handler:* Es2pApiServerHandlerSmdpp, *server_cert: str = None*, *client_cert_verify: str = None*)

ES2+ (SMDP+ side) API Server.

call_cancelOrder(*request: twisted.web.server.Request*) → dict

Perform ES2+ CancelOrder function (SGP.22 section 5.3.3).

call_confirmOrder(*request: twisted.web.server.Request*) → dict

Perform ES2+ ConfirmOrder function (SGP.22 section 5.3.2).

call_downloadOrder(*request: twisted.web.server.Request*) → dict

Perform ES2+ DownloadOrder function (SGP.22 section 5.3.1).

call_releaseProfile(*request: twisted.web.server.Request*) → dict

Perform ES2+ CancelOrder function (SGP.22 section 5.3.4).

class pySim.esim.es2p.HandleDownloadProgressInfo(**args*, *role='legacy_client'*, ***kwargs*)

Parameters

- args – (see JsonHttpApiClient and JsonHttpApiServer)
- role – role ('server' or 'client') in which the JsonHttpApiFunction should be created.
- kwargs – (see JsonHttpApiClient and JsonHttpApiServer)

class pySim.esim.es2p.ReleaseProfile(**args*, *role='legacy_client'*, ***kwargs*)

Parameters

- args – (see JsonHttpApiClient and JsonHttpApiServer)
- role – role ('server' or 'client') in which the JsonHttpApiFunction should be created.

- kwargs – (see JsonHttpApiClient and JsonHttpApiServer)

pySim.esim.es8p

Implementation of GSMA eSIM RSP (Remote SIM Provisioning) ES8+ as per SGP22 v3.0 Section 5.5

class pySim.esim.es8p.BoundProfilePackage(*metadata:* ProfileMetadata | *None* = *None*)

Representing a bound profile package (BPP) as defined in SGP.22 Section 2.5.4

decode(*euicc_ot*, *eid: str*, *bpp_bin: bytes*)

Decode a BPP into the PPP and subsequently UPP. This is what happens inside an eUICC.

encode(*ss:* RspSessionState, *dp_pb:* CertAndPrivkey) → bytes

Generate a bound profile package (SGP.22 2.5.4).

class pySim.esim.es8p.ProfileMetadata(*iccid_bin: bytes*, *spn: str*, *profile_name: str*, *profile_class='operational'*)

Representation of Profile metadata. Right now only the mandatory bits are supported, but in general this should follow the StoreMetadataRequest of SGP.22 5.5.3

add_notification(*event: str*, *address: str*)

Add an 'other' notification to the notification configuration of the metadata

gen_store_metadata_request() → bytes

Generate encoded (but unsigned) StoreMetadataRequest DO (SGP.22 5.5.3)

set_icon(*is_png: bool*, *icon_data: bytes*)

Set the icon that is part of the metadata.

class pySim.esim.es8p.ProtectedProfilePackage(*metadata:* ProfileMetadata | *None* = *None*)

Representing a protected profile package (PPP) as defined in SGP.22 Section 2.5.3

classmethod from_upp(*upp:* UnprotectedProfilePackage, *bsp:* BspInstance) → *ProtectedProfilePackage*

Generate the PPP as a sequence of encrypted and MACed Command TLVs representing the UPP

class pySim.esim.es8p.UnprotectedProfilePackage(*metadata:* ProfileMetadata | *None* = *None*)

Representing an unprotected profile package (UPP) as defined in SGP.22 Section 2.5.2

classmethod from_der(*der: bytes*, *metadata:* ProfileMetadata | *None* = *None*) → *UnprotectedProfilePackage*

Load an UPP from its DER representation.

to_der()

Return the DER representation of the UPP.

pySim.esim.es8p.gen_init_sec_chan_signed_part(*iscsp: Dict*) → bytes

Generate the concatenated remoteOpId, transactionId, controlRefTemplate and smdpOtpk data objects without the outer SEQUENCE tag / length or the remainder of initialiseSecureChannel, as is required for signing purpose.

pySim.esim.es8p.gen_initialiseSecureChannel(*transactionId: str*, *host_id: bytes*, *smdp_otpk: bytes*, *euicc_otpk: bytes*, *dp_pb*)

Generate decoded representation of (signed) initialiseSecureChannel (SGP.22 5.5.2)

pySim.esim.es8p.gen_replace_session_keys(*ppk_enc: bytes*, *ppk_cmac: bytes*, *initial_mcv: bytes*) → bytes

Generate encoded (but unsigned) ReplaceSessionKeysReqest DO (SGP.22 5.5.4)

pySim.esim.es8p.wrap_as_der_tlv(*tag: int*, *val: bytes*) → bytes

Wrap the 'value' into a DER-encoded TLV.

pySim.esim.es9p

GSMA eSIM RSP ES9+ interface according to SGP.22 v2.5

class pySim.esim.es9p.AuthenticateClient(**args*, *role='legacy_client'*, ***kwargs*)

Parameters

- args – (see JsonHttpApiClient and JsonHttpApiServer)
- role – role ('server' or 'client') in which the JsonHttpApiFunction should be created.
- kwargs – (see JsonHttpApiClient and JsonHttpApiServer)

class pySim.esim.es9p.CancelSession(**args*, *role='legacy_client'*, ***kwargs*)

Parameters

- args – (see JsonHttpApiClient and JsonHttpApiServer)
- role – role ('server' or 'client') in which the JsonHttpApiFunction should be created.
- kwargs – (see JsonHttpApiClient and JsonHttpApiServer)

class pySim.esim.es9p.Es9PlusApiFunction(**args*, *role='legacy_client'*, ***kwargs*)

Parameters

- args – (see JsonHttpApiClient and JsonHttpApiServer)
- role – role ('server' or 'client') in which the JsonHttpApiFunction should be created.
- kwargs – (see JsonHttpApiClient and JsonHttpApiServer)

class pySim.esim.es9p.GetBoundProfilePackage(**args*, *role='legacy_client'*, ***kwargs*)

Parameters

- args – (see JsonHttpApiClient and JsonHttpApiServer)
- role – role ('server' or 'client') in which the JsonHttpApiFunction should be created.
- kwargs – (see JsonHttpApiClient and JsonHttpApiServer)

class pySim.esim.es9p.HandleNotification(**args*, *role='legacy_client'*, ***kwargs*)

Parameters

- args – (see JsonHttpApiClient and JsonHttpApiServer)
- role – role ('server' or 'client') in which the JsonHttpApiFunction should be created.
- kwargs – (see JsonHttpApiClient and JsonHttpApiServer)

class pySim.esim.es9p.InitiateAuthentication(**args*, *role='legacy_client'*, ***kwargs*)

Parameters

- args – (see JsonHttpApiClient and JsonHttpApiServer)
- role – role ('server' or 'client') in which the JsonHttpApiFunction should be created.
- kwargs – (see JsonHttpApiClient and JsonHttpApiServer)

1.7.2 GSMA SGP.21/22 Remote SIM Provisioning (RSP) - Low Level

pySim.esim.bsp

Implementation of GSMA eSIM RSP (Remote SIM Provisioning BSP (BPP Protection Protocol), where BPP is the Bound Profile Package. So the full expansion is the "GSMA eSIM Remote SIM Provisioning Bound Profile Packate Protection Protocol"

Originally (SGP.22 v2.x) this was called SCP03t, but it has since been renamed to BSP.

class pySim.esim.bsp.BspAlgo

Base class representing a cryptographic algorithm within the BSP (BPP Security Protocol).

class pySim.esim.bsp.BspAlgoCrypt(*s_enc: bytes*)

Base class representing an encryption/decryption algorithm within the BSP (BPP Security Protocol).

decrypt(*data: bytes*) → bytes

Decrypt given input bytes using the key material given in constructor.

encrypt(*data: bytes*) → bytes

Encrypt given input bytes using the key material given in constructor.

class pySim.esim.bsp.BspAlgoCryptAES128(*s_enc: bytes*)

AES-CBC-128 implementation of the BPP Security Protocol for GSMA SGP.22 eSIM.

class pySim.esim.bsp.BspAlgoMac(*s_mac: bytes*, *initial_mac_chaining_value: bytes*)

Base class representing a message authentication code algorithm within the BSP (BPP Security Protocol).

class pySim.esim.bsp.BspAlgoMacAES128(*s_mac: bytes*, *initial_mac_chaining_value: bytes*)

AES-CMAC-128 implementation of the BPP Security Protocol for GSMA SGP.22 eSIM.

class pySim.esim.bsp.BspInstance(*s_enc: bytes*, *s_mac: bytes*, *initial_mcv: bytes*)

An instance of the BSP crypto. Initialized once with the key material via constructor, then the user can call any number of encrypt_and_mac cycles to protect plaintext and generate the respective ciphertext.

encrypt_and_mac_one(*tag: int*, *plaintext: bytes*) → bytes

Encrypt + MAC a single plaintext TLV. Returns the protected ciphertext.

classmethod from_kdf(*shared_secret: bytes*, *key_type: int*, *key_length: int*, *host_id: bytes*, *eid: bytes*)

Convenience constructor for constructing an instance with keys from KDF.

mac_only_one(*tag: int*, *plaintext: bytes*) → bytes

MAC a single plaintext TLV. Returns the protected ciphertext.

pySim.esim.bsp.bsp_key_derivation(*shared_secret: bytes*, *key_type: int*, *key_length: int*, *host_id: bytes*, *eid*, *l: int = 16*)

BSP protocol key derivation as per SGP.22 v3.0 Section 2.6.4.2

pySim.esim.http_json_api

GSMA eSIM RSP HTTP/REST/JSON interface according to SGP.22 v2.5

exception pySim.esim.http_json_api.ApiError(*func_ex_status: dict*)

Exception representing an error at the API level (status != Executed).

class pySim.esim.http_json_api.ApiParam

A class representing a single parameter in the API.

classmethod decode(*data*)

[Validate and] Decode the given value.

classmethod encode(*data*)

[Validate and] Encode the given value.

classmethod verify_decoded(*data*)

Verify the decoded representation of a value. Should raise an exception if something is odd.

classmethod verify_encoded(*data*)

Verify the encoded representation of a value. Should raise an exception if something is odd.

class pySim.esim.http_json_api.ApiParamBase64

class pySim.esim.http_json_api.ApiParamBoolean

Base class representing an API parameter of 'boolean' type.

class pySim.esim.http_json_api.ApiParamFqdn

String, as a list of domain labels concatenated using the full stop (dot, period) character as separator between labels. Labels are restricted to the Alphanumeric mode character set defined in table 5 of ISO/IEC 18004

classmethod verify_encoded(*data*)

Verify the encoded representation of a value. Should raise an exception if something is odd.

class pySim.esim.http_json_api.ApiParamInteger

Base class representing an API parameter of 'integer' type.

classmethod verify_decoded(*data*)

Verify the decoded representation of a value. Should raise an exception if something is odd.

classmethod verify_encoded(*data*)

Verify the encoded representation of a value. Should raise an exception if something is odd.

class pySim.esim.http_json_api.ApiParamString

Base class representing an API parameter of 'string' type.

exception pySim.esim.http_json_api.HttpHeaderError

exception pySim.esim.http_json_api.HttpStatusError

class pySim.esim.http_json_api.JsonHttpApiFunction(**args*, *role='legacy_client'*, ***kwargs*)

Base class for representing an HTTP[s] API Function.

Parameters

- args – (see JsonHttpApiClient and JsonHttpApiServer)
- role – role ('server' or 'client') in which the JsonHttpApiFunction should be created.
- kwargs – (see JsonHttpApiClient and JsonHttpApiServer)

decode_client(*data: dict*) → dict

[further] Decode and validate the JSON-Dict of the response body.

decode_server(*data: dict*) → dict

[further] Decode and validate the JSON-Dict of the request body.

encode_client(*data: dict*) → dict

Validate an encode input dict into JSON-serializable dict for request body.

encode_server(*data: dict*) → dict

Validate an encode input dict into JSON-serializable dict for response body.

rewrite_url(*data: dict*, *url: str*) → Tuple[dict, str]

Rewrite a static URL using information passed in the data dict. This method may be overloaded by a derived class to allow fully dynamic URLs. The input parameters required for the URL rewriting may be passed using data parameter. In case those parameters are additional parameters that are not intended to be passed to the encode_client method later, they must be removed explcitly.

Parameters

- data – (see JsonHttpApiClient and JsonHttpApiServer)
- url – statically generated URL string (see comment in JsonHttpApiClient)

class pySim.esim.http_json_api.JsonRequestHeader

SGP.22 section 6.5.1.3.

classmethod verify_decoded(*data*)

Verify the decoded representation of a value. Should raise an exception if something is odd.

class pySim.esim.http_json_api.JsonResponseHeader

SGP.22 section 6.5.1.4.

classmethod verify_decoded(*data*)

Verify the decoded representation of a value. Should raise an exception if something is odd.

class pySim.esim.http_json_api.SmdpAddress

pySim.esim.x509_cert

Implementation of X.509 certificate handling in GSMA eSIM as per SGP22 v3.0

class pySim.esim.x509_cert.CertAndPrivkey(*required_policy_oid: ObjectIdentifier | None = None*, *cert: Certificate | None = None*, *priv_key=None*)

A pair of certificate and private key, as used for ECDSA signing.

ecdsa_sign(*plaintext: bytes*) → bytes

Sign some input-data using an ECDSA signature compliant with SGP.22, which internally refers to Global Platform 2.2 Annex E, which in turn points to BSI TS-03111 which states “concatenated raw R + S values”.

get_authority_key_identifier() → AuthorityKeyIdentifier

Return the AuthorityKeyIdentifier X.509 extension of the certificate.

get_cert_as_der() → bytes

Return certificate encoded as DER.

get_subject_alt_name() → SubjectAlternativeName

Return the SubjectAlternativeName X.509 extension of the certificate.

class pySim.esim.x509_cert.CertificateSet(*root_cert: Certificate*)

A set of certificates consisting of a trusted [self-signed] CA root certificate, and an optional number of intermediate certificates. Can be used to verify the certificate chain of any given other certificate.

add_intermediate_cert(*cert: Certificate*)

Add a potential intermediate certificate to the CertificateSet.

verify_cert_chain(*cert: Certificate*, *max_depth: int = 100*)

Verify if a given certificate’s signature chain can be traced back to the root CA of this CertificateSet.

exception pySim.esim.x509_cert.VerifyError

An error during certificate verification,

pySim.esim.x509_cert.cert_get_auth_key_id(*cert: Certificate*) → bytes

Obtain the authority key identifier of the given cert object (as raw bytes).

pySim.esim.x509_cert.cert_get_subject_key_id(*cert: Certificate*) → bytes

Obtain the subject key identifier of the given cert object (as raw bytes).

pySim.esim.x509_cert.cert_policy_has_oid(*cert: Certificate*, *match_oid: ObjectIdentifier*) → bool

Determine if given certificate has a certificatePolicy extension of matching OID.

pySim.esim.x509_cert.check_signed(*signed: Certificate*, *signer: Certificate*) → bool

Verify if 'signed' certificate was signed using 'signer'.

pySim.esim.x509_cert.ecdsa_dss_to_tr03111(*sig: bytes*) → bytes

convert from DER format to BSI TR-03111; first get long integers; then convert those to bytes.

1.7.3 SIMalliance / TCA Interoperable Profile

pySim.esim.saip

Implementation of SimAlliance/TCA Interoperable Profile handling

class pySim.esim.saip.ErasedStateBytesIO(*initial_bytes=b''*)

BytesIO whose implicit growth carries the erased state instead of NUL.

Writing at a position past the end of a BytesIO pads the gap with NUL. In a TCA profile file that gap is a region the Profile Package deliberately skipped, which on the eUICC keeps the content the file was created with – 'FF' when it was created without a fill/repeat pattern.

write(*b*)

Write bytes to file.

Return the number of bytes written.

class pySim.esim.saip.File(*pename: str*, *l: List[Tuple] | None = None*, *template:* FileTemplate *| None = None*, *name: str | None = None*)

Internal representation of a file in a profile filesystem.

Parameters

- pename – Name string of the profile element
- l – List of tuples [fileDescriptor, fillFileContent, fillFileOffset profile elements]
- template – Applicable FileTemplate describing defaults as per SAIP spec
- name – Human-readable name like EF.IMSI, DF.TELECOM, ADF.USIM, …

check_template_modification_rules()

Check template modification rules as per SAIP section 8.3.3.

expand_fill_pattern() → bytes

Expand the fill/repeat pattern as per TS 102 222 Section 6.3.2.2.2

TS 102 222 defines both patterns record-wise for a record oriented EF: tag 'C1' initialises "the first W-1 bytes of each record", and tag 'C2' repeats within a record and truncates "at the end of each record". Expanding once across file_size instead puts the leading bytes of the pattern only in the first record, which for EF.EXT* or EF.SMS means every later record claims a record type or status of 'FF'.

file_content_from_tuples(*l: List[Tuple]*) → bytes | None

linearize a list of fillFileContent / fillFileOffset tuples into a stream of bytes.

file_content_to_tuples(*optimize: bool = False*) → List[Tuple]

Encode the file contents into a list of fillFileContent / fillFileOffset tuples that can be fed into the asn.1 encoder. If optimize is True, it will try to encode only the differences from the fillFileContent of the profile template. Otherwise, the entire file contents will be encoded as-is.

property file_size: int | None

Return the size of the file in bytes.

from_fileDescriptor(*fileDescriptor: dict*)

Convert from 'fileDescriptor' as used by asn1tools for SAIP to internal representation

from_template(*template:* FileTemplate)

Determine defaults for file based on given FileTemplate.

from_tuples(*l: List[Tuple]*)

Parse a list of fileDescriptor, fillFileContent, fillFileOffset tuples into this instance.

static get_tuplelist_item(*l: List[Tuple]*, *key: str*)

get the [first] value matching given key from a list of (key, value) tuples.

static path_from_gfm(*bin_path: bytes*)

convert from byte-array of 16bit FIDs to list of integers

static path_to_gfm(*path: List[int]*) → bytes

convert from list of 16bit integers to byte-array

to_fileDescriptor() → dict

Convert from internal representation to 'fileDescriptor' as used by asn1tools for SAIP

to_gfm() → List[Tuple]

Generate a list of filePath, createFCP, fillFileContent, fillFileOffset tuples into this instance.

to_tuples() → List[Tuple]

Generate a list of fileDescriptor, fillFileContent, fillFileOffset tuples into this instance.

class pySim.esim.saip.FsNode(*fid: int*, *parent:* FsNode | *None*, *file:* File | *None = None*, *name: str | None = None*)

A node in the filesystem hierarchy. Each node can have a parent node and any number of children. Each node is identified uniquely within the parent by its numeric FID and its optional human-readable name. Each node usually is associated with an instance of the File class for the actual content of the file. FsNode is the base class used by more specific nodes, such as FsNode{EF,DF,ADF,MF}.

property fid_path: List[int]

Return the path of the node as list of integers.

property mf: FsNodeMF

Return the MF (root) of the hierarchy.

property name_path: List[str]

Return the path of the node as list of integers.

walk(*fn*, ***kwargs*)

call 'fn(self, **kwargs) for the File.

class pySim.esim.saip.FsNodeADF(*df_name: Hexstr*, *fid: int | None = None*, *parent:* FsNodeDF *| None = None*, *file:* File *| None = None*, *name: str | None = None*)

An ADF (Application Dedicated File) in the filesystem hierarchy.

class pySim.esim.saip.FsNodeDF(*fid: int*, *parent: FsNodeDf*, *file:* File *| None = None*, *name: str | None = None*)

A DF (Dedicated File) in the filesystem hierarchy.

add_child(*child:* FsNode)

Add a child to the list of children of this DF.

add_file(*file:* File) → *FsNodeDF*

Create and link an appropriate FsNode for the given 'file' and insert it. Returns the new current DF (it might have changed).

lookup_by_fidpath(*path: List[int]*) → *FsNode*

Look-up a FsNode based on the [fid baesd] given (absolute) path.

lookup_by_path(*path:* Path) → *FsNode*

Look-up a FsNode based on the [name based] given (absolute) path.

walk(*fn*, ***kwargs*)

call 'fn(self, **kwargs) for the DF and recursively for all children.

class pySim.esim.saip.FsNodeEF(*fid: int*, *parent:* FsNode *| None*, *file:* File *| None = None*, *name: str | None = None*)

An EF (Entry File) in the filesystem hierarchy.

class pySim.esim.saip.FsNodeMF(*file:* File *| None = None*)

The MF (Master File) in the filesystem hierarchy.

class pySim.esim.saip.FsProfileElement(*decoded=None*, *mandated: bool = True*, ***kwargs*)

A file-system bearing profile element, like MF, USIM, ...

We keep two major representations of the data: * The "decoded" member, as introduced by our parent class, containing asn1tools syntax * the "files" dict, consisting of File values indexed by PE-name strings

The methods pe2files and files2pe convert between those two representations.

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

add_file(*file:* File)

Add a File to the ProfileElement.

create_file(*pename: str*) → *File*

Programmatically create a file by its PE-Name.

file2pe(*file:* File)

Update the "decoded" member for the given file with the contents from the given File instance. We expect that the File instance is part of self.files

file_template_for_path(*path:* Path, *adf: str | None = None*) → *FileTemplate* | None

Resolve the FileTemplate for given path, if we have any matching.

Parameters

- path – the path for which we would like to resolve the FileTemplate
- adf – string name of the ADF which might be used with this PE

files2pe()

Update the "decoded" member for each file with the contents of the "files" member.

pe2files()

Update the "files" member with the contents of the "decoded" member.

supports_file_for_path(*path:* Path, *adf: str | None = None*) → bool

Does this ProfileElement support a file of given path?

class pySim.esim.saip.Naa

A class defining a Network Access Application (NAA)

class pySim.esim.saip.NaaCsim

A class representing the CSIM (CDMA) Network Access Application (NAA)

class pySim.esim.saip.NaaIsim

A class representing the ISIM Network Access Application (NAA)

class pySim.esim.saip.NaaUsim

A class representing the USIM Network Access Application (NAA)

class pySim.esim.saip.NonMatch(*a*, *b*, *size*)

Representing a contiguous non-matching block of data; the opposite of difflib.Match

Create new instance of Match(a, b, size)

classmethod from_matchlist(*l: List[Match]*, *size: int*) → List[*NonMatch*]

Build a list of non-matching blocks of data from its inverse (list of matching blocks). The caller must ensure that the input list is ordered, non-overlapping and only contains matches at equal offsets in a and b.

class pySim.esim.saip.ProfileElement(*decoded=None*, *mandated: bool = True*, *pe_sequence:* ProfileElementSequence | *None = None*)

Generic Class representing a Profile Element (PE) within a SAIP Profile. This may be used directly, but it's more likely sub-classed with a specific class for the specific profile element type, like e.g ProfileElementHeader, ProfileElementMF, …

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

classmethod class_for_petype(*pe_type: str*) → *ProfileElement* | None

Return the subclass implementing the given pe-type string.

classmethod from_der(*der: bytes*, *pe_sequence:* ProfileElementSequence | *None = None*) → *ProfileElement*

Construct an instance from given raw, DER encoded bytes.

Parameters

- der – raw, DER-encoded bytes of a single PE
- pe_sequence – back-reference to the PE-Sequence of which this PE is part of

property header

The decoded ProfileHeader.

property header_name: str

Return the name of the header field within the profile element.

property identification

An unique number for the PE within the PE-Sequence.

Type
The identification value

property templateID

Return the decoded templateID used by this profile element (if any).

to_der() → bytes

Build an encoded DER representation of the instance.

class pySim.esim.saip.ProfileElementAKA(*decoded: dict | None = None*, ***kwargs*)

Class representing the ProfileElement for Authentication and Key Agreement (AKA).

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

set_mapping(*aid: bytes*, *options: int = 6*)

Configure akaParametes for a mapping from another AID.

set_milenage(*k: bytes*, *opc: bytes*)

Configure akaParametes for MILENAGE.

set_tuak(*k: bytes*, *topc: bytes*, *num_of_keccak: int = 1*)

Configure akaParametes for TUAK.

set_xor3g(*k: bytes*)

Configure akaParametes for XOR-3G.

class pySim.esim.saip.ProfileElementApplication(*decoded: dict | None = None*, ***kwargs*)

Class representing an application ProfileElement.

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

add_instance(*aid: Hexstr*, *class_aid: Hexstr*, *inst_aid: Hexstr*, *app_privileges: Hexstr*, *app_spec_pars: Hexstr*, *uicc_toolkit_app_spec_pars: Hexstr = None*, *uicc_access_app_spec_pars: Hexstr = None*, *uicc_adm_access_app_spec_pars: Hexstr = None*, *volatile_memory_quota: Hexstr = None*, *non_volatile_memory_quota: Hexstr = None*, *process_data: list[Hexstr] = None*)

Create a new instance and add it to the instanceList

classmethod from_file(*filename: str*, *aid: Hexstr*, *sd_aid: Hexstr = None*, *non_volatile_code_limit: int = None*, *volatile_data_limit: int = None*, *non_volatile_data_limit: int = None*, *hash_value: Hexstr = None*) → *ProfileElementApplication*

Fill contents of application ProfileElement from a .cap file.

remove_instance(*inst_aid: Hexstr*)

Remove an instance from the instanceList

to_file(*filename: str*)

Write loadBlockObject contents of application ProfileElement to a .cap or .ijc file.

class pySim.esim.saip.ProfileElementCD(*decoded: dict | None = None*, ***kwargs*)

Class representing the ProfileElement for DF.CD

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

class pySim.esim.saip.ProfileElementCDMA(*decoded=None*, *mandated: bool = True*, *pe_sequence: ProfileElementSequence | None = None*)

Class representing the ProfileElement for the CDMA parameters (PE-CDMAParameter).

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

class pySim.esim.saip.ProfileElementDf5GProSe(*decoded: dict | None = None*, ***kwargs*)

Class representing the ProfileElement for DF.5GProSe

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

class pySim.esim.saip.ProfileElementDf5GS(*decoded: dict | None = None*, ***kwargs*)

Class representing the ProfileElement for ADF.USIM/DF.5GS

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

class pySim.esim.saip.ProfileElementDfSAIP(*decoded: dict | None = None*, ***kwargs*)

Class representing the ProfileElement for DF.SAIP

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

class pySim.esim.saip.ProfileElementDfSNPN(*decoded: dict | None = None*, ***kwargs*)

Class representing the ProfileElement for DF.SNPN

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

class pySim.esim.saip.ProfileElementEAP(*decoded: dict | None = None*, ***kwargs*)

Class representing the ProfileElement for DF.EAP

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

class pySim.esim.saip.ProfileElementEnd(*decoded: dict | None = None*, ***kwargs*)

Class representing the ProfileElement for the End of the PE-Sequence.

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

class pySim.esim.saip.ProfileElementGFM(*decoded=None*, *mandated: bool = True*, ***kwargs*)

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

files2pe()

Update the "decoded" member from the "files" member.

pe2files()

Update the "files" member with the contents of the "decoded" member.

supports_file_for_path(*path: Path*, *adf: str | None = None*) → bool

Does this ProfileElement support a file of given path?

class pySim.esim.saip.ProfileElementGsmAccess(*decoded: dict | None = None*, ***kwargs*)

Class representing the ProfileElement for ADF.USIM/DF.GSM-ACCESS

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

class pySim.esim.saip.ProfileElementHeader(*decoded: dict | None = None*, *ver_major: int | None = 2*, *ver_minor: int | None = 3*, *iccid: Hexstr | None = '00000000000000000000'*, *profile_type: str | None = None*, ***kwargs*)

Class representing the ProfileElement for the Header of the PE-Sequence.

You would usually initialize an instance either with a "decoded" argument (as read from a DER-encoded SAIP file via asn1tools), or [some of] the other arguments in case you're constructing a Profile Header from scratch.

Parameters

- decoded – asn1tools-generated decoded structure for this PE

- ver_major – Major SAIP version
- ver_minor – Minor SAIP version
- iccid – ICCID of the profile
- profile_type – operational, testing or bootstrap

class pySim.esim.saip.ProfileElementISIM(*decoded: dict | None = None*, ***kwargs*)

Class representing the ProfileElement for ADF.ISIM Mandatory Files

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

class pySim.esim.saip.ProfileElementIoT(*decoded: dict | None = None*, ***kwargs*)

Class representing the ProfileElement for the IoT Minimal Profile mandatory files. A single PE creates the MF-level files and ADF.USIM (Section 8.3.4.8.1 and 9.10.1 of TCA SAIP V3.4.1).

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

add_file(*file:* File)

Add a File to the ProfileElement.

file_template_for_path(*path:* Path, *adf: str | None = None*) → *FileTemplate* | None

Resolve the FileTemplate for given path, if we have any matching.

Parameters

- path – the path for which we would like to resolve the FileTemplate
- adf – string name of the ADF which might be used with this PE

class pySim.esim.saip.ProfileElementMF(*decoded: dict | None = None*, ***kwargs*)

Class representing the ProfileElement for the MF (Master File)

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

class pySim.esim.saip.ProfileElementNonStandard(*decoded=None*, *mandated: bool = True*, *pe_sequence: ProfileElementSequence | None = None*)

Class representing a non-standard, issuer specific ProfileElement (PE-NonStandard).

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

class pySim.esim.saip.ProfileElementOptISIM(*decoded: dict | None = None*, ***kwargs*)

Class representing the ProfileElement for ADF.ISIM Optional Files

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

class pySim.esim.saip.ProfileElementOptIoT(*decoded: dict | None = None*, ***kwargs*)

Class representing the ProfileElement for the IoT Minimal Profile optional files below ADF.USIM, including DF.5GS and DF.SAIP (Section 8.3.4.8.2 and 9.10.2 of TCA SAIP V3.4.1).

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

add_file(*file: File*)

Add a File to the ProfileElement.

file_template_for_path(*path: Path*, *adf: str | None = None*) → *FileTemplate* | None

Resolve the FileTemplate for given path, if we have any matching.

Parameters

- path – the path for which we would like to resolve the FileTemplate
- adf – string name of the ADF which might be used with this PE

class pySim.esim.saip.ProfileElementOptUSIM(*decoded: dict | None = None*, ***kwargs*)

Class representing the ProfileElement for ADF.USIM Optional Files

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

class pySim.esim.saip.ProfileElementPhonebook(*decoded: dict | None = None*, ***kwargs*)

Class representing the ProfileElement for DF.PHONEBOOK

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

class pySim.esim.saip.ProfileElementPin(*decoded: dict | None = None*, ***kwargs*)

Class representing the ProfileElement for a PIN (Personal Identification Number)

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

add_pin(*key_ref: int*, *pin_value: bytes*, *max_attempts: int = 3*, *retries_left: int = 3*, *unblock_ref: int | None = None*, *pin_attrib: int = 7*)

Add a PIN to the pinCodes ProfileElement

class pySim.esim.saip.ProfileElementPuk(*decoded: dict | None = None*, ***kwargs*)

Class representing the ProfileElement for a PUK (PIN Unblocking Code)

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

add_puk(*key_ref: int*, *puk_value: bytes*, *max_attempts: int = 10*, *retries_left: int = 10*)

Add a PUK to the pukCodes ProfileElement

class pySim.esim.saip.ProfileElementRFM(*decoded: dict | None = None*, *inst_aid: bytes | None = None*, *sd_aid: bytes | None = None*, *adf_aid: bytes | None = None*, *tar_list: List[bytes] | None = []*, *msl: int | None = 6*, ***kwargs*)

Class representing the ProfileElement for RFM (Remote File Management).

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

class pySim.esim.saip.ProfileElementSD(*decoded: dict | None = None*, ***kwargs*)

Class representing a securityDomain ProfileElement.

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

class C9(***kwargs*)

nested_collection_cls

alias of UiccSdInstallParams

add_key(*key:* SecurityDomainKey)

Add a given SecurityDomainKey to the keyList of the securityDomain.

add_scp(*scp: int*, *i: int*)

Add given SCP (and i parameter) to list of SCP of the Security Domain Install Params. Example: add_scp(0x03, 0x70) for SCP03, or add_scp(0x02, 0x55) for SCP02.

find_key(*key_version_number: int*, *key_id: int*) → *SecurityDomainKey* | None

Find and return (if any) the SecurityDomainKey for given KVN + KID.

has_scp(*scp: int*) → bool

Determine if SD Installation parameters already specify given SCP.

remove_scp(*scp: int*)

Remove given SCP from list of SCP of the Security Domain Install Params.

class pySim.esim.saip.ProfileElementSSD(*decoded: dict | None = None*, ***kwargs*)

Class representing a securityDomain ProfileElement for a SSD.

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute

- pe_sequence – back-reference to the PE-Sequence of which we're part of

class pySim.esim.saip.ProfileElementSequence

A sequence of ProfileElement objects, which is the overall representation of an eSIM profile.

This primarily contains a list of PEs (pe_list member) as well as a number of convenience indexes like the pe_by_type and pes_by_naa dicts that allow easier access to individual PEs within the sequence.

After calling the constructor, you have to further initialize the instance by either calling the parse_der() method, or by manually adding individual PEs, including the header and end PEs.

add_file_at_path(*path:* Path, *l: List*)

Add a file at given path. This assumes that there's only one instance of USIM/ISIM/CSIM inside the profile, as otherwise the path name would not be globally unique.

add_hdr_and_end()

Initialize the PE Sequence with a header and end PE.

add_ssd(*ssd:* ProfileElementSSD)

Add a SSD (Supplementary Security Domain) After MNO-SD/ISD-P.

append(*pe:* ProfileElement)

Append a given PE to the end of the PE Sequence

cd(*path: List[int]*)

Change the current directory to the [absolute] "path".

property cur_df: FsNodeDF | None

Current DF; this is where the next files are created.

classmethod from_der(*der: bytes*) → *ProfileElementSequence*

Construct an instance from given raw, DER encoded bytes.

get_closest_prev_pe_for_templateID(*cur:* ProfileElement, *tid: OID*) → *ProfileElement* | None

Return the PE of given templateID that is the closest PE prior to the given PE in the PE-Sequence.

get_index_by_pe(*pe:* ProfileElement) → int

Return a list with the indices of all instances of PEs of petype.

get_index_by_type(*petype: str*) → List[int]

Return a list with the indices of all instances of PEs of petype.

get_pe_for_type(*tname: str*) → *ProfileElement* | None

Return a single profile element for given profile element type. Works only for types of which there is only a single instance in the PE Sequence!

get_pes_for_templateID(*tid: OID*) → List[*ProfileElement*]

Return list of profile elements present for given profile element type.

get_pes_for_type(*tname: str*) → List[*ProfileElement*]

Return list of profile elements present for given profile element type.

property iccid: str | None

The ICCID of the profile.

insert_after_pe(*pe_before:* ProfileElement, *pe_new:* ProfileElement) → None

Insert a given [new] ProfileElement after a given [existing] PE in the PE Sequence.

insert_at_index(*idx: int*, *pe:* ProfileElement) → None

Insert a given [new] ProfileElement at given index into the PE Sequence.

static naa_for_path(*path:* Path) → *Naa* | None

determine the NAA for the given path

parse_der(*der: bytes*) → None

Parse a sequence of PE from SAIP DER format and store the result in self.pe_list.

pe_for_path(*path:* Path) → Tuple[*ProfileElement* | None, *FileTemplate* | None]

Return the ProfileElement instance that can contain a file with matching path. This will either be an existing PE within the sequence, or it will be a newly-allocated PE that is inserted into the sequence.

static peclass_for_path(*path:* Path) → Tuple[*ProfileElement* | None, *FileTemplate* | None]

Return the ProfileElement class that can contain a file with given path.

rebuild_mandatory_gfstelist()

(Re-)build the eUICC Mandatory GFSTEList of the ProfileHeader based on what's in the PE-Sequence. You would normally call this at the very end, before encoding a PE-Sequence to its DER format.

rebuild_mandatory_services()

(Re-)build the eUICC Mandatory services list of the ProfileHeader based on what's in the PE-Sequence. You would normally call this at the very end, before encoding a PE-Sequence to its DER format.

remove_naas_of_type(*naa:* Naa) → None

Remove all instances of NAAs of given type. This can be used, for example, to remove all CSIM NAAs from a profile. Will not just remove the PEs, but also any records in 'eUICC-Mandatory-services' or 'eUICC-Mandatory-GFSTEList'.

renumber_identification()

Re-generate the 'identification' numbering of all PE headers.

to_der() → bytes

Build an encoded DER representation of the instance.

class pySim.esim.saip.ProfileElementTelecom(*decoded: dict | None = None*, ***kwargs*)

Class representing the ProfileElement for DF.TELECOM

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE
- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

class pySim.esim.saip.ProfileElementUSIM(*decoded: dict | None = None*, ***kwargs*)

Class representing the ProfileElement for ADF.USIM Mandatory Files

Instantiate a new ProfileElement. This is usually either called with the 'decoded' argument after reading a SAIP-DER-encoded PE. Alternatively, when constructing a PE from scratch, decoded is None, and a minimal PE-Header is generated.

Parameters

- decoded – asn1tools-generated decoded structure for this PE

- mandated – Whether or not the PE-Header should contain the mandated attribute
- pe_sequence – back-reference to the PE-Sequence of which we're part of

class pySim.esim.saip.SecurityDomainKey(*key_version_number: int*, *key_id: int*, *key_usage_qualifier: dict*, *key_components: List[SecurityDomainKeyComponent]*, *key_access: int = 0*, *key_counter_value: int | None = None*)

Representation of a key used for SCP access to a security domain.

static counter_len_for_kvn(*kvn: int*) → int

Return the counter length of the protocol a KVN is reserved for (see the KVN table in pySim.global_platform.uicc). Raises ValueError if that is not SCP80, SCP02 or SCP03.

classmethod from_saip_dict(*saip: dict*) → *SecurityDomainKey*

Construct instance from the dict as generated by SAIP asn.1 decoder.

to_saip_dict() → dict

Express instance in the dict format required by SAIP asn.1 encoder.

class pySim.esim.saip.SecurityDomainKeyComponent(*key_type: str*, *key_data: bytes*, *mac_length: int = 8*)

Representation of a key-component of a key for a security domain.

classmethod from_saip_dict(*saip: dict*) → *SecurityDomainKeyComponent*

Construct instance from the dict as generated by SAIP asn.1 decoder.

to_saip_dict() → dict

Express instance in the dict format required by SAIP asn.1 encoder.

pySim.esim.saip.bertlv_first_segment(*binary: bytes*) → Tuple[bytes, bytes]

obtain the first segment of a binary concatenation of BER-TLV objects. Returns: tuple of first TLV and remainder.

pySim.esim.saip.oid

Implementation of SimAlliance/TCA Interoperable Profile OIDs

class pySim.esim.saip.oid.eOID(*initializer*)

OID helper for TCA eUICC prefix

pySim.esim.saip.personalization

Implementation of Personalization of eSIM profiles in SimAlliance/TCA Interoperable Profile.

class pySim.esim.saip.personalization.Adm1(*input_value=None*)

class pySim.esim.saip.personalization.Adm2(*input_value=None*)

class pySim.esim.saip.personalization.AlgoConfig(*input_value=None*)

classmethod apply_val(*pes: ProfileElementSequence*, *val*)

This is what subclasses implement: store a value in a decoded profile package. Write the given val in the right format in all the right places in pes.

classmethod get_values_from_pes(*pes: ProfileElementSequence*)

This is what subclasses implement: yield all values from a decoded profile package. Find all values in the pes, and yield them decoded to a valid cls.input_value format. Should be a generator function, i.e. use 'yield' instead of 'return'.

Yielded value must be a dict(). Usually, an implementation will return only one key, like

{ "ICCID": "12345678901234567 89" }

Some implementations have more than one value to return, like

{ “IMSI”: “00101012345678”, “IMSI-ACC” : “5” }

Implementation example:

for pe in pes:

if my_condition(pe):
yield { cls.name: b2h(my_bin_value_from(pe)) }

class pySim.esim.saip.personalization.AlgorithmID(*input_value=None*)

use validate_val() from EnumParam, and apply_val() from AlgoConfig. In get_values_from_pes(), return enum value names, not raw values.

default_source

alias of ConstantSource

classmethod get_values_from_pes(*pes:* ProfileElementSequence)

This is what subclasses implement: yield all values from a decoded profile package. Find all values in the pes, and yield them decoded to a valid cls.input_value format. Should be a generator function, i.e. use ‘yield’ instead of ‘return’.

Yielded value must be a dict(). Usually, an implementation will return only one key, like

{ “ICCID”: “12345678901234567 89” }

Some implementations have more than one value to return, like

{ “IMSI”: “00101012345678”, “IMSI-ACC” : “5” }

Implementation example:

for pe in pes:

if my_condition(pe):
yield { cls.name: b2h(my_bin_value_from(pe)) }

class pySim.esim.saip.personalization.BinaryParam(*input_value=None*)

default_source

alias of RandomHexDigitSource

classmethod get_typical_input_len()

return a good length to use as the visible width of a user interface input field. May be overridden by subclasses. This default implementation returns the maximum allowed value length – a good fit for most subclasses.

classmethod validate_val(*val*)

This is a default implementation, with the behavior configured by subclasses’ allow_types…max_len settings. subclasses may override this function: Validate the contents of val, and raise ValueError on validation errors. Return a sanitized version of val, that is ready for cls.apply_val().

class pySim.esim.saip.personalization.ClassVarMeta(*name*, *bases*, *namespace*, ***kwargs*)

Metaclass that puts all additional keyword-args into the class. We use this to have one class definition for something like a PIN, and then have derived classes for PIN1, PIN2, …

class pySim.esim.saip.personalization.ConfigurableParameter(*input_value=None*)

Base class representing a part of the eSIM profile that is configurable during the personalization process (with dynamic data from elsewhere).

This class is abstract, you will only use subclasses in practice.

Subclasses have to implement the apply_val() classmethods, and may choose to override the default validate_val() implementation. The default validate_val() is a generic validator that uses the following class members (defined in subclasses) to configure the validation; if any of them is None, it means that the particular validation is skipped:

allow_types: a list of types permitted as argument to validate_val(); allow_types = (bytes, str,) allow_chars: if val is a str, accept only these characters; allow_chars = "0123456789" strip_chars: if val is a str, remove these characters; strip_chars = ' trn' min_len: minimum length of an input str; min_len = 4 max_len: maximum length of an input str; max_len = 8 allow_len: permit only specific lengths; allow_len = (8, 16, 32) numeric_base: indicate hex / decimal, if any; numeric_base = None; numeric_base = 10; numeric_base = 16

Subclasses may change the meaning of these by overriding validate_val(), for example that the length counts resulting bytes instead of a hexstring length. Most subclasses will be covered by the default validate_val().

Usage examples, by example of Iccid:

1) use a ConfigurableParameter instance, with .input_value and .value state:

```
iccid = Iccid()
try:
  iccid.input_value = '123456789012345678'
  iccid.validate()
except ValueError:
  print(f"failed to validate {iccid.name} == {iccid.input_value}")

pes = ProfileElementSequence.from_der(der_data_from_file)
try:
  iccid.apply(pes)
except ValueError:
  print(f"failed to apply {iccid.name} := {iccid.input_value}")

changed_der = pes.to_der()
```

2) use a ConfigurableParameter class, without state:

```
cls = Iccid
input_val = '123456789012345678'

try:
  clean_val = cls.validate_val(input_val)
except ValueError:
  print(f"failed to validate {cls.get_name()} = {input_val}")

pes = ProfileElementSequence.from_der(der_data_from_file)
try:
  cls.apply_val(pes, clean_val)
except ValueError:
  print(f"failed to apply {cls.get_name()} = {input_val}")

changed_der = pes.to_der()
```

apply(*pes:* ProfileElementSequence)

Place self.value into the ProfileElementSequence at the right place. If self.value is None, this implicitly calls self.validate() first, to generate a sanitized self.value from self.input_value. To override apply() in a subclass, rather override the classmethod apply_val().

classmethod apply_val(*pes:* ProfileElementSequence, *val*)

This is what subclasses implement: store a value in a decoded profile package. Write the given val in the right format in all the right places in pes.

classmethod get_len_range()

considering all of min_len, max_len and allow_len, get a tuple of the resulting (min, max) of permitted value length. For example, if an input value is an int, which needs to be represented with a minimum nr of digits, this function is useful to easily get that minimum permitted length.

classmethod get_name()

Return cls.name when it is set, otherwise return the python class name converted from 'CamelCase' to 'snake_case'. When using class *instances*, you can just use my_instance.name. When using *classes*, cls.get_name() returns the same name a class instance would have.

classmethod get_typical_input_len()

return a good length to use as the visible width of a user interface input field. May be overridden by subclasses. This default implementation returns the maximum allowed value length – a good fit for most subclasses.

classmethod get_value_from_pes(*pes:* ProfileElementSequence)

Same as get_values_from_pes() but expecting a single value. get_values_from_pes() may return values like this:

[{ 'AlgorithmID': 'Milenage' }, { 'AlgorithmID': 'Milenage' }]

This ensures that all these entries are identical and would return only
{ 'AlgorithmID': 'Milenage' }.

This is relevant for any profile element that may appear multiple times in the same PES (only a few), where each occurrence should reflect the same value (all currently known parameters).

abstractmethod classmethod get_values_from_pes(*pes:* ProfileElementSequence) → Generator

This is what subclasses implement: yield all values from a decoded profile package. Find all values in the pes, and yield them decoded to a valid cls.input_value format. Should be a generator function, i.e. use 'yield' instead of 'return'.

Yielded value must be a dict(). Usually, an implementation will return only one key, like

{ "ICCID": "1234567890123456789" }

Some implementations have more than one value to return, like

{ "IMSI": "00101012345678", "IMSI-ACC" : "5" }

Implementation example:

```
for pe in pes:
    if my_condition(pe):
        yield { cls.name: b2h(my_bin_value_from(pe)) }
```

validate()

Validate self.input_value and place the result in self.value. This is also called implicitly by apply(), if self.value is still None. To override validation in a subclass, rather re-implement the classmethod validate_val().

classmethod validate_val(*val*)

This is a default implementation, with the behavior configured by subclasses' allow_types…max_len settings. subclasses may override this function: Validate the contents of val, and raise ValueError on validation errors. Return a sanitized version of val, that is ready for cls.apply_val().

class pySim.esim.saip.personalization.DecimalHexParam(*input_value=None*)

The input value is decimal digits. The decimal value is stored such that each hexadecimal digit represents one decimal digit, useful for various PIN type parameters.

Optionally, the value is stored with padding, for example: rpad = 8 would store '123' as '123fffff'. This is also common in PIN type parameters.

classmethod decimal_hex_to_str(*val*)

useful for get_values_from_pes() implementations of subclasses

classmethod validate_val(*val*)

This is a default implementation, with the behavior configured by subclasses' allow_types... max_len settings. subclasses may override this function: Validate the contents of val, and raise ValueError on validation errors. Return a sanitized version of val, that is ready for cls.apply_val().

class pySim.esim.saip.personalization.DecimalParam(*input_value=None*)

Decimal digits. The input value may be a string of decimal digits like '012345', or an int. The output of validate_val() is a string with only decimal digits 0-9, in the required length with leading zeros if necessary.

classmethod validate_val(*val*)

This is a default implementation, with the behavior configured by subclasses' allow_types... max_len settings. subclasses may override this function: Validate the contents of val, and raise ValueError on validation errors. Return a sanitized version of val, that is ready for cls.apply_val().

class pySim.esim.saip.personalization.EfUstServiceParam(*input_value=None*)

superclass for EF-UST service flag parameters

classmethod apply_val(*pes:* ProfileElementSequence, *val*)

This is what subclasses implement: store a value in a decoded profile package. Write the given val in the right format in all the right places in pes.

default_source

alias of ConstantSource

classmethod get_values_from_pes(*pes:* ProfileElementSequence)

This is what subclasses implement: yield all values from a decoded profile package. Find all values in the pes, and yield them decoded to a valid cls.input_value format. Should be a generator function, i.e. use 'yield' instead of 'return'.

Yielded value must be a dict(). Usually, an implementation will return only one key, like

{ "ICCID": "12345678901234567890" }

Some implementations have more than one value to return, like

{ "IMSI": "00101012345678", "IMSI-ACC" : "5" }

Implementation example:

for pe in pes:

if my_condition(pe):
yield { cls.name: b2h(my_bin_value_from(pe)) }

class pySim.esim.saip.personalization.EnumParam(*input_value=None*)

ConfigurableParameter for named value enumerations.

Subclasses define an own value_map, and implement their own apply_val() and get_values_from_pes().

classmethod validate_val(*val*)

This is a default implementation, with the behavior configured by subclasses' allow_types…max_len settings. subclasses may override this function: Validate the contents of val, and raise ValueError on validation errors. Return a sanitized version of val, that is ready for cls.apply_val().

class pySim.esim.saip.personalization.EuiccMandatoryServiceGetIdentity(*input_value=None*)

eUICC Mandatory Services: get-identity. The eUICC must be capable of providing a 5G identity using SUCI-CalcInfo located in the USIM's DF-SAIP, see parameter 5G-SUCI-CalcInfo-USIM.

class pySim.esim.saip.personalization.EuiccMandatoryServiceParam(*input_value=None*)

superclass for managing items of the ProfileHeader / eUICC-Mandatory-services ServicesList

classmethod apply_val(*pes:* ProfileElementSequence, *val*)

This is what subclasses implement: store a value in a decoded profile package. Write the given val in the right format in all the right places in pes.

default_source

alias of ConstantSource

classmethod get_values_from_pes(*pes:* ProfileElementSequence)

This is what subclasses implement: yield all values from a decoded profile package. Find all values in the pes, and yield them decoded to a valid cls.input_value format. Should be a generator function, i.e. use 'yield' instead of 'return'.

Yielded value must be a dict(). Usually, an implementation will return only one key, like

{ "ICCID": "12345678901234567 89" }

Some implementations have more than one value to return, like

{ "IMSI": "00101012345678", "IMSI-ACC" : "5" }

Implementation example:

```
for pe in pes:
    if my_condition(pe):
        yield { cls.name: b2h(my_bin_value_from(pe)) }
```

class pySim.esim.saip.personalization.EuiccMandatoryServiceProfileA(*input_value=None*)

eUICC Mandatory Services: profile-a-x25519. The eUICC must be able to estblish a 5G identity using an X25519 key, as provided in a profile-A ("identifier": 1) key in SUCI-CalcInfo located in the USIM's DF-SAIP, see parameter 5G-SUCI-CalcInfo-USIM.

class pySim.esim.saip.personalization.EuiccMandatoryServiceProfileB(*input_value=None*)

eUICC Mandatory Services: profile-b-p256. The eUICC must be able to estblish a 5G identity using a P256 key, as provided in a profile-B ("identifier": 2) key in SUCI-CalcInfo located in the USIM's DF-SAIP, see parameter 5G-SUCI-CalcInfo-USIM.

class pySim.esim.saip.personalization.Iccid(*input_value=None*)

ICCID Parameter. Input: string of decimal digits. If the string of digits is only 18 digits long, add a Luhn check digit.

classmethod apply_val(*pes:* ProfileElementSequence, *val*)

This is what subclasses implement: store a value in a decoded profile package. Write the given val in the right format in all the right places in pes.

default_source

alias of IncDigitSource

classmethod get_values_from_pes(*pes:* ProfileElementSequence)

This is what subclasses implement: yield all values from a decoded profile package. Find all values in the pes, and yield them decoded to a valid cls.input_value format. Should be a generator function, i.e. use 'yield' instead of 'return'.

Yielded value must be a dict(). Usually, an implementation will return only one key, like

{ "ICCID": "1234567890123456789" }

Some implementations have more than one value to return, like

{ "IMSI": "00101012345678", "IMSI-ACC" : "5" }

Implementation example:

```
for pe in pes:
    if my_condition(pe):
        yield { cls.name: b2h(my_bin_value_from(pe)) }
```

classmethod validate_val(*val*)

This is a default implementation, with the behavior configured by subclasses' allow_types…max_len settings. subclasses may override this function: Validate the contents of val, and raise ValueError on validation errors. Return a sanitized version of val, that is ready for cls.apply_val().

class pySim.esim.saip.personalization.Imsi(*input_value=None*)

Configurable IMSI. Expects value to be a string of digits. Automatically sets the ACC to the last digit of the IMSI.

classmethod apply_val(*pes:* ProfileElementSequence, *val*)

This is what subclasses implement: store a value in a decoded profile package. Write the given val in the right format in all the right places in pes.

default_source

alias of IncDigitSource

classmethod get_values_from_pes(*pes:* ProfileElementSequence)

This is what subclasses implement: yield all values from a decoded profile package. Find all values in the pes, and yield them decoded to a valid cls.input_value format. Should be a generator function, i.e. use 'yield' instead of 'return'.

Yielded value must be a dict(). Usually, an implementation will return only one key, like

{ "ICCID": "1234567890123456789" }

Some implementations have more than one value to return, like

{ "IMSI": "00101012345678", "IMSI-ACC" : "5" }

Implementation example:

```
for pe in pes:
    if my_condition(pe):
        yield { cls.name: b2h(my_bin_value_from(pe)) }
```

class pySim.esim.saip.personalization.IntegerParam(*input_value=None*)

classmethod get_values_from_pes(*pes:* ProfileElementSequence)

This is what subclasses implement: yield all values from a decoded profile package. Find all values in the pes, and yield them decoded to a valid cls.input_value format. Should be a generator function, i.e. use 'yield' instead of 'return'.

Yielded value must be a dict(). Usually, an implementation will return only one key, like

{ “ICCID”: “12345678901234567890” }

Some implementations have more than one value to return, like

{ “IMSI”: “001010123456789”, “IMSI-ACC” : “5” }

Implementation example:

for pe in pes:

if my_condition(pe):
yield { cls.name: b2h(my_bin_value_from(pe)) }

classmethod validate_val(*val*)

This is a default implementation, with the behavior configured by subclasses’ allow_types…max_len settings. subclasses may override this function: Validate the contents of val, and raise ValueError on validation errors. Return a sanitized version of val, that is ready for cls.apply_val().

class pySim.esim.saip.personalization.K(*input_value=None*)

use validate_val() from BinaryParam, and apply_val() from AlgoConfig

class pySim.esim.saip.personalization.MilenageRotationConstants(*input_value=None*)

rotation constants r1,r2,r3,r4,r5 of Milenage, Range 0..127. See 3GPP TS 35.206 Sections 2.3 + 5.3. Provided as octet-string concatenation of all 5 constants. Expects a bytes-like object of length 5, with each byte in the range of 0..127. The default value by 3GPP is ‘4000204060’ (hex notation)

default_source

alias of ConstantSource

classmethod validate_val(*val*)

allow_len checks the length, this in addition checks the value range

class pySim.esim.saip.personalization.MilenageXoringConstants(*input_value=None*)

XOR-ing constants c1,c2,c3,c4,c5 of Milenage, 128bit each. See 3GPP TS 35.206 Sections 2.3 + 5.3. Provided as octet-string concatenation of all 5 constants. The default value by 3GPP is the concatenation of:

```
00000000000000000000000000000000
00000000000000000000000000000001
00000000000000000000000000000002
00000000000000000000000000000004
00000000000000000000000000000008
```

default_source

alias of ConstantSource

class pySim.esim.saip.personalization.MncLen(*input_value=None*)

MNC length. Must be either 2 or 3. Sets only the MNC length field in EF-AD (Administrative Data).

classmethod apply_val(*pes:* ProfileElementSequence, *val*)

val must be an int: either 2 or 3

default_source

alias of ConstantSource

classmethod get_values_from_pes(*pes:* ProfileElementSequence)

This is what subclasses implement: yield all values from a decoded profile package. Find all values in the pes, and yield them decoded to a valid cls.input_value format. Should be a generator function, i.e. use ‘yield’ instead of ‘return’.

Yielded value must be a dict(). Usually, an implementation will return only one key, like

{ “ICCID”: “12345678901234567 89” }

Some implementations have more than one value to return, like

{ “IMSI”: “00101012345678”, “IMSI-ACC” : “5” }

Implementation example:

for pe in pes:

if my_condition(pe):
yield { cls.name: b2h(my_bin_value_from(pe)) }

class pySim.esim.saip.personalization.Opc(*input_value=None*)

class pySim.esim.saip.personalization.Pin(*input_value=None*)

Configurable PIN (Personal Identification Number). String of digits.

classmethod apply_val(*pes:* ProfileElementSequence, *val*)

This is what subclasses implement: store a value in a decoded profile package. Write the given val in the right format in all the right places in pes.

default_source

alias of RandomDigitSource

classmethod get_values_from_pes(*pes:* ProfileElementSequence)

This is what subclasses implement: yield all values from a decoded profile package. Find all values in the pes, and yield them decoded to a valid cls.input_value format. Should be a generator function, i.e. use ‘yield’ instead of ‘return’.

Yielded value must be a dict(). Usually, an implementation will return only one key, like

{ “ICCID”: “12345678901234567 89” }

Some implementations have more than one value to return, like

{ “IMSI”: “00101012345678”, “IMSI-ACC” : “5” }

Implementation example:

for pe in pes:

if my_condition(pe):
yield { cls.name: b2h(my_bin_value_from(pe)) }

class pySim.esim.saip.personalization.Pin1(*input_value=None*)

class pySim.esim.saip.personalization.Pin2(*input_value=None*)

classmethod apply_val(*pes:* ProfileElementSequence, *val*)

This is what subclasses implement: store a value in a decoded profile package. Write the given val in the right format in all the right places in pes.

classmethod get_values_from_pes(*pes:* ProfileElementSequence)

This is what subclasses implement: yield all values from a decoded profile package. Find all values in the pes, and yield them decoded to a valid cls.input_value format. Should be a generator function, i.e. use ‘yield’ instead of ‘return’.

Yielded value must be a dict(). Usually, an implementation will return only one key, like

{ “ICCID”: “12345678901234567 89” }

Some implementations have more than one value to return, like

{ “IMSI”: “00101012345678”, “IMSI-ACC” : “5” }

Implementation example:

for pe in pes:

if my_condition(pe):
yield { cls.name: b2h(my_bin_value_from(pe)) }

class pySim.esim.saip.personalization.Puk(*input_value=None*)

Configurable PUK (Pin Unblock Code). String ASCII-encoded digits.

classmethod apply_val(*pes:* ProfileElementSequence, *val*)

This is what subclasses implement: store a value in a decoded profile package. Write the given val in the right format in all the right places in pes.

default_source

alias of RandomDigitSource

classmethod get_values_from_pes(*pes:* ProfileElementSequence)

This is what subclasses implement: yield all values from a decoded profile package. Find all values in the pes, and yield them decoded to a valid cls.input_value format. Should be a generator function, i.e. use ‘yield’ instead of ‘return’.

Yielded value must be a dict(). Usually, an implementation will return only one key, like

{ “ICCID”: “12345678901234567 89” }

Some implementations have more than one value to return, like

{ “IMSI”: “00101012345678”, “IMSI-ACC” : “5” }

Implementation example:

for pe in pes:

if my_condition(pe):
yield { cls.name: b2h(my_bin_value_from(pe)) }

class pySim.esim.saip.personalization.Puk1(*input_value=None*)

class pySim.esim.saip.personalization.Puk2(*input_value=None*)

class pySim.esim.saip.personalization.SdKey(*input_value=None*)

Configurable Security Domain (SD) Key. Value is presented as bytes. Non-abstract implementations are generated in SdKey.generate_sd_key_classes

classmethod apply_val(*pes:* ProfileElementSequence, *val*)

This is what subclasses implement: store a value in a decoded profile package. Write the given val in the right format in all the right places in pes.

classmethod generate_sd_key_classes(*sd_key_defs=(('SCP02', (32, 33, 34, 255), (('AES', {'allow_len': (16, 24, 32), 'key_type': EnumIntegerString.new(136, 'aes')}),), (('ENC', {'key_id': 1, 'key_usage_qual': 24}), ('MAC', {'key_id': 2, 'key_usage_qual': 20}), ('DEK', {'key_id': 3, 'key_usage_qual': 72}))), ('SCP03', (48, 49, 50), (('AES', {'allow_len': (16, 24, 32), 'key_type': EnumIntegerString.new(136, 'aes')}),), (('ENC', {'key_id': 1, 'key_usage_qual': 24}), ('MAC', {'key_id': 2, 'key_usage_qual': 20}), ('DEK', {'key_id': 3, 'key_usage_qual': 72}))), ('SCP80', (1, 2, 3), (('DES', {'allow_len': (16,), 'key_type': EnumIntegerString.new(128, 'des')}), ('AES', {'allow_len': (16, 24, 32), 'key_type': EnumIntegerString.new(136, 'aes')}))), (('ENC', {'key_id': 1, 'key_usage_qual': 24}), ('MAC', {'key_id': 2, 'key_usage_qual': 20}), ('DEK', {'key_id': 3, 'key_usage_qual': 72}))), ('SCP81', (64, 65, 66), (('TLSPSK', {'allow_len': (16, 24, 32), 'key_id': 1, 'key_type': EnumIntegerString.new(133, 'tls_psk'), 'key_usage_qual': 60}),)), ('SCP81', (64, 65, 66), (('DES', {'allow_len': (16,), 'key_type': EnumIntegerString.new(128, 'des')}), ('AES', {'allow_len': (16, 24, 32), 'key_type': EnumIntegerString.new(136, 'aes')})), (('DEK', {'key_id': 2, 'key_usage_qual': 72}),))))*)

This generates python classes to be exported in this module, as subclasses of class SdKey.

We create SdKey subclasses dynamically from a list. You can list all of them via:

from pySim.esim.saip.personalization import SdKey SdKey.all_implementations

or

print('

'.join(sorted(f'{x.__name__} {x.name}' for x in SdKey.all_implementations)))

at time of writing this comment, this prints:

SdKeyScp02Kvn20AesDek SCP02-KVN20-AES-DEK SdKeyScp02Kvn20AesEnc SCP02-KVN20-AES-ENC SdKeyScp02Kvn20AesMac SCP02-KVN20-AES-MAC SdKeyScp02Kvn21AesDek SCP02-KVN21-AES-DEK SdKeyScp02Kvn21AesEnc SCP02-KVN21-AES-ENC SdKeyScp02Kvn21AesMac SCP02-KVN21-AES-MAC SdKeyScp02Kvn22AesDek SCP02-KVN22-AES-DEK SdKeyScp02Kvn22AesEnc SCP02-KVN22-AES-ENC SdKeyScp02Kvn22AesMac SCP02-KVN22-AES-MAC SdKeyScp02KvnffAesDek SCP02-KVNff-AES-DEK SdKeyScp02KvnffAesEnc SCP02-KVNff-AES-ENC SdKeyScp02KvnffAesMac SCP02-KVNff-AES-MAC SdKeyScp03Kvn30AesDek SCP03-KVN30-AES-DEK SdKeyScp03Kvn30AesEnc SCP03-KVN30-AES-ENC SdKeyScp03Kvn30AesMac SCP03-KVN30-AES-MAC SdKeyScp03Kvn31AesDek SCP03-KVN31-AES-DEK SdKeyScp03Kvn31AesEnc SCP03-KVN31-AES-ENC SdKeyScp03Kvn31AesMac SCP03-KVN31-AES-MAC SdKeyScp03Kvn32AesDek SCP03-KVN32-AES-DEK SdKeyScp03Kvn32AesEnc SCP03-KVN32-AES-ENC SdKeyScp03Kvn32AesMac SCP03-KVN32-AES-MAC SdKeyScp80Kvn01AesDek SCP80-KVN01-AES-DEK SdKeyScp80Kvn01AesEnc SCP80-KVN01-AES-ENC SdKeyScp80Kvn01AesMac SCP80-KVN01-AES-MAC SdKeyScp80Kvn01DesDek SCP80-KVN01-DES-DEK SdKeyScp80Kvn01DesEnc SCP80-KVN01-DES-ENC SdKeyScp80Kvn01DesMac SCP80-KVN01-DES-MAC SdKeyScp80Kvn02AesDek SCP80-KVN02-AES-DEK SdKeyScp80Kvn02AesEnc SCP80-KVN02-AES-ENC SdKeyScp80Kvn02AesMac SCP80-KVN02-AES-MAC SdKeyScp80Kvn02DesDek SCP80-KVN02-DES-DEK SdKeyScp80Kvn02DesEnc SCP80-KVN02-DES-ENC SdKeyScp80Kvn02DesMac SCP80-KVN02-DES-MAC SdKeyScp80Kvn03AesDek SCP80-KVN03-AES-DEK SdKeyScp80Kvn03AesEnc SCP80-KVN03-AES-ENC

SdKeyScp80Kvn03AesMac SCP80-KVN03-AES-MAC SdKeyScp80Kvn03DesDek SCP80-KVN03-DES-DEK SdKeyScp80Kvn03DesEnc SCP80-KVN03-DES-ENC SdKeyScp80Kvn03DesMac SCP80-KVN03-DES-MAC SdKeyScp81Kvn40AesDek SCP81-KVN40-AES-DEK SdKeyScp81Kvn40DesDek SCP81-KVN40-DES-DEK SdKeyScp81Kvn40Tlspsk SCP81-KVN40-TLSPSK SdKeyScp81Kvn41AesDek SCP81-KVN41-AES-DEK SdKeyScp81Kvn41DesDek SCP81-KVN41-DES-DEK SdKeyScp81Kvn41Tlspsk SCP81-KVN41-TLSPSK SdKeyScp81Kvn42AesDek SCP81-KVN42-AES-DEK SdKeyScp81Kvn42DesDek SCP81-KVN42-DES-DEK SdKeyScp81Kvn42Tlspsk SCP81-KVN42-TLSPSK

classmethod get_values_from_pes(*pes:* ProfileElementSequence)

This is what subclasses implement: yield all values from a decoded profile package. Find all values in the pes, and yield them decoded to a valid cls.input_value format. Should be a generator function, i.e. use 'yield' instead of 'return'.

Yielded value must be a dict(). Usually, an implementation will return only one key, like

{ "ICCID": "12345678901234567 89" }

Some implementations have more than one value to return, like

{ "IMSI": "00101012345678", "IMSI-ACC" : "5" }

Implementation example:

for pe in pes:

if my_condition(pe):
yield { cls.name: b2h(my_bin_value_from(pe)) }

class pySim.esim.saip.personalization.SdKeyScp02Kvn20AesDek(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp02Kvn20AesEnc(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp02Kvn20AesMac(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp02Kvn21AesDek(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp02Kvn21AesEnc(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp02Kvn21AesMac(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp02Kvn22AesDek(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp02Kvn22AesEnc(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp02Kvn22AesMac(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp02KvnffAesDek(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp02KvnffAesEnc(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp02KvnffAesMac(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp03Kvn30AesDek(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp03Kvn30AesEnc(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp03Kvn30AesMac(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp03Kvn31AesDek(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp03Kvn31AesEnc(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp03Kvn31AesMac(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp03Kvn32AesDek(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp03Kvn32AesEnc(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp03Kvn32AesMac(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp80Kvn01AesDek(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp80Kvn01AesEnc(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp80Kvn01AesMac(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp80Kvn01DesDek(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp80Kvn01DesEnc(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp80Kvn01DesMac(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp80Kvn02AesDek(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp80Kvn02AesEnc(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp80Kvn02AesMac(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp80Kvn02DesDek(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp80Kvn02DesEnc(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp80Kvn02DesMac(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp80Kvn03AesDek(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp80Kvn03AesEnc(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp80Kvn03AesMac(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp80Kvn03DesDek(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp80Kvn03DesEnc(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp80Kvn03DesMac(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp81Kvn40AesDek(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp81Kvn40DesDek(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp81Kvn40Tlspsk(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp81Kvn41AesDek(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp81Kvn41DesDek(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp81Kvn41Tlspsk(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp81Kvn42AesDek(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp81Kvn42DesDek(*input_value=None*)

class pySim.esim.saip.personalization.SdKeyScp81Kvn42Tlspsk(*input_value=None*)

class pySim.esim.saip.personalization.SmspTpScAddr(*input_value=None*)

Configurable SMSC (SMS Service Centre) TP-SC-ADDR. Expects to be a phone number in national or international format (designated by a leading +). Automatically sets the NPI to E.164 and the TON based on presence or absence of leading +.

classmethod apply_val(*pes:* ProfileElementSequence, *val*)

val must be a tuple (international[bool], digits[str]). For example, an input of "+1234" corresponds to (True, "1234"); An input of "1234" corresponds to (False, "1234").

default_source

alias of ConstantSource

classmethod get_values_from_pes(*pes:* ProfileElementSequence)

This is what subclasses implement: yield all values from a decoded profile package. Find all values in the pes, and yield them decoded to a valid cls.input_value format. Should be a generator function, i.e. use 'yield' instead of 'return'.

Yielded value must be a dict(). Usually, an implementation will return only one key, like

{ "ICCID": "12345678901234567 89" }

Some implementations have more than one value to return, like

{ "IMSI": "001010123456 78", "IMSI-ACC" : "5" }

Implementation example:

for pe in pes:

if my_condition(pe):
yield { cls.name: b2h(my_bin_value_from(pe)) }

classmethod validate_val(*val*)

This is a default implementation, with the behavior configured by subclasses' allow_types…max_len settings. subclasses may override this function: Validate the contents of val, and raise ValueError on validation errors. Return a sanitized version of val, that is ready for cls.apply_val().

class pySim.esim.saip.personalization.SuciActive(*input_value=None*)

EF-UST service nr 124: enable or disable the SUCI service.

class pySim.esim.saip.personalization.SuciCalcInfoParameter(*input_value=None*)

SUCI Calculation Information as in section 4.4.11.8 of 3GPP TS 31.102

classmethod apply_val(*pes:* ProfileElementSequence, *val*)

This is what subclasses implement: store a value in a decoded profile package. Write the given val in the right format in all the right places in pes.

default_source

alias of ConstantSource

classmethod get_values_from_pes(*pes:* ProfileElementSequence)

This is what subclasses implement: yield all values from a decoded profile package. Find all values in the pes, and yield them decoded to a valid cls.input_value format. Should be a generator function, i.e. use 'yield' instead of 'return'.

Yielded value must be a dict(). Usually, an implementation will return only one key, like

{ "ICCID": "12345678901234567 89" }

Some implementations have more than one value to return, like

{ “IMSI”: “00101012345678”, “IMSI-ACC” : “5” }

Implementation example:

for pe in pes:

if my_condition(pe):
yield { cls.name: b2h(my_bin_value_from(pe)) }

classmethod validate_val(*val*)

This is a default implementation, with the behavior configured by subclasses’ allow_types…max_len settings. subclasses may override this function: Validate the contents of val, and raise ValueError on validation errors. Return a sanitized version of val, that is ready for cls.apply_val().

class pySim.esim.saip.personalization.SuciCalcInfoUe(*input_value=None*)

SUCI Calculation Information as in section 4.4.11.8 of 3GPP TS 31.102, readable by UE (DF-5GS)

class pySim.esim.saip.personalization.SuciCalcInfoUsim(*input_value=None*)

SUCI Calculation Information as in section 4.4.11.8 of 3GPP TS 31.102, readable only by USIM (DF-SAIP)

class pySim.esim.saip.personalization.SuciInUsim(*input_value=None*)

EF-UST service nr 125: calculate SUCI in UE or in USIM

class pySim.esim.saip.personalization.SuciRi(*input_value=None*)

SUCI Routing Indicator as in section 4.4.11.11 of 3GPP TS 31.102

classmethod apply_val(*pes:* ProfileElementSequence, *val*)

This is what subclasses implement: store a value in a decoded profile package. Write the given val in the right format in all the right places in pes.

default_source

alias of ConstantSource

classmethod get_values_from_pes(*pes:* ProfileElementSequence)

This is what subclasses implement: yield all values from a decoded profile package. Find all values in the pes, and yield them decoded to a valid cls.input_value format. Should be a generator function, i.e. use ‘yield’ instead of ‘return’.

Yielded value must be a dict(). Usually, an implementation will return only one key, like

{ “ICCID”: “12345678901234567 89” }

Some implementations have more than one value to return, like

{ “IMSI”: “00101012345678”, “IMSI-ACC” : “5” }

Implementation example:

for pe in pes:

if my_condition(pe):
yield { cls.name: b2h(my_bin_value_from(pe)) }

class pySim.esim.saip.personalization.TuakNumberOfKeccak(*input_value=None*)

Number of iterations of Keccak-f[1600] permutation as recomended by Section 7.2 of 3GPP TS 35.231

default_source

alias of ConstantSource

pySim.esim.saip.personalization.file_replace_content(*file: List[Tuple]*, *new_content: bytes*)

Completely replace all fillFileContent of a decoded 'File' with the new_content.

pySim.esim.saip.personalization.remove_unwanted_tuples_from_list(*l: List[Tuple]*, *unwanted_keys: List[str]*) → List[Tuple]

In a list of tuples, remove all tuples whose first part equals 'unwanted_key'.

pySim.esim.saip.templates

Implementation of SimAlliance/TCA Interoperable Profile Templates.

class pySim.esim.saip.templates.FileTemplate(*fid: int*, *name: str*, *ftype*, *nb_rec: int | None*, *size: int | None*, *arr: int*, *sfi: int | None = None*, *default_val: str | None = None*, *content_rqd: bool = True*, *params: List | None = None*, *ass_serv: List[int] | None = None*, *high_update: bool = False*, *pe_name: str | None = None*, *repeat: bool = False*, *ppath: List[int] = []*)

Representation of a single file in a SimAlliance/TCA Profile Template. The argument order is done to match that of the tables in Section 9 of the SAIP specification.

Parameters

- fid – The 16bit file-identifier of the file
- name – The name of the file in human-readable "EF.FOO", "DF.BAR" notation
- ftype – The type of the file; can be 'MF', 'ADF', 'DF', 'TR', 'LF', 'CY', 'BT'
- nb_rec – Then number of records (only valid for 'LF' and 'CY')
- size – The size of the file ('TR', 'BT'); size of each record ('LF, 'CY')
- arr – The record number of EF.ARR for referenced access rules
- sfi – The short file identifier, if any
- default_val – The default value [pattern] of the file
- content_rqd – Whether an instance of template *must* specify file contents
- params – A list of parameters that an instance of the template *must* specify
- ass_serv – The associated service[s] of the service table
- high_update – Is this file of "high update frequency" type?
- pe_name – The name of this file in the ASN.1 type of the PE. Auto-generated for most.
- repeat – Whether the default_val pattern is a repeating pattern.
- ppath – The intermediate path between the base_df of the ProfileTemplate and this file. If not specified, the file will be created immediately underneath the base_df.

expand_default_value_pattern(*length: int | None = None*) → bytes | None

Expand the default value pattern to the specified length.

The SAIP section 9 tables give a linear-fixed or cyclic file its default per record (EF ICI: 20 records of 38 bytes, default 'FF…FF0000000001FFFF'), which is what _default_value_len already assumes. A request longer than one record therefore expands one record and repeats it; expanding once across the file puts the tail of the default into the last record only.

get_file_by_path(*path: List[str]*) → *FileTemplate* | None

Return a FileTemplate matching the given path within this ProfileTemplate.

property path

Return the path of the given File within the hierarchy.

print_tree(*indent: str = ''*)

recursive printing of FileTemplate tree structure.

class pySim.esim.saip.templates.FilesAtMF

Files at MF as per Section 9.2

class pySim.esim.saip.templates.FilesCD

Files at DF.CD as per Section 9.3

class pySim.esim.saip.templates.FilesDf5GProSe

DF.ProSe Files at ADF.USIM as per Section 9.5.14

parent

alias of FilesUsimMandatory

class pySim.esim.saip.templates.FilesDfSnpn

DF.SNPN Files at ADF.USIM as per Section 9.5.13

parent

alias of FilesUsimMandatory

class pySim.esim.saip.templates.FilesEap

Files at DF.EAP as per Section 9.8

class pySim.esim.saip.templates.FilesIotMandatory

Mandatory Files of the IoT Minimal Profile (MF and ADF.USIM) as per Section 9.10.1 v3.4.1

class pySim.esim.saip.templates.FilesIotOptional

Optional Files of the IoT Minimal Profile below ADF.USIM as per Section 9.10.2 v3.4.1

extends

alias of FilesIotMandatory

class pySim.esim.saip.templates.FilesIsimMandatory

Mandatory Files at ADF.ISIM as per Section 9.6.1

class pySim.esim.saip.templates.FilesIsimOptional

Optional Files at ADF.ISIM as per Section 9.6.2 of v2.3.1

extends

alias of FilesIsimMandatory

class pySim.esim.saip.templates.FilesIsimOptionalv2

Optional Files at ADF.ISIM as per Section 9.6.2

extends

alias of FilesIsimMandatory

class pySim.esim.saip.templates.FilesTelecom

Files at DF.TELECOM as per Section 9.4 v2.3.1

class pySim.esim.saip.templates.FilesTelecomV2

Files at DF.TELECOM as per Section 9.4

class pySim.esim.saip.templates.FilesUsimDf5GS
DF.5GS Files at ADF.USIM as per Section 9.5.11 v2.3.1

parent
alias of FilesUsimMandatory

class pySim.esim.saip.templates.FilesUsimDf5GSv2
DF.5GS Files at ADF.USIM as per Section 9.5.11.2

parent
alias of FilesUsimMandatory

class pySim.esim.saip.templates.FilesUsimDf5GSv3
DF.5GS Files at ADF.USIM as per Section 9.5.11.3

parent
alias of FilesUsimMandatory

class pySim.esim.saip.templates.FilesUsimDf5GSv4
DF.5GS Files at ADF.USIM as per Section 9.5.11.4

parent
alias of FilesUsimMandatory

class pySim.esim.saip.templates.FilesUsimDfGsmAccess
DF.GSM-ACCESS Files at ADF.USIM as per Section 9.5.4

parent
alias of FilesUsimMandatory

class pySim.esim.saip.templates.FilesUsimDfPhonebook
DF.PHONEBOOK Files at ADF.USIM as per Section 9.5.3

class pySim.esim.saip.templates.FilesUsimDfSaip
DF.SAIP Files at ADF.USIM as per Section 9.5.12

parent
alias of FilesUsimMandatory

class pySim.esim.saip.templates.FilesUsimMandatory
Mandatory Files at ADF.USIM as per Section 9.5.1 v2.3.1

class pySim.esim.saip.templates.FilesUsimMandatoryV2
Mandatory Files at ADF.USIM as per Section 9.5.1

class pySim.esim.saip.templates.FilesUsimOptional
Optional Files at ADF.USIM as per Section 9.5.2 v2.3.1

extends
alias of FilesUsimMandatory

class pySim.esim.saip.templates.FilesUsimOptionalV2
Optional Files at ADF.USIM as per Section 9.5.2

extends
alias of FilesUsimMandatoryV2

class pySim.esim.saip.templates.FilesUsimOptionalV3

Optional Files at ADF.USIM as per Section 9.5.2.3 v3.3.1

extends

alias of FilesUsimMandatoryV2

class pySim.esim.saip.templates.ProfileTemplate

Representation of a SimAlliance/TCA Profile Template. Each Template is identified by its OID and consists of a number of file definitions. We implement each profile template as a class derived from this base class. Each such derived class is a singleton and has no instances.

classmethod base_df() → *FileTemplate*

Return the FileTemplate for the base DF of the given template. This may be a DF or ADF within this template, or refer to another template (e.g. mandatory USIM if we are optional USIM.

class pySim.esim.saip.templates.ProfileTemplateRegistry

A registry of profile templates. Exists as a singleton class with no instances and only classmethods.

classmethod add(*tpl:* ProfileTemplate)

Add a ProfileTemplate to the registry. There can only be one Template per OID.

classmethod get_by_oid(*oid: List[int]* | *str*) → *ProfileTemplate* | None

Look-up the ProfileTemplate based on its OID. The OID can be given either in dotted-string format, or as a list of integers.

class pySim.esim.saip.templates.SaipSpecVersion

Represents a specific version of the SIMalliance / TCA eUICC Profile Package: Interoperable Format Technical Specification.

static for_version(*req_version: str*) → *SaipSpecVersion* | None

Return the subclass for the requested version number string.

classmethod suports_template_OID(*OID: OID*) → bool

Return if a given spec version supports a template of given OID.

classmethod version_match(*ver: str*) → bool

Check if the given version-string matches the classes version. trailing zeroes are ignored, so that for example 2.2.0 will be considered equal to 2.2

class pySim.esim.saip.templates.SaipSpecVersion101

class pySim.esim.saip.templates.SaipSpecVersion20

class pySim.esim.saip.templates.SaipSpecVersion21

class pySim.esim.saip.templates.SaipSpecVersion22

class pySim.esim.saip.templates.SaipSpecVersion23

class pySim.esim.saip.templates.SaipSpecVersion231

class pySim.esim.saip.templates.SaipSpecVersion31

class pySim.esim.saip.templates.SaipSpecVersion32

class pySim.esim.saip.templates.SaipSpecVersion331

pySim.esim.saip.validation

Implementation of SimAlliance/TCA Interoperable Profile validation.

class pySim.esim.saip.validation.CheckBasicStructure

ProfileConstraintChecker for the basic profile structure constraints.

check_identification_unique(*pes:* ProfileElementSequence)

Ensure that each PE has a unique identification value.

check_mandatory_services(*pes:* ProfileElementSequence)

Ensure that the PE for the mandatory services exist.

check_mandatory_services_aka(*pes:* ProfileElementSequence)

Ensure that no unnecessary authentication related services are marked as mandatory but not actually used within the profile

check_number_of_occurrence(*pes:* ProfileElementSequence)

Check The number of occurrence of various ProfileElements.

check_optional_ordering(*pes:* ProfileElementSequence)

Check the ordering of optional PEs following the respective mandatory ones.

check_start_and_end(*pes:* ProfileElementSequence)

Check for mandatory header and end ProfileElements at the right position.

class pySim.esim.saip.validation.FileCheckBasicStructure

Validator for the basic structure of a decoded file.

check_forbidden(*l: List[Tuple]*)

Perform checks for forbidden parameters as described in Section 8.3.3.

check_seqence(*l: List[Tuple]*)

Check the sequence/ordering.

exception pySim.esim.saip.validation.FileError

Raised when a FileConstraintChecker finds an error in a file [structure].

class pySim.esim.saip.validation.ProfileConstraintChecker

Base class of a constraint checker for a ProfileElementSequence.

check(*pes:* ProfileElementSequence)

Execute all the check_* methods of the ProfileConstraintChecker against the given ProfileElementSequence

exception pySim.esim.saip.validation.ProfileError

Raised when a ProfileConstraintChecker finds an error in a file [structure].

1.8 osmo-smdpp

osmo-smdpp is a proof-of-concept implementation of a minimal **SM-DP+** as specified for the *GSMA Consumer eSIM Remote SIM provisioning*.

At least at this point, it is intended to be used for research and development, and not as a production SM-DP+.

Unless you are a GSMA SAS-SM accredited SM-DP+ operator and have related DPtls, DPauth and DPpb certificates signed by the GSMA CI, you **can not use osmo-smdpp with regular production eUICC**. This is due to how the GSMA eSIM security architecture works. You can, however, use osmo-smdpp with so-called *test-eUICC*, which contain certificates/keys signed by GSMA test certificates as laid out in GSMA SGP.26.

At this point, osmo-smdpp does not support anything beyond the bare minimum required to download eSIM profiles to an eUICC. Specifically, there is no ES2+ interface, and there is no built-in support for profile personalization yet.

osmo-smdpp currently

- [by default] uses test certificates copied from GSMA SGP.26 into *./smdpp-data/certs*, assuming that your osmo-smdpp would be running at the host name *testsmdpplus1.example.com*. You can of course replace those certificates with your own, whether SGP.26 derived or part of a *private root CA* setup with matching eUICCs.
- doesn't understand profile state. Any profile can always be downloaded any number of times, irrespective of the EID or whether it was downloaded before. This is actually very useful for R&D and testing, as it doesn't require you to generate new profiles all the time. This logic of course is unsuitable for production usage.
- doesn't perform any personalization, so the IMSI/ICCID etc. are always identical (the ones that are stored in the respective UPP *.der* files)
- **is absolutely insecure**, as it
- does not perform all of the mandatory certificate verification (it checks the certificate chain, but not the expiration dates nor any CRL)
- does not evaluate/consider any *Confirmation Code*
- stores the sessions in an unencrypted *python shelve* and is hence leaking one-time key materials used for profile encryption and signing.

1.8.1 Running osmo-smdpp

osmo-smdpp comes with built-in TLS support which is enabled by default. However, it is always possible to disable the built-in TLS support if needed.

In order to use osmo-smdpp without the built-in TLS support, it has to be put behind a TLS reverse proxy, which terminates the ES9+ HTTPS traffic from the LPA, and then forwards it as plain HTTP to osmo-smdpp.

NOTE: The built in TLS support in osmo-smdpp makes use of the python *twisted* framework. Older versions of this framework appear to have problems when using the example elliptic curve certificates (both NIST and Brainpool) from GSMA.

nginx as TLS proxy

If you chose to use *nginx* as TLS reverse proxy, you can use the following configuration snippet:

```
upstream smdpp {
        server localhost:8000;
}

server {
        listen 443 ssl;
        server_name testsmdpplus1.example.com;

        ssl_certificate /my/path/to/pysim/smdpp-data/certs/DPtls/CERT_S_SM_DP_TLS_NIST.pem;
        ssl_certificate_key /my/path/to/pysim/smdpp-data/certs/DPtls/SK_S_SM_DP_TLS_NIST.
↪pem;

        location / {
                proxy_read_timeout 600s;

                proxy_hide_header X-Powered-By;
                proxy_set_header X-Real-IP $remote_addr;
```

(continues on next page)

(continued from previous page)

```
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto https;
            proxy_set_header X-Forwarded-Port $proxy_port;
            proxy_set_header Host $host;

            proxy_pass http://smdpp/;
        }
}
```

You can of course achieve a similar functionality with apache, lighttpd or many other web server software.

supplementary files

The *smdpp-data/certs* directory contains the DPtls, DPauth and DPpb as well as CI certificates used; they are copied from GSMA SGP.26 v2. You can of course replace them with custom certificates if you're operating eSIM with a *private root CA*.

The *smdpp-data/upp* directory contains the UPP (Unprotected Profile Package) used. The file names (without .der suffix) are looked up by the matchingID parameter from the activation code presented by the LPA.

commandline options

Typically, you just run osmo-smdpp without any arguments, and it will bind its built-in HTTPS ES9+ interface to *localhost* TCP port 443. In this case an external TLS reverse proxy is not needed.

osmo-smdpp currently doesn't have any configuration file.

There are command line options for binding:

Bind the HTTPS ES9+ to a port other than 443:

```
./osmo-smdpp.py -p 8443
```

Disable the built-in TLS support and bind the plain-HTTP ES9+ to a port 8000:

```
./osmo-smdpp.py -p 8000 --nossl
```

Bind the HTTP ES9+ to a different local interface:

```
./osmo-smdpp.py -H 127.0.0.2
```

DNS setup for your LPA

The LPA must resolve *testsmdpplus1.example.com* to the IP address of your TLS proxy.

It must also accept the TLS certificates used by your TLS proxy. In case osmo-smdpp is used with built-in TLS support, it will use the certificates provided in smdpp-data.

NOTE: The HTTPS ES9+ interface cannot be addressed by the LPA directly via its IP address. The reason for this is that the included SGP.26 (DPtls) test certificates explicitly restrict the hostname to *testsmdpplus1.example.com* in the *X509v3 Subject Alternative Name* extension. Using a bare IP address as hostname may cause the certificate to be rejected by the LPA.

Supported eUICC

If you run osmo-smdpp with the included SGP.26 (DPauth, DPpb) certificates, you must use an eUICC with matching SGP.26 certificates, i.e. the EUM certificate must be signed by a SGP.26 test root CA and the eUICC certificate in turn must be signed by that SGP.26 EUM certificate.

sysmocom (sponsoring development and maintenance of pySim and osmo-smdpp) is selling SGP.26 test eUICC as *sysmoEUICC1-C2T*. They are publicly sold in the sysmocom webshop.

In general you can use osmo-smdpp also with certificates signed by any other certificate authority. You just always must ensure that the certificates of the SM-DP+ are signed by the same root CA as those of your eUICCs.

Hypothetically, osmo-smdpp could also be operated with GSMA production certificates, but it would require that somebody brings the code in-line with all the GSMA security requirements (HSM support, ...) and operate it in a GSMA SAS-SM accredited environment and pays for the related audits.

1.9 sim-rest-server

Sometimes there are use cases where a [remote] application will need access to a USIM for authentication purposes. This is, for example, in case an IMS test client needs to perform USIM based authentication against an IMS core.

The pysim repository contains two programs: *sim-rest-server.py* and *sim-rest-client.py* that implement a simple approach to achieve the above:

sim-rest-server.py speaks to a [usually local] USIM via the PC/SC API and provides a high-level REST API towards [local or remote] applications that wish to perform UMTS AKA using the USIM.

sim-rest-client.py implements a small example client program to illustrate how the REST API provided by *sim-rest-server.py* can be used.

1.9.1 REST API Calls

POST /sim-auth-api/v1/slot/SLOT_NR

where SLOT_NR is the integer-encoded slot number (corresponds to PC/SC reader number). When using a single sysmoOCTSIM board, this is in the range of 0..7

Example: */sim-auth-api/v1/slot/0* for the first slot.

Request Body

The request body is a JSON document, comprising of

1. the RAND and AUTN parameters as hex-encoded string
2. the application against which to authenticate (USIM, ISIM)

Example:

```
{
   "rand": "bb685a4b2fc4d697b9d6a129dd09a091",
   "autn": "eea7906f8210000004faf4a7df279b56"
}
```

HTTP Status Codes

HTTP status codes are used to represent errors within the REST server and the SIM reader hardware. They are not used to communicate protocol level errors reported by the SIM Card. An unsuccessful authentication will hence have a *200 OK* HTTP Status code and then encode the SIM specific error information in the Response Body.

Status	Code	Description
200	OK	Successful execution
400	Bad Request	Request body is malformed
404	Not Found	Specified SIM Slot doesn't exist
410	Gone	No SIM card inserted in slot

Response Body

The response body is a JSON document, either

1. a successful outcome; encoding RES, CK, IK as hex-encoded string
2. a sync failure; encoding AUTS as hex-encoded string
3. errors #. authentication error (incorrect MAC) #. authentication error (security context not supported) #. key freshness failure #. unspecified card error

Example (success):

```
{
  "successful_3g_authentication": {
      "res": "b15379540ec93985",
      "ck": "713fde72c28cbd282a4cd4565f3d6381",
      "ik": "2e641727c95781f1020d319a0594f31a",
      "kc": "771a2c995172ac42"
   }
}
```

Example (re-sync case):

```
{
   "synchronisation_failure": {
      "auts": "dc2a591fe072c92d7c46ecfe97e5"
   }
}
```

1.9.2 Concrete example using the included sysmoISIM-SJA2

This was tested using SIMs ending in IMSI numbers 45890…45899

The following command were executed successfully:

Slot 0

```
$ /usr/local/src/pysim/contrib/sim-rest-client.py -c 1 -n 0 -k 841EAD87BC9D974ECA1C167409357601 -
↪o 3211CACDD64F51C3FD3013ECD9A582A0
-> {'rand': 'fb195c7873b20affa278887920b9dd57', 'autn': 'd420895a6aa2000089cd016f8d8ae67c'}
<- {'successful_3g_authentication': {'res': '131004db2ff1ce8e', 'ck': 'd42eb5aa085307903271b2422b698bad
↪', 'ik': '485f81e6fd957fe3cad374adf12fe1ca', 'kc': '64d3f2a32f801214'}}
```

Slot 1

```
$ /usr/local/src/pysim/contrib/sim-rest-client.py -c 1 -n 1 -k 5C2CE9633FF9B502B519A4EACD16D9DF -
↪o 9834D619E71A02CD76F00CC7AA34FB32
-> {'rand': '433dc5553db95588f1d8b93870930b66', 'autn': '126bafdcbe9e00000026a208da61075d'}
```

(continues on next page)

(continued from previous page)

```
<- {'successful_3g_authentication': {'res': '026d7ac42d379207', 'ck': '83a90ba331f47a95c27a550b174c4a1f',
↪ 'ik': '31e1d10329ffaf0ca1684a1bf0b0a14a', 'kc': 'd15ac5b0fff73ecc'}}
```

1.10 suci-keytool

Subscriber concealment is an important feature of the 5G SA architecture: It avoids the many privacy issues associated with having a permanent identifier (SUPI, traditionally the IMSI) transmitted in plain text over the air interface. Using SUCI solves this issue not just for the air interface; it even ensures the SUPI/IMSI is not known to the visited network (VPLMN) at all.

In principle, the SUCI mechanism works by encrypting the SUPI by asymmetric (public key) cryptography: Only the HPLMN is in possession of the private key and hence can decrypt the SUCI to the SUPI, while each subscriber has the public key in order to encrypt their SUPI into the SUCI. In reality, the details are more complex, as there are ephemeral keys and cryptographic MAC involved.

In any case, in order to operate a SUCI-enabled 5G SA network, you will have to

1. generate a ECC key pair of public + private key
2. deploy the public key on your USIMs
3. deploy the private key on your 5GC, specifically the UDM function

pysim contains (in its *contrib* directory) a small utility program that can make it easy to generate such keys: *suci-keytool.py*

1.10.1 Generating keys

Example: Generating a *secp256r1* ECC public key pair and storing it to */tmp/suci.key*:

```
$ ./contrib/suci-keytool.py --key-file /tmp/suci.key generate-key --curve secp256r1
```

1.10.2 Dumping public keys

In order to store the key to SIM cards as part of *ADF.USIM/DF.5GS/EF.SUCI_Calc_Info*, you will need a hexadecimal representation of the public key. You can achieve that using the *dump-pub-key* operation of suci-keytool:

Example: Dumping the public key part from a previously generated key file:

```
$ ./contrib/suci-keytool.py --key-file /tmp/suci.key dump-pub-key
0473152f32523725f5175d255da2bd909de97b1d06449a9277bc629fe42112f8643e6b69aa6dce6c86714ccbe6f2e0f4f4898d102e2b3f0
```

If you want the point-compressed representation, you can use the *–compressed* option:

```
$ ./contrib/suci-keytool.py --key-file /tmp/suci.key dump-pub-key --compressed
0373152f32523725f5175d255da2bd909de97b1d06449a9277bc629fe42112f864
```

1.10.3 suci-keytool syntax

Generate or export SUCI keys for 5G SA networks

```
usage: contrib/suci-keytool.py [-h] --key-file KEY_FILE
                               {generate-key,dump-pub-key} ...
```

Positional Arguments

command Possible choices: generate-key, dump-pub-key

The command to perform

Named Arguments

--key-file The key file to use

Sub-commands

generate-key

Generate a new key pair

```
contrib/suci-keytool.py generate-key [-h] --curve {secp256r1,curve25519}
```

Named Arguments

--curve Possible choices: secp256r1, curve25519

The ECC curve to use

dump-pub-key

Dump the public key

```
contrib/suci-keytool.py dump-pub-key [-h] [--compressed]
```

Named Arguments

--compressed Use point compression

Default: False

1.11 saip-tool

eSIM profiles are stored as a sequence of profile element (PE) objects in an ASN.1 DER encoded binary file. To inspect, verify or make changes to those files, the *saip-tool.py* utility can be used.

NOTE: The file format, eSIM SAIP (SimAlliance Interoperable Profile) is specified in *TCA eUICC Profile Package: Interoperable Format Technical Specification*

1.11.1 Profile Package Examples

pySim ships with a set of TS48 profile package examples. Those examples can be found in *pysim/smdpp-data/upp*. The files can be used as input for *saip-tool.py*. (see also GSMA TS.48 - Generic eUICC Test Profile for Device Testing)

See also: https://github.com/GSMATerminals/Generic-eUICC-Test-Profile-for-Device-Testing-Public

1.11.2 JAVA card applets

The *saip-tool.py* can also be used to manage JAVA-card applets (Application PE) inside a profile package. The user has the option to add, remove and inspect applications and their instances. In the following we will discuss a few JAVA-card related use-cases of *saip-tool.py*

NOTE: see also *contrib* folder for script examples (*saip-tool_example_*.sh*)

Inserting applications

An application is usually inserted in two steps. In the first step, the application PE is created and populated with the executable code from a provided *.cap* or *.ijc* file. The user also has to pick a suitable load block AID.

The application instance, which exists inside the application PE, is created in a second step. Here the user must reference the load block AID and pick, among other application related parameters, a suitable class and instance AID.

Example: Adding a JAVA-card applet to an existing profile package

```
# Step #1: Create the application PE and load the ijc contents from the .cap file:
$ ./contrib/saip-tool.py upp.der add-app --output-file upp_with_app.der --applet-file app.cap --aid
↪'1122334455'
Read 28 PEs from file 'upp.der'
Applying applet file: 'app.cap'...
application PE inserted into PE Sequence after securityDomain PE AID: a000000151000000
Writing 29 PEs to file 'upp_with_app.der'...

# Step #2: Create the application instance inside the application PE created in step #1:
$ ./contrib/saip-tool.py upp_with_app.der add-app-inst --output-file upp_with_app_and_instance.der \
      --aid '1122334455' \
      --class-aid '112233445501' \
      --inst-aid '112233445501' \
      --app-privileges '00' \
      --app-spec-pars '00' \
      --uicc-toolkit-app-spec-pars '0100150500000000000000000000000'
Read 29 PEs from file 'upp_with_app.der'
Found Load Package AID: 1122334455, adding new instance AID: 112233445501 to Application PE...
Writing 29 PEs to file 'upp_with_app_and_instance.der'...
```

NOTE: The parameters of the sub-commands *add-app* and *add-app-inst* are application specific. It is up to the application developer to pick parameters that suit the application correctly. For an exact command reference see section *saip-tool syntax*. For parameter details see *TCA eUICC Profile Package: Interoperable Format Technical Specification*, section 8.7 and ETSI TS 102 226, section 8.2.1.3.2

Inspecting applications

To inspect the application PE contents of an existing profile package, sub-command *info* with parameter '–apps' can be used. This command lists out all application and their parameters in detail. This allows an application developer to check if the applet insertion was carried out as expected.

Example: Listing applications and their parameters

```
$ ./contrib/saip-tool.py upp_with_app_and_instance.der info --apps
Read 29 PEs from file 'upp_with_app_and_instance.der'
Application #0:
      loadBlock:
            loadPackageAID: '1122334455' (5 bytes)
            loadBlockObject: '01000fdecaffed010204000105d07002ca440200...
↪681080056810a00633b44104b431066800a10231' (569 bytes)
      instanceList[0]:
            applicationLoadPackageAID: '1122334455' (5 bytes)
            classAID: '112233445501' (8 bytes)
            instanceAID: '112233445501' (8 bytes)
            applicationPrivileges: '00' (1 bytes)
            lifeCycleState: '07' (1 bytes)
```

(continues on next page)

(continued from previous page)

```
            applicationSpecificParametersC9: '00' (1 bytes)
            applicationParameters:
                uiccToolkitApplicationSpecificParametersField: '0100150500000000000000000000000000'
↪(16 bytes)
```

In case further analysis with external tools or transfer of applications from one profile package to another is necessary, the executable code in the *loadBlockObject* field can be extracted to an *.ijc* or an *.cap* file.

Example: Extracting applications from a profile package

```
$ ./contrib/saip-tool.py upp_with_app_and_instance.der extract-apps --output-dir ./apps --format ijc
Read 29 PEs from file 'upp_with_app_and_instance.der'
Writing Load Package AID: 1122334455 to file ./apps/8949449999999990023f-1122334455.ijc
```

Removing applications

An application PE can be removed using sub-command *remove-app*. The user passes the load package AID as parameter. Then *saip-tool.py* will search for the related application PE and delete it from the PE sequence.

Example: Remove an application from a profile package

```
$ ./contrib/saip-tool.py upp_with_app_and_instance.der remove-app --output-file upp_without_app.
↪der --aid '1122334455'
Read 29 PEs from file 'upp_with_app_and_instance.der'
Found Load Package AID: 1122334455, removing related PE (id=23) from Sequence...
Removing PE application (id=23) from Sequence...
Writing 28 PEs to file 'upp_without_app.der'...
```

In some cases it is useful to remove only an instance from an existing application PE. This may be the case when the an application developer wants to modify parameters of an application by removing and re-adding the instance. The operation basically rolls the state back to step 1 explained in section *Inserting applications*

Example: Remove an application instance from an application PE

```
$ ./contrib/saip-tool.py upp_with_app_and_instance.der remove-app-inst --output-file upp_without_
↪app.der --aid '1122334455' --inst-aid '112233445501'
Read 29 PEs from file 'upp_with_app_and_instance.der'
Found Load Package AID: 1122334455, removing instance AID: 112233445501 from Application PE...
Removing instance from Application PE...
Writing 29 PEs to file 'upp_with_app.der'...
```

1.11.3 saip-tool syntax

Utility program to work with eSIM SAIP (SimAlliance Interoperable Profile) files.

```
usage: contrib/saip-tool.py [-h]
                            [--loglevel {DEBUG,INFO,WARNING,ERROR,CRITICAL}]
                            [--debug]
                            INPUT_UPP
                            {split,dump,check,extract-pe,remove-pe,remove-naa,info,extract-apps,add-app,
↪remove-app,add-app-inst,remove-app-inst,edit-mand-srv-list,tree} ...
```

Positional Arguments

INPUT_UPP Unprotected Profile Package Input file

command Possible choices: split, dump, check, extract-pe, remove-pe, remove-naa, info, extract-apps, add-app, remove-app, add-app-inst, remove-app-inst, edit-mand-srv-list, tree

The command to perform

Named Arguments

--loglevel Possible choices: DEBUG, INFO, WARNING, ERROR, CRITICAL

Set the logging level

Default: "INFO"

--debug Enable DEBUG logging

Default: False

Sub-commands

split

Split PE-Sequence into individual PEs

```
contrib/saip-tool.py split [-h] [--output-prefix OUTPUT_PREFIX]
```

Named Arguments

--output-prefix Prefix path/filename for output files

Default: "."

dump

Dump information on PE-Sequence

```
contrib/saip-tool.py dump [-h] [--dump-decoded]
                          {all_pe,all_pe_by_type,all_pe_by_naa}
```

Positional Arguments

mode Possible choices: all_pe, all_pe_by_type, all_pe_by_naa

Named Arguments

--dump-decoded Dump decoded PEs

Default: False

check

Run constraint checkers on PE-Sequence

```
contrib/saip-tool.py check [-h]
```

extract-pe

Extract specified PE to (DER encoded) file

```
contrib/saip-tool.py extract-pe [-h] --pe-file PE_FILE
                                [--identification IDENTIFICATION]
```

Named Arguments

--pe-file	PE file name
--identification	Extract PE matching specified identification

remove-pe

Remove specified PEs from PE-Sequence

```
contrib/saip-tool.py remove-pe [-h] --output-file OUTPUT_FILE
                               [--identification IDENTIFICATION] [--type TYPE]
```

Named Arguments

--output-file	Output file name
--identification	Remove PEs matching specified identification
	Default: []
--type	Remove PEs matching specified type
	Default: []

remove-naa

Remove specified NAAs from PE-Sequence

```
contrib/saip-tool.py remove-naa [-h] --output-file OUTPUT_FILE
                                --naa-type {csim,usim,isim}
```

Named Arguments

--output-file	Output file name
--naa-type	Possible choices: csim, usim, isim
	Network Access Application type to remove

info

Display information about the profile

```
contrib/saip-tool.py info [-h] [--apps]
```

Named Arguments

--apps List applications and their related instances

Default: False

extract-apps

Extract applications as loadblock file

```
contrib/saip-tool.py extract-apps [-h] [--output-dir OUTPUT_DIR]
                                  [--format {ijc,cap}]
```

Named Arguments

--output-dir Output directory (where to store files)

Default: "."

--format Possible choices: ijc, cap

Data format of output files

Default: "cap"

add-app

Add application to PE-Sequence

```
contrib/saip-tool.py add-app [-h] --output-file OUTPUT_FILE
                             --applet-file APPLET_FILE --aid AID
                             [--sd-aid SD_AID]
                             [--non-volatile-code-limit NON_VOLATILE_CODE_LIMIT]
                             [--volatile-data-limit VOLATILE_DATA_LIMIT]
                             [--non-volatile-data-limit NON_VOLATILE_DATA_LIMIT]
                             [--hash-value HASH_VALUE]
```

Named Arguments

--output-file Output file name

--applet-file Applet file name

--aid Load package AID

--sd-aid Security Domain AID

--non-volatile-code-limit Non volatile code limit (C6)

--volatile-data-limit Volatile data limit (C7)

--non-volatile-data-limit Non volatile data limit (C8)

--hash-value Hash value

remove-app

Remove application from PE-Sequence

```
contrib/saip-tool.py remove-app [-h] --output-file OUTPUT_FILE --aid AID
```

Named Arguments

--output-file Output file name

--aid Load package AID

add-app-inst

Add application instance to Application PE

```
contrib/saip-tool.py add-app-inst [-h] --output-file OUTPUT_FILE --aid AID
                          --class-aid CLASS_AID --inst-aid INST_AID
                          [--app-privileges APP_PRIVILEGES]
                          [--volatile-memory-quota VOLATILE_MEMORY_QUOTA]
                          [--non-volatile-memory-quota NON_VOLATILE_MEMORY_QUOTA]
                          [--app-spec-pars APP_SPEC_PARS]
                          [--uicc-toolkit-app-spec-pars UICC_TOOLKIT_APP_SPEC_PARS]
                          [--uicc-access-app-spec-pars UICC_ACCESS_APP_SPEC_PARS]
                          [--uicc-adm-access-app-spec-pars UICC_ADM_ACCESS_APP_SPEC_PARS]
                          [--process-data PROCESS_DATA]
```

Named Arguments

--output-file Output file name

--aid Load package AID

--class-aid Class AID

--inst-aid Instance AID (must match Load package AID)

--app-privileges Application privileges

Default: “000000”

--volatile-memory-quota Volatile memory quota (C7)

--non-volatile-memory-quota Non volatile memory quota (C8)

--app-spec-pars Application specific parameters (C9)

Default: “00”

--uicc-toolkit-app-spec-pars UICC toolkit application specific parameters field

--uicc-access-app-spec-pars UICC Access application specific parameters field

--uicc-adm-access-app-spec-pars UICC Administrative access application specific parameters field

--process-data Process personalization APDUs

Default: []

remove-app-inst

Remove application instance from Application PE

```
contrib/saip-tool.py remove-app-inst [-h] --output-file OUTPUT_FILE --aid AID
                                     --inst-aid INST_AID
```

Named Arguments

--output-file Output file name

--aid Load package AID

--inst-aid Instance AID

edit-mand-srv-list

Add/Remove service flag from/to mandatory services list

```
contrib/saip-tool.py edit-mand-srv-list [-h] --output-file OUTPUT_FILE
                                        [--add-flag {contactless,usim,isim,csim,milenage,tuak128,cave,gba-usim,gba-
↪isim,mbms,eap,javacard,multos,multiple-usim,multiple-isim,multiple-csim,tuak256,usim-test-algorithm,
↪ber-tlv,dfLink,cat-tp,get-identity,profile-a-x25519,profile-b-p256,suciCalculatorApi,dns-resolution,
↪scp11ac,scp11c-authorization-mechanism,s16mode,eaka}]
                                        [--remove-flag {contactless,usim,isim,csim,milenage,tuak128,cave,gba-usim,
↪gba-isim,mbms,eap,javacard,multos,multiple-usim,multiple-isim,multiple-csim,tuak256,usim-test-
↪algorithm,ber-tlv,dfLink,cat-tp,get-identity,profile-a-x25519,profile-b-p256,suciCalculatorApi,dns-
↪resolution,scp11ac,scp11c-authorization-mechanism,s16mode,eaka}]
```

Named Arguments

--output-file Output file name

--add-flag Possible choices: contactless, usim, isim, csim, milenage, tuak128, cave, gba-usim, gba-isim, mbms, eap, javacard, multos, multiple-usim, multiple-isim, multiple-csim, tuak256, usim-test-algorithm, ber-tlv, dfLink, cat-tp, get-identity, profile-a-x25519, profile-b-p256, suciCalculatorApi, dns-resolution, scp11ac, scp11c-authorization-mechanism, s16mode, eaka

Add flag to mandatory services list

Default: []

--remove-flag Possible choices: contactless, usim, isim, csim, milenage, tuak128, cave, gba-usim, gba-isim, mbms, eap, javacard, multos, multiple-usim, multiple-isim, multiple-csim, tuak256, usim-test-algorithm, ber-tlv, dfLink, cat-tp, get-identity, profile-a-x25519, profile-b-p256, suciCalculatorApi, dns-resolution, scp11ac, scp11c-authorization-mechanism, s16mode, eaka

Remove flag from mandatory services list

Default: []

tree

Display the filesystem tree

```
contrib/saip-tool.py tree [-h]
```

1.12 smpp-ota-tool

The *smpp-ota-tool* allows users to send OTA SMS messages containing APDU scripts (RFM, RAM) via an SMPP server. The intended audience are developers who want to test/evaluate the OTA SMS interface of a SIM/UICC/eUICC. *smpp-ota-tool* is intended to be used as a companion tool for *pySim-smpp2sim*, however it should be usable on any other SMPP server (such as a production SMSC of a live cellular network) as well.

From the technical perspective *smpp-ota-tool* takes the role of an SMPP ESME. It takes care of the encoding, encryption and checksumming (signing) of the RFM/RAM OTA SMS and eventually submits it to the SMPP server. The program then waits for a response. The response is automatically parsed and printed on stdout. This makes the program also suitable to be called from shell scripts.

Note

In the following we will we will refer to *SIM* as one of the following: *SIM*, *USIM*, *ISIM*, *UICC*, *eUICC*, *eSIM*.

1.12.1 Applying OTA keys

Depending on the *SIM* type you will receive one or more sets of keys which you can use to communicate with the *SIM* through a secure channel protocol. When using the OTA SMS method, the SCP80 protocol is used and it therefore crucial to use a keyset that is actually suitable for SCP80.

A keyset usually consists of three keys:

1. KIC: the key used for ciphering (encryption/decryption)
2. KID: the key used to compute a cryptographic checksum (signing)
3. KIK: the key used to encrypt/decrypt key material (key rotation, adding of new keys)

From the transport security perspective, only KIC and KID are relevant. The KIK (also referenced as "Data Encryption Key", DEK) is only used when keys are rotated or new keys are added (see also ETSI TS 102 226, section 8.2.1.5).

When the keyset is programmed into the security domain of the *SIM*, it is tied to a specific cryptographic algorithm (3DES, AES128 or AES256) and a so called Key Version Number (KVN). The term "Key Version Number" is misleading, since it is actually not a version number. It is a unique identifier of a certain keyset which also identifies for which secure channel protocol the keyset may be used. Keysets with a KVN from 1-15 (0x01-0x0F) are suitable for SCP80. This means that it is not only important to know just the KIC/KID/KIK keys. Also the related algorithms and the KVN numbers must be known.

Note

SCP80 keysets typically start counting from 1 upwards. Typical configurations use a set of 3 keysets with KVN numbers 1-3.

1.12.2 Addressing an Application

When communicating with a specific application on a *SIM* via SCP80, it is important to address that application with the correct parameters. The following two parameters must be known in advance:

1. TAR: The Toolkit Application Reference (TAR) number is a three byte value that uniquely addresses an application on the *SIM*. The exact values may vary (see also ETSI TS 101 220, Table D.1).
2. MSL: The Minimum Security Level (MSL) is a bit-field that dictates which of the security measures encoded in the SPI are mandatory (see also ETSI TS 102 225, section 5.1.1).

1.12.3 A practical example

Note

This tutorial assumes that pySim-smpp2sim is running on the local machine with its default parameters. See also *pySim-smpp2sim*.

Let's assume that an OTA SMS shall be sent to the SIM RFM application of an sysmoISIM-SJA2. What we want to do is to select DF.GSM and to get the select response back.

We have received the following key material from the *SIM* vendor:

```
KIC1: F09C43EE1A0391665CC9F05AF4E0BD10
KID1: 01981F4A20999F62AF99988007BAF6CA
KIK1: 8F8AEE5CDCC5D361368BC45673D99195
KIC2: 01022916E945B656FDE03F806A105FA2
KID2: D326CB69F160333CC5BD1495D448EFD6
KIK2: 08037E0590DFE049D4975FFB8652F625
KIC3: 2B22824D0D27A3A1CEEC512B312082B4
KID3: F1697766925A11F4458295590137B672
KIK3: C7EE69B2C5A1C8E160DD36A38EB517B3
```

Those are three keysets. The enumeration is directly equal to the KVN used. All three keysets are 3DES keys, which means triple_des_cbc2 is the correct algorithm to use.

Note

The key set configuration can be confirmed by retrieving the key configuration using *get_data key_information* from within an SCP02 session on ADF.ISD.

In this example we intend to address the SIM RFM application on the *SIM*. Which according to the manual has TAR B00010 and MSL 0x06. When we hold 0x06 = 0b00000110 against the SPI coding chart (see also ETSI TS 102 225, section 5.1.1). We can deduct that Ciphering and Cryptographic Checksum are mandatory.

Note

The MSL (see also ETSI TS 102 226, section 6.1) is assigned to an application by the *SIM* issuer. It is a custom decision and may vary with different *SIM* types/profiles. In the case of sysmoISIM-SJS1/SJA2/SJA5 the counter requirement has been waived to simplify lab/research type use. In productive environments, *SIM* applications should ideally use an MSL that makes the counter mandatory.

In order to select DF.GSM (0x7F20) and to retrieve the select response, two APDUs are needed. The first APDU is the select command A0A40000027F20 and the second is the related get-response command A0C0000016. Those APDUs will be concatenated and are sent in a single message. The message containing the concatenated APDUs works as a script that is received by the SIM RFM application and then executed. This method poses some limitations that have to be taken into account when making requests like this (see also ETSI TS 102 226, section 5).

With this information we may now construct a commandline for *smpp-ota-tool.py*. We will pass the KVN as kid_idx and kic_idx (see also ETSI TS 102 225, Table 2, fields *KIc* and *KID*). Both index values should refer to the same keyset/KVN as keysets should not be mixed. (*smpp-ota-tool* still provides separate parameters anyway to allow testing with invalid keyset combinations)

```
$ PYTHONPATH=./ ./contrib/smpp-ota-tool.py --kic F09C43EE1A0391665CC9F05AF4E0BD10 --kid␣
↪01981F4A20999F62AF99988107BAF6CA --kid_idx 1 --kic_idx 1 --algo-crypt triple_des_cbc2 --algo-
↪auth triple_des_cbc2 --tar B00010 --apdu A0A40000027F20 --apdu A0C0000016
2026-02-26 17:13:56 INFO Connecting to localhost:2775...
2026-02-26 17:13:56 INFO C-APDU sending: a0a40000027f20a0c0000016...
2026-02-26 17:13:56 INFO SMS-TPDU sending:␣
↪027000002815061915 15b00010da1d6cbbd0d11ce4330d844c7408340943e843f67a6d7b0674730881605fd62d...
2026-02-26 17:13:56 INFO SMS-TPDU sent, waiting for response...
2026-02-26 17:13:56 INFO SMS-TPDU received:␣
↪027100002c12b000107ddf58d1780f771638b3975759f4296cf5c31efc87a16a1b61921426baa16da1b5ba1a9951d59a39
2026-02-26 17:13:56 INFO SMS-TPDU decoded: (Container(rpl=44, rhl=18, tar=b'\xb0\x00\x10', cntr=b
↪'\x00\x00\x00\x00\x00', pcntr=0, response_status=uEnumIntegerString.new(0, 'por_ok'), cc_rc=b'\
↪x8f\xea\xf5.\xf4\x0e\xc2\x14', secured_data=b'\x02\x90\x00\x00\x00\xff\xff\x7f \x02\x00\x00\x00\
↪x00\x00\t\xb1\x065\x04\x00\x83\x8a\x83\x8a'), Container(number_of_commands=2, last_status_
↪word=u'9000', last_response_data=u'0000ffff7f2002000000000009b106350400838a838a'))
2026-02-26 17:13:56 INFO R-APDU received: 0000ffff7f2002000000000009b106350400838a838a 9000
0000ffff7f2002000000000009b106350400838a838a 9000
2026-02-26 17:13:56 INFO Disconnecting...
```

The result we see is the select response of DF.GSM and a status word indicating that the last command has been processed normally.

As we can see, this mechanism now allows us to perform small administrative tasks remotely. We can read the contents of files remotely or make changes to files. Depending on the changes we make, there may be security issues arising from replay attacks. With the commandline above, the communication is encrypted and protected by a cryptographic checksum, so an adversary can neither read, nor alter the message. However, an adversary could still replay an intercepted message and the *SIM* would happily execute the contained APDUs again.

To prevent this, we may include a replay protection counter within the message. In this case, the MSL indicates that a replay protection counter is not required. However, to extended the security of our messages, we may chose to use a counter anyway. In the following example, we will encode a counter value of 100. We will instruct the *SIM* to make sure that the value we send is higher than the counter value that is currently stored in the *SIM*.

To add a replay connection counter we add the commandline arguments *–cntr-req* to set the counter requirement and *–cntr* to pass the counter value.

```
$ PYTHONPATH=./ ./contrib/smpp-ota-tool.py --kic F09C43EE1A0391665CC9F05AF4E0BD10 --kid␣
↪01981F4A20999F62AF99988107BAF6CA --kid_idx 1 --kic_idx 1 --algo-crypt triple_des_cbc2 --algo-
↪auth triple_des_cbc2 --tar B00010 --apdu A0A40000027F20 --apdu A0C0000016 --cntr-req counter_
↪must_be_higher --cntr 100
2026-02-26 17:16:39 INFO Connecting to localhost:2775...
2026-02-26 17:16:39 INFO C-APDU sending: a0a40000027f20a0c0000016...
2026-02-26 17:16:39 INFO SMS-TPDU sending:␣
↪027000002815161915 15b000103a4f599e94f2b5dcfbbda984761b7977df6514c57a580fb4844787c436d2eade...
2026-02-26 17:16:39 INFO SMS-TPDU sent, waiting for response...
2026-02-26 17:16:39 INFO SMS-TPDU received:␣
↪027100002c12b0001049fb0315f6c6401b553867f412cefaf9355b38271178edb342a3bc9cc7e670cdc1f45eea6ffcbb39
2026-02-26 17:16:39 INFO SMS-TPDU decoded: (Container(rpl=44, rhl=18, tar=b'\xb0\x00\x10', cntr=b
↪'\x00\x00\x00\x00d', pcntr=0, response_status=uEnumIntegerString.new(0, 'por_ok'), cc_rc=b'\xa9/
↪\xc7\xc9\x00"\xab5', secured_data=b'\x02\x90\x00\x00\x00\xff\xff\x7f \x02\x00\x00\x00\x00\x00\
↪t\xb1\x065\x04\x00\x83\x8a\x83\x8a'), Container(number_of_commands=2, last_status_word=u
↪'9000', last_response_data=u'0000ffff7f2002000000000009b106350400838a838a'))
2026-02-26 17:16:39 INFO R-APDU received: 0000ffff7f2002000000000009b106350400838a838a 9000
0000ffff7f2002000000000009b106350400838a838a 9000
2026-02-26 17:16:39 INFO Disconnecting...
```

The *SIM* has accepted the message. The message got processed and the *SIM* has set its internal to 100. As an experiment, we may try to re-use the counter value:

```
$ PYTHONPATH=./ ./contrib/smpp-ota-tool.py --kic F09C43EE1A0391665CC9F05AF4E0BD10 --kid␣
↪01981F4A20999F62AF99988107BAF6CA --kid_idx 1 --kic_idx 1 --algo-crypt triple_des_cbc2 --algo-
↪auth triple_des_cbc2 --tar B00010 --apdu A0A40000027F20 --apdu A0C0000016 --cntr-req counter_
↪must_be_higher --cntr 100
2026-02-26 17:16:43 INFO Connecting to localhost:2775...
2026-02-26 17:16:43 INFO C-APDU sending: a0a40000027f20a0c0000016...
2026-02-26 17:16:43 INFO SMS-TPDU sending:␣
↪027000002815161915 15b000103a4f599e94f2b5dcfbbda984761b7977df6514c57a580fb4844787c436d2eade...
2026-02-26 17:16:43 INFO SMS-TPDU sent, waiting for response...
2026-02-26 17:16:43 INFO SMS-TPDU received: 027100000b0ab0001000000000000006
2026-02-26 17:16:43 INFO SMS-TPDU decoded: (Container(rpl=11, rhl=10, tar=b'\xb0\x00\x10', cntr=b
↪'\x00\x00\x00\x00\x00', pcntr=0, response_status=uEnumIntegerString.new(6, 'undefined_security_
↪error'), cc_rc=b'', secured_data=b''), None)
Traceback (most recent call last):
File "/home/user/work/git_master/pysim/./contrib/smpp-ota-tool.py", line 238, in <module>
 resp, sw = smpp_handler.transceive_apdu(apdu, opts.src_addr, opts.dest_addr, opts.timeout)
            ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
↪^^^^^^^^^
File "/home/user/work/git_master/pysim/./contrib/smpp-ota-tool.py", line 162, in transceive_apdu
  raise ValueError("Response does not contain any last_response_data, no R-APDU received!")
ValueError: Response does not contain any last_response_data, no R-APDU received!
2026-02-26 17:16:43 INFO Disconnecting...
```

As we can see, the *SIM* has rejected the message with an *undefined_security_error*. The replay-protection-counter ensures that a message can only be sent once.

Note

The replay-protection-counter is implemented as a 5 byte integer value (see also ETSI TS 102 225, Table 3). When the counter has reached its maximum, it will not overflow nor can it be reset.

1.12.4 Expanded remote application data format

smpp-ota-tool uses the TS 102 226 section 5.1 compact remote application data format by default. This format concatenates C-APDUs into one command string and only the result of the LAST executed command is reported back. Retrieving the response data therefore requires a GET RESPONSE C-APDU, and only a single GET RESPONSE command may occur per script.

The TS 102 226 section 5.2 expanded remote application data format removes these limitations: Each C-APDU is wrapped in its own C-APDU TLV inside a Command Scripting template, and the response is a Response Scripting template that contains one R-APDU TLV with the full response data and status word per executed command. To use it, pass --format expanded; every --apdu argument then becomes its own C-APDU TLV.

Note

The expanded format does not use GET RESPONSE. To retrieve response data from a case 2 or case 4 command, include an Le field in the C-APDU. i.e. Le='00' instructs the card to return all available response data in the R-APDU, with no 256-byte limit (TS 102 226, section 5.2.1.1). Without the Le field no response data is returned, except a status word for the last command!.

For example, a GP GET STATUS of all applications (80F24002024F00) returns a registry that can be much larger than 256 bytes. In the compact format the card would only answer with 61xx procedure bytes. In the expanded format, appending Le='00' (i.e. 80F24002024F0000) makes the card return the whole registry in one exchange:

```
$ PYTHONPATH=./ ./contrib/smpp-ota-tool.py --kic <KIC> --kid <KID> --kid-idx 1 --kic-idx 1 \
   --algo-crypt triple_des_cbc2 --algo-auth triple_des_cbc2 --tar 000000 --cntr-req no_counter \
   --format expanded --apdu 80F24002024F0000
```

The response data (a concatenation of GlobalPlatform registry TLVs) can then be decoded with pySim.global_platform.GpRegistryRelatedData.from_tlv().

1.12.5 smpp-ota-tool syntax

Tool to send OTA SMS RFM/RAM messages via SMPP

```
usage: contrib/smpp-ota-tool.py [-h] [--host HOST] [--port PORT]
                                [--system-id SYSTEM_ID] [--password PASSWORD]
                                [--verbose]
                                [--algo-crypt {single_des,triple_des_cbc2,aes_cbc}]
                                [--algo-auth {single_des,triple_des_cbc2,aes_cmac}]
                                --kic KIC [--kic-idx KIC_IDX] --kid KID
                                [--kid-idx KID_IDX] [--cntr CNTR] --tar TAR
                                [--cntr-req {no_counter,counter_no_replay_or_seq,counter_must_be_higher,
↪counter_must_be_lower}]
                                [--no-ciphering]
                                [--rc-cc-ds {no_rc_cc_ds,rc,cc,ds}]
                                [--por-in-submit] [--por-no-ciphering]
                                [--por-rc-cc-ds {no_rc_cc_ds,rc,cc,ds}]
                                [--por-req {no_por,por_required,por_only_when_error}]
                                [--src-addr SRC_ADDR] [--dest-addr DEST_ADDR]
                                [--timeout TIMEOUT]
                                [--format {compact,expanded}] -a APDU
```

Named Arguments

--host	Host/IP of the SMPP server Default: "localhost"
--port	TCP port of the SMPP server Default: 2775
--system-id	System ID to use to bind to the SMPP server Default: "test"
--password	Password to use to bind to the SMPP server Default: "test"
--verbose	Enable verbose logging Default: False
--algo-crypt	Possible choices: single_des, triple_des_cbc2, aes_cbc OTA crypt algorithm Default: "triple_des_cbc2"
--algo-auth	Possible choices: single_des, triple_des_cbc2, aes_cmac OTA auth algorithm Default: "triple_des_cbc2"
--kic	OTA key (KIC)
--kic-idx	OTA key index (KIC) Default: 1
--kid	OTA key (KID)
--kid-idx	OTA key index (KID) Default: 1
--cntr	replay protection counter Default: 0
--tar	Toolkit Application Reference
--cntr-req	Possible choices: no_counter, counter_no_replay_or_seq, counter_must_be_higher, counter_must_be_lower Counter requirement Default: "no_counter"
--no-ciphering	Disable ciphering Default: False
--rc-cc-ds	Possible choices: no_rc_cc_ds, rc, cc, ds message check (rc=redundency check, cc=crypt. checksum, ds=digital signature) Default: "cc"

--por-in-submit	require PoR to be sent via SMS-SUBMIT
	Default: False
--por-no-ciphering	Disable ciphering (PoR)
	Default: False
--por-rc-cc-ds	Possible choices: no_rc_cc_ds, rc, cc, ds
	PoR check (rc=redundency check, cc=crypt. checksum, ds=digital signature)
	Default: “cc”
--por-req	Possible choices: no_por, por_required, por_only_when_error
	Proof of Receipt requirements
	Default: “por_required”
--src-addr	SMS source address (MSISDN)
	Default: “12”
--dest-addr	SMS destination address (MSISDN)
	Default: “23”
--timeout	Maximum response waiting time
	Default: 10
--format	Possible choices: compact, expanded
	Remote Application data format: ‘compact’ or ‘expanded’
	Default: “compact”
-a, --apdu	C-APDU to send

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

p

pySim.card_handler, 180
pySim.card_key_provider, 180
pySim.commands, 166
pySim.esim, 183
pySim.esim.bsp, 189
pySim.esim.es2p, 184
pySim.esim.es8p, 187
pySim.esim.es9p, 188
pySim.esim.http_json_api, 189
pySim.esim.rsp, 184
pySim.esim.saip, 192
pySim.esim.saip.oid, 206
pySim.esim.saip.personalization, 206
pySim.esim.saip.templates, 221
pySim.esim.saip.validation, 225
pySim.esim.x509_cert, 191
pySim.exceptions, 179
pySim.filesystem, 154
pySim.transport, 171
pySim.transport.calypso, 173
pySim.transport.modem_atcmd, 173
pySim.transport.pcsc, 174
pySim.transport.serial, 174
pySim.utils, 175

INDEX

A

activate_file() (*pySim.commands.SimCardCommands method*), 166
ActivationCode (*class in pySim.esim*), 183
add() (*pySim.esim.saip.templates.ProfileTemplateRegistry class method*), 224
add_application_df() (*pySim.filesystem.CardMF method*), 159
add_child() (*pySim.esim.saip.FsNodeDF method*), 194
add_file() (*pySim.esim.saip.FsNodeDF method*), 194
add_file() (*pySim.esim.saip.FsProfileElement method*), 194
add_file() (*pySim.esim.saip.ProfileElementIoT method*), 200
add_file() (*pySim.esim.saip.ProfileElementOptIoT method*), 201
add_file() (*pySim.filesystem.CardDF method*), 156
add_file_at_path() (*pySim.esim.saip.ProfileElementSequence method*), 204
add_files() (*pySim.filesystem.CardDF method*), 156
add_files() (*pySim.filesystem.CardModel class method*), 159
add_hdr_and_end() (*pySim.esim.saip.ProfileElementSequence method*), 204
add_instance() (*pySim.esim.saip.ProfileElementApplication method*), 197
add_intermediate_cert() (*pySim.esim.x509_cert.CertificateSet method*), 191
add_key() (*pySim.esim.saip.ProfileElementSD method*), 203
add_notification() (*pySim.esim.es8p.ProfileMetadata method*), 187
add_pin() (*pySim.esim.saip.ProfileElementPin method*), 202
add_puk() (*pySim.esim.saip.ProfileElementPuk method*), 202
add_scp() (*pySim.esim.saip.ProfileElementSD method*), 203
add_ssd() (*pySim.esim.saip.ProfileElementSequence method*), 204
Adm1 (*class in pySim.esim.saip.personalization*), 206
Adm2 (*class in pySim.esim.saip.personalization*), 206
AlgoConfig (*class in pySim.esim.saip.personalization*), 206
AlgorithmID (*class in pySim.esim.saip.personalization*), 207
ApiError, 189
ApiParam (*class in pySim.esim.http_json_api*), 189
ApiParamBase64 (*class in pySim.esim.http_json_api*), 190
ApiParamBoolean (*class in pySim.esim.http_json_api*), 190
ApiParamFqdn (*class in pySim.esim.http_json_api*), 190
ApiParamInteger (*class in pySim.esim.http_json_api*), 190
ApiParamString (*class in pySim.esim.http_json_api*), 190
append() (*pySim.esim.saip.ProfileElementSequence method*), 204
apply() (*pySim.esim.saip.personalization.ConfigurableParameter method*), 208
apply_matching_models() (*pySim.filesystem.CardModel static method*), 160
apply_val() (*pySim.esim.saip.personalization.AlgoConfig class method*), 206
apply_val() (*pySim.esim.saip.personalization.ConfigurableParameter class method*), 208
apply_val() (*pySim.esim.saip.personalization.EfUstServiceParam class method*), 210
apply_val() (*pySim.esim.saip.personalization.EuiccMandatoryServicePar class method*), 211
apply_val() (*pySim.esim.saip.personalization.Iccid class method*), 211
apply_val() (*pySim.esim.saip.personalization.Imsi class method*), 212
apply_val() (*pySim.esim.saip.personalization.MncLen class method*), 213
apply_val() (*pySim.esim.saip.personalization.Pin class method*), 214
apply_val() (*pySim.esim.saip.personalization.Pin2 class method*), 214
apply_val() (*pySim.esim.saip.personalization.Puk class*

method), 215
apply_val() (*pySim.esim.saip.personalization.SdKey class method*), 215
apply_val() (*pySim.esim.saip.personalization.SmspTpScAddr class method*), 219
apply_val() (*pySim.esim.saip.personalization.SuciCalcInfoParameter class method*), 219
apply_val() (*pySim.esim.saip.personalization.SuciRi class method*), 220
argparse_add_args() (*pySim.card_key_provider.CardKeyProvider static method*), 181
argparse_add_args() (*pySim.card_key_provider.CardKeyProviderCsv static method*), 181
argparse_add_args() (*pySim.card_key_provider.CardKeyProviderPgsql static method*), 182
argparse_add_reader_args() (*in module pySim.transport*), 173
authenticate() (*pySim.commands.SimCardCommands method*), 166
AuthenticateClient (*class in pySim.esim.es9p*), 188

B

base_df() (*pySim.esim.saip.templates.ProfileTemplate class method*), 224
bertlv_first_segment() (*in module pySim.esim.saip*), 206
BerTlvEF (*class in pySim.filesystem*), 154
BerTlvEF.ShellCommands (*class in pySim.filesystem*), 155
binary_size() (*pySim.commands.SimCardCommands method*), 166
BinaryParam (*class in pySim.esim.saip.personalization*), 207
BoundProfilePackage (*class in pySim.esim.es8p*), 187
boxed_heading_str() (*in module pySim.utils*), 177
bsp_key_derivation() (*in module pySim.esim.bsp*), 189
BspAlgo (*class in pySim.esim.bsp*), 189
BspAlgoCrypt (*class in pySim.esim.bsp*), 189
BspAlgoCryptAES128 (*class in pySim.esim.bsp*), 189
BspAlgoMac (*class in pySim.esim.bsp*), 189
BspAlgoMacAES128 (*class in pySim.esim.bsp*), 189
BspInstance (*class in pySim.esim.bsp*), 189
build_select_path_to() (*pySim.filesystem.CardFile method*), 158
bytes_for_nibbles() (*in module pySim.utils*), 177

C

calculate_luhn() (*in module pySim.utils*), 177
call_cancelOrder() (*pySim.esim.es2p.Es2pApiClient method*), 185
call_cancelOrder() (*pySim.esim.es2p.Es2pApiServerHandlerSmdpp method*), 186
call_cancelOrder() (*pySim.esim.es2p.Es2pApiServerSmdpp method*), 186
call_confirmOrder() (*pySim.esim.es2p.Es2pApiClient method*), 185
call_confirmOrder() (*pySim.esim.es2p.Es2pApiServerHandlerSmdpp method*), 186
call_confirmOrder() (*pySim.esim.es2p.Es2pApiServerSmdpp method*), 186
call_downloadOrder() (*pySim.esim.es2p.Es2pApiClient method*), 185
call_downloadOrder() (*pySim.esim.es2p.Es2pApiServerHandlerSmdpp method*), 186
call_downloadOrder() (*pySim.esim.es2p.Es2pApiServerSmdpp method*), 186
call_handleDownloadProgressInfo() (*pySim.esim.es2p.Es2pApiClient method*), 185
call_handleDownloadProgressInfo() (*pySim.esim.es2p.Es2pApiServerHandlerMno method*), 186
call_handleDownloadProgressInfo() (*pySim.esim.es2p.Es2pApiServerMno method*), 186
call_releaseProfile() (*pySim.esim.es2p.Es2pApiClient method*), 185
call_releaseProfile() (*pySim.esim.es2p.Es2pApiServerHandlerSmdpp method*), 186
call_releaseProfile() (*pySim.esim.es2p.Es2pApiServerSmdpp method*), 186
CalypsoSimLink (*class in pySim.transport.calypso*), 173
CancelOrder (*class in pySim.esim.es2p*), 184
CancelSession (*class in pySim.esim.es9p*), 188
card_key_provider_argparse_add_args() (*in module pySim.card_key_provider*), 182
card_key_provider_get() (*in module pySim.card_key_provider*), 182
card_key_provider_get_field() (*in module pySim.card_key_provider*), 183
card_key_provider_init() (*in module pySim.card_key_provider*), 183
card_key_provider_register() (*in module pySim.card_key_provider*), 183
CardADF (*class in pySim.filesystem*), 155
CardApplication (*class in pySim.filesystem*), 156
CardCommand (*class in pySim.utils*), 175
CardCommandSet (*class in pySim.utils*), 175
CardDF (*class in pySim.filesystem*), 156
CardDF.ShellCommands (*class in pySim.filesystem*), 156
CardEF (*class in pySim.filesystem*), 157
CardFile (*class in pySim.filesystem*), 157
CardHandler (*class in pySim.card_handler*), 180
CardHandlerAuto (*class in pySim.card_handler*), 180
CardHandlerBase (*class in pySim.card_handler*), 180
CardKeyFieldCryptor (*class in pySim.card_key_provider*), 180

CardKeyProvider (*class in pySim.card_key_provider*), 181
CardKeyProviderCsv (*class in pySim.card_key_provider*), 181
CardKeyProviderPgsql (*class in pySim.card_key_provider*), 182
CardMF (*class in pySim.filesystem*), 159
CardModel (*class in pySim.filesystem*), 159
cd() (*pySim.esim.saip.ProfileElementSequence method*), 204
cert_get_auth_key_id() (*in module pySim.esim.x509_cert*), 192
cert_get_subject_key_id() (*in module pySim.esim.x509_cert*), 192
cert_policy_has_oid() (*in module pySim.esim.x509_cert*), 192
CertAndPrivkey (*class in pySim.esim.x509_cert*), 191
CertificateSet (*class in pySim.esim.x509_cert*), 191
change_chv() (*pySim.commands.SimCardCommands method*), 166
check() (*pySim.esim.saip.validation.ProfileConstraintChecker method*), 225
check_forbidden() (*pySim.esim.saip.validation.FileCheckBasicStructure method*), 225
check_identification_unique() (*pySim.esim.saip.validation.CheckBasicStructure method*), 225
check_mandatory_services() (*pySim.esim.saip.validation.CheckBasicStructure method*), 225
check_mandatory_services_aka() (*pySim.esim.saip.validation.CheckBasicStructure method*), 225
check_number_of_occurrence() (*pySim.esim.saip.validation.CheckBasicStructure method*), 225
check_optional_ordering() (*pySim.esim.saip.validation.CheckBasicStructure method*), 225
check_seqence() (*pySim.esim.saip.validation.FileCheckBasicStructure method*), 225
check_signed() (*in module pySim.esim.x509_cert*), 192
check_start_and_end() (*pySim.esim.saip.validation.CheckBasicStructure method*), 225
check_template_modification_rules() (*pySim.esim.saip.File method*), 192
CheckBasicStructure (*class in pySim.esim.saip.validation*), 225
cla_with_lchan() (*in module pySim.commands*), 171
class_for_petype() (*pySim.esim.saip.ProfileElement class method*), 195
ClassVarMeta (*class in pySim.esim.saip.personalization*), 207
close() (*pySim.esim.rsp.RspSessionStore method*), 184
compile_asn1_subdir() (*in module pySim.esim*), 184
ConfigurableParameter (*class in pySim.esim.saip.personalization*), 207
ConfirmOrder (*class in pySim.esim.es2p*), 185
connect() (*pySim.transport.calypso.CalypsoSimLink method*), 173
connect() (*pySim.transport.LinkBase method*), 171
connect() (*pySim.transport.modem_atcmd.ModemATCommandLink method*), 173
connect() (*pySim.transport.pcsc.PcscSimLink method*), 174
connect() (*pySim.transport.serial.SerialSimLink method*), 175
counter_len_for_kvn() (*pySim.esim.saip.SecurityDomainKey static method*), 206
create_file() (*pySim.commands.SimCardCommands method*), 166
create_file() (*pySim.esim.saip.FsProfileElement method*), 194
cur_df (*pySim.esim.saip.ProfileElementSequence property*), 204
CyclicEF (*class in pySim.filesystem*), 160

D

DataObject (*class in pySim.utils*), 175
DataObjectChoice (*class in pySim.utils*), 176
DataObjectCollection (*class in pySim.utils*), 176
DataObjectSequence (*class in pySim.utils*), 176
deactivate_file() (*pySim.commands.SimCardCommands method*), 166
dec_imsi() (*in module pySim.utils*), 177
decimal_hex_to_str() (*pySim.esim.saip.personalization.DecimalHexParam class method*), 210
DecimalHexParam (*class in pySim.esim.saip.personalization*), 210
DecimalParam (*class in pySim.esim.saip.personalization*), 210
decode() (*pySim.esim.es8p.BoundProfilePackage method*), 187
decode() (*pySim.esim.http_json_api.ApiParam class method*), 189
decode() (*pySim.utils.DataObject method*), 175
decode() (*pySim.utils.DataObjectChoice method*), 176
decode() (*pySim.utils.DataObjectCollection method*), 176
decode() (*pySim.utils.DataObjectSequence method*), 176
decode_bin() (*pySim.filesystem.TransparentEF method*), 164
decode_client() (*pySim.esim.http_json_api.JsonHttpApiFunction method*), 190

decode_hex() (*pySim.filesystem.TransparentEF method*), 164
decode_multi() (*pySim.utils.DataObjectSequence method*), 176
decode_record_bin() (*pySim.filesystem.LinFixedEF method*), 161
decode_record_bin() (*pySim.filesystem.TransRecEF method*), 163
decode_record_hex() (*pySim.filesystem.LinFixedEF method*), 161
decode_record_hex() (*pySim.filesystem.TransRecEF method*), 163
decode_select_response() (*pySim.filesystem.CardFile method*), 158
decode_select_response() (*pySim.filesystem.CardMF method*), 159
decode_server() (*pySim.esim.http_json_api.JsonHttpApiFunction method*), 190
decode_str() (*pySim.esim.ActivationCode static method*), 183
decomposeATR() (*in module pySim.utils*), 177
decrypt() (*pySim.esim.bsp.BspAlgoCrypt method*), 189
decrypt_field() (*pySim.card_key_provider.CardKeyFieldCrypt method*), 180
default_source (*pySim.esim.saip.personalization.AlgorithmID attribute*), 207
default_source (*pySim.esim.saip.personalization.BinaryParam attribute*), 207
default_source (*pySim.esim.saip.personalization.EfUstServiceParam attribute*), 210
default_source (*pySim.esim.saip.personalization.EuiccMandatoryServices attribute*), 211
default_source (*pySim.esim.saip.personalization.Iccid attribute*), 211
default_source (*pySim.esim.saip.personalization.Imsi attribute*), 212
default_source (*pySim.esim.saip.personalization.MilenageRotationConstants attribute*), 213
default_source (*pySim.esim.saip.personalization.MilenageXoringConstants attribute*), 213
default_source (*pySim.esim.saip.personalization.MncLen attribute*), 213
default_source (*pySim.esim.saip.personalization.Pin attribute*), 214
default_source (*pySim.esim.saip.personalization.Puk attribute*), 215
default_source (*pySim.esim.saip.personalization.SmspTpScAddr attribute*), 219
default_source (*pySim.esim.saip.personalization.SuciCalcInfoParameter attribute*), 219
default_source (*pySim.esim.saip.personalization.SuciRi attribute*), 220
default_source (*pySim.esim.saip.personalization.TuakNumberOfKeccak attribute*), 220
delete_file() (*pySim.commands.SimCardCommands method*), 166
derive_mcc() (*in module pySim.utils*), 177
derive_milenage_opc() (*in module pySim.utils*), 178
derive_mnc() (*in module pySim.utils*), 178
disable_chv() (*pySim.commands.SimCardCommands method*), 166
disconnect() (*pySim.transport.calypso.CalypsoSimLink method*), 173
disconnect() (*pySim.transport.LinkBase method*), 171
disconnect() (*pySim.transport.modem_atcmd.ModemATCommandLink method*), 174
disconnect() (*pySim.transport.pcsc.PcscSimLink method*), 174
disconnect() (*pySim.transport.serial.SerialSimLink method*), 175
do_decode_hex() (*pySim.filesystem.LinFixedEF.ShellCommands method*), 161
do_decode_hex() (*pySim.filesystem.TransparentEF.ShellCommands method*), 164
do_delete_all() (*pySim.filesystem.BerTlvEF.ShellCommands method*), 155
do_delete_data() (*pySim.filesystem.BerTlvEF.ShellCommands method*), 155
do_edit_binary_decoded() (*pySim.filesystem.TransparentEF.ShellCommands method*), 164
do_edit_record_decoded() (*pySim.filesystem.LinFixedEF.ShellCommands method*), 161
do_read_binary() (*pySim.filesystem.TransparentEF.ShellCommands method*), 164
do_read_binary_decoded() (*pySim.filesystem.TransparentEF.ShellCommands method*), 164
do_read_record() (*pySim.filesystem.LinFixedEF.ShellCommands method*), 161
do_read_record_decoded() (*pySim.filesystem.LinFixedEF.ShellCommands method*), 161
do_read_records() (*pySim.filesystem.LinFixedEF.ShellCommands method*), 161
do_read_records_decoded() (*pySim.filesystem.LinFixedEF.ShellCommands method*), 161
do_retrieve_data() (*pySim.filesystem.BerTlvEF.ShellCommands method*), 155
do_retrieve_tags() (*pySim.filesystem.BerTlvEF.ShellCommands method*), 155
do_set_data() (*pySim.filesystem.BerTlvEF.ShellCommands method*), 155
do_update_binary() (*pySim.filesystem.TransparentEF.ShellCommands method*), 164
do_update_binary_decoded()

(*pySim.filesystem.TransparentEF.ShellCommands method*), 164
do_update_record() (*pySim.filesystem.LinFixedEF.ShellCommands method*), 161
do_update_record_decoded() (*pySim.filesystem.LinFixedEF.ShellCommands method*), 161
done() (*pySim.card_handler.CardHandlerBase method*), 180
DownloadOrder (*class in pySim.esim.es2p*), 185

E

ecdsa_dss_to_tr03111() (*in module pySim.esim.x509_cert*), 192
ecdsa_sign() (*pySim.esim.x509_cert.CertAndPrivkey method*), 191
EfUstServiceParam (*class in pySim.esim.saip.personalization*), 210
enable_chv() (*pySim.commands.SimCardCommands method*), 166
enc_imsi() (*in module pySim.utils*), 178
enc_plmn() (*in module pySim.utils*), 178
encode() (*pySim.esim.es8p.BoundProfilePackage method*), 187
encode() (*pySim.esim.http_json_api.ApiParam class method*), 190
encode() (*pySim.utils.DataObjectSequence method*), 176
encode_bin() (*pySim.filesystem.TransparentEF method*), 165
encode_client() (*pySim.esim.http_json_api.JsonHttpApiFunction method*), 190
encode_hex() (*pySim.filesystem.TransparentEF method*), 165
encode_multi() (*pySim.utils.DataObjectSequence method*), 177
encode_record_bin() (*pySim.filesystem.LinFixedEF method*), 162
encode_record_bin() (*pySim.filesystem.TransRecEF method*), 163
encode_record_hex() (*pySim.filesystem.LinFixedEF method*), 162
encode_record_hex() (*pySim.filesystem.TransRecEF method*), 163
encode_server() (*pySim.esim.http_json_api.JsonHttpApiFunction method*), 190
encrypt() (*pySim.esim.bsp.BspAlgoCrypt method*), 189
encrypt_and_mac_one() (*pySim.esim.bsp.BspInstance method*), 189
encrypt_field() (*pySim.card_key_provider.CardKeyFieldCryptor method*), 181
EnumParam (*class in pySim.esim.saip.personalization*), 210
envelope() (*pySim.commands.SimCardCommands method*), 167
eOID (*class in pySim.esim.saip.oid*), 206
ErasedStateBytesIO (*class in pySim.esim.saip*), 192
error() (*pySim.card_handler.CardHandlerBase method*), 180
Es2pApiClient (*class in pySim.esim.es2p*), 185
Es2pApiServer (*class in pySim.esim.es2p*), 185
Es2pApiServerHandlerMno (*class in pySim.esim.es2p*), 185
Es2pApiServerHandlerSmdpp (*class in pySim.esim.es2p*), 186
Es2pApiServerMno (*class in pySim.esim.es2p*), 186
Es2pApiServerSmdpp (*class in pySim.esim.es2p*), 186
Es2PlusApiFunction (*class in pySim.esim.es2p*), 185
Es9PlusApiFunction (*class in pySim.esim.es9p*), 188
EuiccMandatoryServiceGetIdentity (*class in pySim.esim.saip.personalization*), 211
EuiccMandatoryServiceParam (*class in pySim.esim.saip.personalization*), 211
EuiccMandatoryServiceProfileA (*class in pySim.esim.saip.personalization*), 211
EuiccMandatoryServiceProfileB (*class in pySim.esim.saip.personalization*), 211
expand_default_value_pattern() (*pySim.esim.saip.templates.FileTemplate method*), 221
expand_fill_pattern() (*pySim.esim.saip.File method*), 192
expand_hex() (*in module pySim.utils*), 178
export() (*pySim.filesystem.BerTlvEF static method*), 155
export() (*pySim.filesystem.CardADF static method*), 155
export() (*pySim.filesystem.CardApplication static method*), 156
export() (*pySim.filesystem.CardFile static method*), 158
export() (*pySim.filesystem.LinFixedEF static method*), 162
export() (*pySim.filesystem.TransparentEF static method*), 165
extends (*pySim.esim.saip.templates.FilesIotOptional attribute*), 222
extends (*pySim.esim.saip.templates.FilesIsimOptional attribute*), 222
extends (*pySim.esim.saip.templates.FilesIsimOptionalv2 attribute*), 222
extends (*pySim.esim.saip.templates.FilesUsimOptional attribute*), 223
extends (*pySim.esim.saip.templates.FilesUsimOptionalV2 attribute*), 223
extends (*pySim.esim.saip.templates.FilesUsimOptionalV3 attribute*), 224
extract_euiccSigned1() (*in module pySim.esim.rsp*), 184
extract_euiccSigned2() (*in module pySim.esim.rsp*), 184

F

fid_path (*pySim.esim.saip.FsNode property*), 193
File (*class in pySim.esim.saip*), 192
file2pe() (*pySim.esim.saip.FsProfileElement method*), 194
file_content_from_tuples() (*pySim.esim.saip.File method*), 192
file_content_to_tuples() (*pySim.esim.saip.File method*), 193
file_replace_content() (*in module pySim.esim.saip.personalization*), 220
file_size (*pySim.esim.saip.File property*), 193
file_template_for_path() (*pySim.esim.saip.FsProfileElement method*), 194
file_template_for_path() (*pySim.esim.saip.ProfileElementIoT method*), 200
file_template_for_path() (*pySim.esim.saip.ProfileElementOptIoT method*), 201
FileCheckBasicStructure (*class in pySim.esim.saip.validation*), 225
FileError, 225
files2pe() (*pySim.esim.saip.FsProfileElement method*), 195
files2pe() (*pySim.esim.saip.ProfileElementGFM method*), 199
FilesAtMF (*class in pySim.esim.saip.templates*), 222
FilesCD (*class in pySim.esim.saip.templates*), 222
FilesDf5GProSe (*class in pySim.esim.saip.templates*), 222
FilesDfSnpn (*class in pySim.esim.saip.templates*), 222
FilesEap (*class in pySim.esim.saip.templates*), 222
FilesIotMandatory (*class in pySim.esim.saip.templates*), 222
FilesIotOptional (*class in pySim.esim.saip.templates*), 222
FilesIsimMandatory (*class in pySim.esim.saip.templates*), 222
FilesIsimOptional (*class in pySim.esim.saip.templates*), 222
FilesIsimOptionalv2 (*class in pySim.esim.saip.templates*), 222
FilesTelecom (*class in pySim.esim.saip.templates*), 222
FilesTelecomV2 (*class in pySim.esim.saip.templates*), 222
FilesUsimDf5GS (*class in pySim.esim.saip.templates*), 222
FilesUsimDf5GSv2 (*class in pySim.esim.saip.templates*), 223
FilesUsimDf5GSv3 (*class in pySim.esim.saip.templates*), 223
FilesUsimDf5GSv4 (*class in pySim.esim.saip.templates*), 223
FilesUsimDfGsmAccess (*class in pySim.esim.saip.templates*), 223
FilesUsimDfPhonebook (*class in pySim.esim.saip.templates*), 223
FilesUsimDfSaip (*class in pySim.esim.saip.templates*), 223
FilesUsimMandatory (*class in pySim.esim.saip.templates*), 223
FilesUsimMandatoryV2 (*class in pySim.esim.saip.templates*), 223
FilesUsimOptional (*class in pySim.esim.saip.templates*), 223
FilesUsimOptionalV2 (*class in pySim.esim.saip.templates*), 223
FilesUsimOptionalV3 (*class in pySim.esim.saip.templates*), 223
FileTemplate (*class in pySim.esim.saip.templates*), 221
find_key() (*pySim.esim.saip.ProfileElementSD method*), 203
for_version() (*pySim.esim.saip.templates.SaipSpecVersion static method*), 224
fork_lchan() (*pySim.commands.SimCardCommands method*), 167
from_bitstring() (*pySim.esim.PMO class method*), 184
from_bytes() (*pySim.utils.DataObject method*), 175
from_bytes() (*pySim.utils.TL0_DataObject method*), 177
from_der() (*pySim.esim.es8p.UnprotectedProfilePackage class method*), 187
from_der() (*pySim.esim.saip.ProfileElement class method*), 195
from_der() (*pySim.esim.saip.ProfileElementSequence class method*), 204
from_file() (*pySim.esim.saip.ProfileElementApplication class method*), 197
from_fileDescriptor() (*pySim.esim.saip.File method*), 193
from_int() (*pySim.esim.PMO class method*), 184
from_kdf() (*pySim.esim.bsp.BspInstance class method*), 189
from_matchlist() (*pySim.esim.saip.NonMatch class method*), 195
from_saip_dict() (*pySim.esim.saip.SecurityDomainKey class method*), 206
from_saip_dict() (*pySim.esim.saip.SecurityDomainKeyComponent class method*), 206
from_string() (*pySim.esim.ActivationCode class method*), 183
from_template() (*pySim.esim.saip.File method*), 193
from_tlv() (*pySim.utils.DataObject method*), 176
from_tuples() (*pySim.esim.saip.File method*), 193
from_upp() (*pySim.esim.es8p.ProtectedProfilePackage class method*), 187

FsNode (*class in pySim.esim.saip*), 193
FsNodeADF (*class in pySim.esim.saip*), 193
FsNodeDF (*class in pySim.esim.saip*), 194
FsNodeEF (*class in pySim.esim.saip*), 194
FsNodeMF (*class in pySim.esim.saip*), 194
FsProfileElement (*class in pySim.esim.saip*), 194
fully_qualified_path() (*pySim.filesystem.CardFile method*), 158
fully_qualified_path_fobj() (*pySim.filesystem.CardFile method*), 158
fully_qualified_path_str() (*pySim.filesystem.CardFile method*), 158

G

gen_init_sec_chan_signed_part() (*in module pySim.esim.es8p*), 187
gen_initialiseSecureChannel() (*in module pySim.esim.es8p*), 187
gen_replace_session_keys() (*in module pySim.esim.es8p*), 187
gen_store_metadata_request() (*pySim.esim.es8p.ProfileMetadata method*), 187
generate_sd_key_classes() (*pySim.esim.saip.personalization.SdKey class method*), 215
get() (*pySim.card_handler.CardHandlerBase method*), 180
get() (*pySim.card_key_provider.CardKeyProvider method*), 181
get() (*pySim.card_key_provider.CardKeyProviderCsv method*), 182
get() (*pySim.card_key_provider.CardKeyProviderPgsql method*), 182
get_addr_type() (*in module pySim.utils*), 178
get_app_names() (*pySim.filesystem.CardMF method*), 159
get_app_selectables() (*pySim.filesystem.CardMF method*), 159
get_atr() (*pySim.commands.SimCardCommands method*), 167
get_atr() (*pySim.transport.calypso.CalypsoSimLink method*), 173
get_atr() (*pySim.transport.LinkBase method*), 171
get_atr() (*pySim.transport.modem_atcmd.ModemATCommandLink method*), 174
get_atr() (*pySim.transport.pcsc.PcscSimLink method*), 174
get_atr() (*pySim.transport.serial.SerialSimLink method*), 175
get_authority_key_identifier() (*pySim.esim.x509_cert.CertAndPrivkey method*), 191
get_by_oid() (*pySim.esim.saip.templates.ProfileTemplateRegistry class method*), 224
get_cert_as_der() (*pySim.esim.x509_cert.CertAndPrivkey method*), 191
get_closest_prev_pe_for_templateID() (*pySim.esim.saip.ProfileElementSequence method*), 204
get_file_by_path() (*pySim.esim.saip.templates.FileTemplate method*), 221
get_index_by_pe() (*pySim.esim.saip.ProfileElementSequence method*), 204
get_index_by_type() (*pySim.esim.saip.ProfileElementSequence method*), 204
get_len_range() (*pySim.esim.saip.personalization.ConfigurableParameter class method*), 209
get_mf() (*pySim.filesystem.CardFile method*), 158
get_name() (*pySim.esim.saip.personalization.ConfigurableParameter class method*), 209
get_pe_for_type() (*pySim.esim.saip.ProfileElementSequence method*), 204
get_pes_for_templateID() (*pySim.esim.saip.ProfileElementSequence method*), 204
get_pes_for_type() (*pySim.esim.saip.ProfileElementSequence method*), 204
get_profile() (*pySim.filesystem.CardFile method*), 158
get_selectable_names() (*pySim.filesystem.CardFile method*), 158
get_selectables() (*pySim.filesystem.CardDF method*), 156
get_selectables() (*pySim.filesystem.CardEF method*), 157
get_selectables() (*pySim.filesystem.CardFile method*), 158
get_selectables() (*pySim.filesystem.CardMF method*), 159
get_subject_alt_name() (*pySim.esim.x509_cert.CertAndPrivkey method*), 191
get_tuplelist_item() (*pySim.esim.saip.File static method*), 193
get_typical_input_len() (*pySim.esim.saip.personalization.BinaryParam class method*), 207
get_typical_input_len() (*pySim.esim.saip.personalization.ConfigurableParameter class method*), 209
get_value_from_pes() (*pySim.esim.saip.personalization.ConfigurableParameter class method*), 209
get_values_from_pes() (*pySim.esim.saip.personalization.AlgoConfig class method*), 206
get_values_from_pes()

(*pySim.esim.saip.personalization.AlgorithmID class method*), 207
get_values_from_pes() (*pySim.esim.saip.personalization.ConfigurableParameter class method*), 209
get_values_from_pes() (*pySim.esim.saip.personalization.EfUstServiceParam class method*), 210
get_values_from_pes() (*pySim.esim.saip.personalization.EuiccMandatoryServiceParam class method*), 211
get_values_from_pes() (*pySim.esim.saip.personalization.Iccid class method*), 211
get_values_from_pes() (*pySim.esim.saip.personalization.Imsi class method*), 212
get_values_from_pes() (*pySim.esim.saip.personalization.IntegerParam class method*), 212
get_values_from_pes() (*pySim.esim.saip.personalization.MncLen class method*), 213
get_values_from_pes() (*pySim.esim.saip.personalization.Pin class method*), 214
get_values_from_pes() (*pySim.esim.saip.personalization.Pin2 class method*), 214
get_values_from_pes() (*pySim.esim.saip.personalization.Puk class method*), 215
get_values_from_pes() (*pySim.esim.saip.personalization.SdKey class method*), 217
get_values_from_pes() (*pySim.esim.saip.personalization.SmspTpScAddr class method*), 219
get_values_from_pes() (*pySim.esim.saip.personalization.SuciCalcInfoParameter class method*), 219
get_values_from_pes() (*pySim.esim.saip.personalization.SuciRi class method*), 220
GetBoundProfilePackage (*class in pySim.esim.es9p*), 188

H

HandleDownloadProgressInfo (*class in pySim.esim.es2p*), 186
HandleNotification (*class in pySim.esim.es9p*), 188
has_scp() (*pySim.esim.saip.ProfileElementSD method*), 203
header (*pySim.esim.saip.ProfileElement property*), 196
header_name (*pySim.esim.saip.ProfileElement property*), 196
HttpHeaderError, 190
HttpStatusError, 190

I

Iccid (*class in pySim.esim.saip.personalization*), 211
iccid (*pySim.esim.saip.ProfileElementSequence property*), 204
identification (*pySim.esim.saip.ProfileElement property*), 196
Imsi (*class in pySim.esim.saip.personalization*), 212
init_reader() (*in module pySim.transport*), 173
InitiateAuthentication (*class in pySim.esim.es9p*), 188
insert_after_pe() (*pySim.esim.saip.ProfileElementSequence method*), 204
insert_at_index() (*pySim.esim.saip.ProfileElementSequence method*), 204
IntegerParam (*class in pySim.esim.saip.personalization*), 212
interpret_sw() (*in module pySim.filesystem*), 165
interpret_sw() (*pySim.filesystem.CardApplication method*), 156
is_parent() (*pySim.filesystem.Path method*), 162

J

JsonEditor (*class in pySim.filesystem*), 160
JsonHttpApiFunction (*class in pySim.esim.http_json_api*), 190
JsonRequestHeader (*class in pySim.esim.http_json_api*), 191
JsonResponseHeader (*class in pySim.esim.http_json_api*), 191

K

K (*class in pySim.esim.saip.personalization*), 213

L

lchan_nr_to_cla() (*in module pySim.commands*), 171
LinFixedEF (*class in pySim.filesystem*), 160
LinFixedEF.ShellCommands (*class in pySim.filesystem*), 160
LinkBase (*class in pySim.transport*), 171
LinkBaseTpdu (*class in pySim.transport*), 172
lookup() (*pySim.utils.CardCommandSet method*), 175
lookup_by_fidpath() (*pySim.esim.saip.FsNodeDF method*), 194
lookup_by_path() (*pySim.esim.saip.FsNodeDF method*), 194
lookup_file_by_fid() (*pySim.filesystem.CardDF method*), 157
lookup_file_by_name() (*pySim.filesystem.CardDF method*), 157

lookup_file_by_sfid() (*pySim.filesystem.CardDF method*), 157

M

mac_only_one() (*pySim.esim.bsp.BspInstance method*), 189
manage_channel() (*pySim.commands.SimCardCommands method*), 167
match() (*pySim.filesystem.CardModel class method*), 160
match_cla() (*pySim.utils.CardCommand method*), 175
max_cmd_len (*pySim.commands.SimCardCommands property*), 167
mcc_from_imsi() (*in module pySim.utils*), 178
mf (*pySim.esim.saip.FsNode property*), 193
MilenageRotationConstants (*class in pySim.esim.saip.personalization*), 213
MilenageXoringConstants (*class in pySim.esim.saip.personalization*), 213
mnc_from_imsi() (*in module pySim.utils*), 178
MncLen (*class in pySim.esim.saip.personalization*), 213
ModemATCommandLink (*class in pySim.transport.modem_atcmd*), 173
module
  pySim.card_handler, 180
  pySim.card_key_provider, 180
  pySim.commands, 166
  pySim.esim, 183
  pySim.esim.bsp, 189
  pySim.esim.es2p, 184
  pySim.esim.es8p, 187
  pySim.esim.es9p, 188
  pySim.esim.http_json_api, 189
  pySim.esim.rsp, 184
  pySim.esim.saip, 192
  pySim.esim.saip.oid, 206
  pySim.esim.saip.personalization, 206
  pySim.esim.saip.templates, 221
  pySim.esim.saip.validation, 225
  pySim.esim.x509_cert, 191
  pySim.exceptions, 179
  pySim.filesystem, 154
  pySim.transport, 171
  pySim.transport.calypso, 173
  pySim.transport.modem_atcmd, 173
  pySim.transport.pcsc, 174
  pySim.transport.serial, 174
  pySim.utils, 175

N

Naa (*class in pySim.esim.saip*), 195
naa_for_path() (*pySim.esim.saip.ProfileElementSequence static method*), 205
NaaCsim (*class in pySim.esim.saip*), 195
NaaIsim (*class in pySim.esim.saip*), 195
NaaUsim (*class in pySim.esim.saip*), 195
name_path (*pySim.esim.saip.FsNode property*), 193
nested_collection_cls (*pySim.esim.saip.ProfileElementSD.C9 attribute*), 203
NoCardError, 179
NonMatch (*class in pySim.esim.saip*), 195
normalizeATR() (*in module pySim.utils*), 178

O

Opc (*class in pySim.esim.saip.personalization*), 214

P

parent (*pySim.esim.saip.templates.FilesDf5GProSe attribute*), 222
parent (*pySim.esim.saip.templates.FilesDfSnpn attribute*), 222
parent (*pySim.esim.saip.templates.FilesUsimDf5GS attribute*), 223
parent (*pySim.esim.saip.templates.FilesUsimDf5GSv2 attribute*), 223
parent (*pySim.esim.saip.templates.FilesUsimDf5GSv3 attribute*), 223
parent (*pySim.esim.saip.templates.FilesUsimDf5GSv4 attribute*), 223
parent (*pySim.esim.saip.templates.FilesUsimDfGsmAccess attribute*), 223
parent (*pySim.esim.saip.templates.FilesUsimDfSaip attribute*), 223
parse_command_apdu() (*in module pySim.utils*), 178
parse_der() (*pySim.esim.saip.ProfileElementSequence method*), 205
Path (*class in pySim.filesystem*), 162
path (*pySim.esim.saip.templates.FileTemplate property*), 221
path_from_gfm() (*pySim.esim.saip.File static method*), 193
path_to_gfm() (*pySim.esim.saip.File static method*), 193
PcscSimLink (*class in pySim.transport.pcsc*), 174
pe2files() (*pySim.esim.saip.FsProfileElement method*), 195
pe2files() (*pySim.esim.saip.ProfileElementGFM method*), 199
pe_for_path() (*pySim.esim.saip.ProfileElementSequence method*), 205
peclass_for_path() (*pySim.esim.saip.ProfileElementSequence static method*), 205
Pin (*class in pySim.esim.saip.personalization*), 214
Pin1 (*class in pySim.esim.saip.personalization*), 214
Pin2 (*class in pySim.esim.saip.personalization*), 214
PMO (*class in pySim.esim*), 184
print_tree() (*pySim.esim.saip.templates.FileTemplate method*), 222
ProactiveHandler (*class in pySim.transport*), 172

ProfileConstraintChecker (*class in pySim.esim.saip.validation*), 225
ProfileElement (*class in pySim.esim.saip*), 195
ProfileElementAKA (*class in pySim.esim.saip*), 196
ProfileElementApplication (*class in pySim.esim.saip*), 196
ProfileElementCD (*class in pySim.esim.saip*), 197
ProfileElementCDMA (*class in pySim.esim.saip*), 197
ProfileElementDf5GProSe (*class in pySim.esim.saip*), 197
ProfileElementDf5GS (*class in pySim.esim.saip*), 198
ProfileElementDfSAIP (*class in pySim.esim.saip*), 198
ProfileElementDfSNPN (*class in pySim.esim.saip*), 198
ProfileElementEAP (*class in pySim.esim.saip*), 198
ProfileElementEnd (*class in pySim.esim.saip*), 198
ProfileElementGFM (*class in pySim.esim.saip*), 199
ProfileElementGsmAccess (*class in pySim.esim.saip*), 199
ProfileElementHeader (*class in pySim.esim.saip*), 199
ProfileElementIoT (*class in pySim.esim.saip*), 200
ProfileElementISIM (*class in pySim.esim.saip*), 200
ProfileElementMF (*class in pySim.esim.saip*), 200
ProfileElementNonStandard (*class in pySim.esim.saip*), 200
ProfileElementOptIoT (*class in pySim.esim.saip*), 201
ProfileElementOptISIM (*class in pySim.esim.saip*), 201
ProfileElementOptUSIM (*class in pySim.esim.saip*), 201
ProfileElementPhonebook (*class in pySim.esim.saip*), 202
ProfileElementPin (*class in pySim.esim.saip*), 202
ProfileElementPuk (*class in pySim.esim.saip*), 202
ProfileElementRFM (*class in pySim.esim.saip*), 202
ProfileElementSD (*class in pySim.esim.saip*), 203
ProfileElementSD.C9 (*class in pySim.esim.saip*), 203
ProfileElementSequence (*class in pySim.esim.saip*), 204
ProfileElementSSD (*class in pySim.esim.saip*), 203
ProfileElementTelecom (*class in pySim.esim.saip*), 205
ProfileElementUSIM (*class in pySim.esim.saip*), 205
ProfileError, 225
ProfileMetadata (*class in pySim.esim.es8p*), 187
ProfileTemplate (*class in pySim.esim.saip.templates*), 224
ProfileTemplateRegistry (*class in pySim.esim.saip.templates*), 224
ProtectedProfilePackage (*class in pySim.esim.es8p*), 187
ProtocolError, 179
Puk (*class in pySim.esim.saip.personalization*), 215
Puk1 (*class in pySim.esim.saip.personalization*), 215
Puk2 (*class in pySim.esim.saip.personalization*), 215
pySim.card_handler
    module, 180
pySim.card_key_provider
    module, 180
pySim.commands
    module, 166
pySim.esim
    module, 183
pySim.esim.bsp
    module, 189
pySim.esim.es2p
    module, 184
pySim.esim.es8p
    module, 187
pySim.esim.es9p
    module, 188
pySim.esim.http_json_api
    module, 189
pySim.esim.rsp
    module, 184
pySim.esim.saip
    module, 192
pySim.esim.saip.oid
    module, 206
pySim.esim.saip.personalization
    module, 206
pySim.esim.saip.templates
    module, 221
pySim.esim.saip.validation
    module, 225
pySim.esim.x509_cert
    module, 191
pySim.exceptions
    module, 179
pySim.filesystem
    module, 154
pySim.transport
    module, 171
pySim.transport.calypso
    module, 173
pySim.transport.modem_atcmd
    module, 173
pySim.transport.pcsc
    module, 174
pySim.transport.serial
    module, 174
pySim.utils
    module, 175

R

read_binary() (*pySim.commands.SimCardCommands method*), 167
read_record() (*pySim.commands.SimCardCommands method*), 167
ReaderError, 179
rebuild_mandatory_gfstelist() (*pySim.esim.saip.ProfileElementSequence*

method), 205
rebuild_mandatory_services() (*pySim.esim.saip.ProfileElementSequence method*), 205
receive_fetch() (*pySim.transport.ProactiveHandler method*), 173
record_count() (*pySim.commands.SimCardCommands method*), 167
record_size() (*pySim.commands.SimCardCommands method*), 167
relative_to_mf() (*pySim.filesystem.Path method*), 162
ReleaseProfile (*class in pySim.esim.es2p*), 186
remove_instance() (*pySim.esim.saip.ProfileElementApplicationDataObject method*), 197
remove_naas_of_type() (*pySim.esim.saip.ProfileElementSequence method*), 205
remove_scp() (*pySim.esim.saip.ProfileElementSD method*), 203
remove_unwanted_tuples_from_list() (*in module pySim.esim.saip.personalization*), 221
renumber_identification() (*pySim.esim.saip.ProfileElementSequence method*), 205
reset_card() (*pySim.commands.SimCardCommands method*), 167
reset_card() (*pySim.transport.LinkBase method*), 171
resize_file() (*pySim.commands.SimCardCommands method*), 168
resume_uicc() (*pySim.commands.SimCardCommands method*), 168
retrieve_data() (*pySim.commands.SimCardCommands method*), 168
rewrite_url() (*pySim.esim.http_json_api.JsonHttpApiFunction method*), 191
RspSessionState (*class in pySim.esim.rsp*), 184
RspSessionStore (*class in pySim.esim.rsp*), 184
run_gsm() (*pySim.commands.SimCardCommands method*), 168

S

SaipSpecVersion (*class in pySim.esim.saip.templates*), 224
SaipSpecVersion101 (*class in pySim.esim.saip.templates*), 224
SaipSpecVersion20 (*class in pySim.esim.saip.templates*), 224
SaipSpecVersion21 (*class in pySim.esim.saip.templates*), 224
SaipSpecVersion22 (*class in pySim.esim.saip.templates*), 224
SaipSpecVersion23 (*class in pySim.esim.saip.templates*), 224
SaipSpecVersion231 (*class in pySim.esim.saip.templates*), 224
SaipSpecVersion31 (*class in pySim.esim.saip.templates*), 224
SaipSpecVersion32 (*class in pySim.esim.saip.templates*), 224
SaipSpecVersion331 (*class in pySim.esim.saip.templates*), 224
sanitize_pin_adm() (*in module pySim.utils*), 179
SdKey (*class in pySim.esim.saip.personalization*), 215
SdKeyScp02Kvn20AesDek (*class in pySim.esim.saip.personalization*), 217
SdKeyScp02Kvn20AesEnc (*class in pySim.esim.saip.personalization*), 217
SdKeyScp02Kvn20AesMac (*class in pySim.esim.saip.personalization*), 217
SdKeyScp02Kvn21AesDek (*class in pySim.esim.saip.personalization*), 217
SdKeyScp02Kvn21AesEnc (*class in pySim.esim.saip.personalization*), 217
SdKeyScp02Kvn21AesMac (*class in pySim.esim.saip.personalization*), 217
SdKeyScp02Kvn22AesDek (*class in pySim.esim.saip.personalization*), 217
SdKeyScp02Kvn22AesEnc (*class in pySim.esim.saip.personalization*), 217
SdKeyScp02Kvn22AesMac (*class in pySim.esim.saip.personalization*), 217
SdKeyScp02KvnffAesDek (*class in pySim.esim.saip.personalization*), 217
SdKeyScp02KvnffAesEnc (*class in pySim.esim.saip.personalization*), 217
SdKeyScp02KvnffAesMac (*class in pySim.esim.saip.personalization*), 217
SdKeyScp03Kvn30AesDek (*class in pySim.esim.saip.personalization*), 217
SdKeyScp03Kvn30AesEnc (*class in pySim.esim.saip.personalization*), 217
SdKeyScp03Kvn30AesMac (*class in pySim.esim.saip.personalization*), 217
SdKeyScp03Kvn31AesDek (*class in pySim.esim.saip.personalization*), 217
SdKeyScp03Kvn31AesEnc (*class in pySim.esim.saip.personalization*), 217
SdKeyScp03Kvn31AesMac (*class in pySim.esim.saip.personalization*), 218
SdKeyScp03Kvn32AesDek (*class in pySim.esim.saip.personalization*), 218
SdKeyScp03Kvn32AesEnc (*class in pySim.esim.saip.personalization*), 218
SdKeyScp03Kvn32AesMac (*class in pySim.esim.saip.personalization*), 218
SdKeyScp80Kvn01AesDek (*class in pySim.esim.saip.personalization*), 218

SdKeyScp80Kvn01AesEnc (*class in pySim.esim.saip.personalization*), 218
SdKeyScp80Kvn01AesMac (*class in pySim.esim.saip.personalization*), 218
SdKeyScp80Kvn01DesDek (*class in pySim.esim.saip.personalization*), 218
SdKeyScp80Kvn01DesEnc (*class in pySim.esim.saip.personalization*), 218
SdKeyScp80Kvn01DesMac (*class in pySim.esim.saip.personalization*), 218
SdKeyScp80Kvn02AesDek (*class in pySim.esim.saip.personalization*), 218
SdKeyScp80Kvn02AesEnc (*class in pySim.esim.saip.personalization*), 218
SdKeyScp80Kvn02AesMac (*class in pySim.esim.saip.personalization*), 218
SdKeyScp80Kvn02DesDek (*class in pySim.esim.saip.personalization*), 218
SdKeyScp80Kvn02DesEnc (*class in pySim.esim.saip.personalization*), 218
SdKeyScp80Kvn02DesMac (*class in pySim.esim.saip.personalization*), 218
SdKeyScp80Kvn03AesDek (*class in pySim.esim.saip.personalization*), 218
SdKeyScp80Kvn03AesEnc (*class in pySim.esim.saip.personalization*), 218
SdKeyScp80Kvn03AesMac (*class in pySim.esim.saip.personalization*), 218
SdKeyScp80Kvn03DesDek (*class in pySim.esim.saip.personalization*), 218
SdKeyScp80Kvn03DesEnc (*class in pySim.esim.saip.personalization*), 218
SdKeyScp80Kvn03DesMac (*class in pySim.esim.saip.personalization*), 218
SdKeyScp81Kvn40AesDek (*class in pySim.esim.saip.personalization*), 218
SdKeyScp81Kvn40DesDek (*class in pySim.esim.saip.personalization*), 218
SdKeyScp81Kvn40Tlspsk (*class in pySim.esim.saip.personalization*), 218
SdKeyScp81Kvn41AesDek (*class in pySim.esim.saip.personalization*), 218
SdKeyScp81Kvn41DesDek (*class in pySim.esim.saip.personalization*), 218
SdKeyScp81Kvn41Tlspsk (*class in pySim.esim.saip.personalization*), 218
SdKeyScp81Kvn42AesDek (*class in pySim.esim.saip.personalization*), 218
SdKeyScp81Kvn42DesDek (*class in pySim.esim.saip.personalization*), 218
SdKeyScp81Kvn42Tlspsk (*class in pySim.esim.saip.personalization*), 218
SecurityDomainKey (*class in pySim.esim.saip*), 206
SecurityDomainKeyComponent (*class in pySim.esim.saip*), 206
select_adf() (*pySim.commands.SimCardCommands method*), 168
select_file() (*pySim.commands.SimCardCommands method*), 168
select_parent_df() (*pySim.commands.SimCardCommands method*), 168
select_path() (*pySim.commands.SimCardCommands method*), 168
send_apdu() (*pySim.commands.SimCardCommands method*), 168
send_apdu() (*pySim.transport.LinkBase method*), 172
send_apdu_checksw() (*pySim.commands.SimCardCommands method*), 168
send_apdu_checksw() (*pySim.transport.LinkBase method*), 172
send_apdu_constr() (*pySim.commands.SimCardCommands method*), 169
send_apdu_constr_checksw() (*pySim.commands.SimCardCommands method*), 169
send_tpdu() (*pySim.transport.calypso.CalypsoSimLink method*), 173
send_tpdu() (*pySim.transport.LinkBaseTpdu method*), 172
send_tpdu() (*pySim.transport.modem_atcmd.ModemATCommandLink method*), 174
send_tpdu() (*pySim.transport.pcsc.PcscSimLink method*), 174
send_tpdu() (*pySim.transport.serial.SerialSimLink method*), 175
SerialSimLink (*class in pySim.transport.serial*), 174
set_data() (*pySim.commands.SimCardCommands method*), 170
set_icon() (*pySim.esim.es8p.ProfileMetadata method*), 187
set_mapping() (*pySim.esim.saip.ProfileElementAKA method*), 196
set_milenage() (*pySim.esim.saip.ProfileElementAKA method*), 196
set_sw_interpreter() (*pySim.transport.LinkBase method*), 172
set_tpdu_format() (*pySim.transport.LinkBaseTpdu method*), 172
set_tuak() (*pySim.esim.saip.ProfileElementAKA method*), 196
set_xor3g() (*pySim.esim.saip.ProfileElementAKA method*), 196
should_exist_for_services() (*pySim.filesystem.CardFile method*), 159
SimCardCommands (*class in pySim.commands*), 166
SmdpAddress (*class in pySim.esim.http_json_api*), 191
SmspTpScAddr (*class in*

pySim.esim.saip.personalization), 219
status() (*pySim.commands.SimCardCommands method*), 170
StdoutApduTracer (*class in pySim.transport*), 173
SuciActive (*class in pySim.esim.saip.personalization*), 219
SuciCalcInfoParameter (*class in pySim.esim.saip.personalization*), 219
SuciCalcInfoUe (*class in pySim.esim.saip.personalization*), 220
SuciCalcInfoUsim (*class in pySim.esim.saip.personalization*), 220
SuciInUsim (*class in pySim.esim.saip.personalization*), 220
SuciRi (*class in pySim.esim.saip.personalization*), 220
suports_template_OID() (*pySim.esim.saip.templates.SaipSpecVersion class method*), 224
supports_file_for_path() (*pySim.esim.saip.FsProfileElement method*), 195
supports_file_for_path() (*pySim.esim.saip.ProfileElementGFM method*), 199
suspend_uicc() (*pySim.commands.SimCardCommands method*), 170
sw_match() (*in module pySim.utils*), 179
SwMatchError, 179
sync() (*pySim.esim.rsp.RspSessionStore method*), 184

T

tabulate_str_list() (*in module pySim.utils*), 179
templateID (*pySim.esim.saip.ProfileElement property*), 196
terminal_profile() (*pySim.commands.SimCardCommands method*), 170
terminate_card_usage() (*pySim.commands.SimCardCommands method*), 170
terminate_df() (*pySim.commands.SimCardCommands method*), 170
terminate_ef() (*pySim.commands.SimCardCommands method*), 170
TL0_DataObject (*class in pySim.utils*), 177
to_bitstring() (*pySim.esim.PMO method*), 184
to_bytes() (*pySim.utils.DataObject method*), 176
to_bytes() (*pySim.utils.TL0_DataObject method*), 177
to_der() (*pySim.esim.es8p.UnprotectedProfilePackage method*), 187
to_der() (*pySim.esim.saip.ProfileElement method*), 196
to_der() (*pySim.esim.saip.ProfileElementSequence method*), 205
to_dict() (*pySim.utils.DataObject method*), 176
to_file() (*pySim.esim.saip.ProfileElementApplication method*), 197
to_fileDescriptor() (*pySim.esim.saip.File method*), 193
to_gfm() (*pySim.esim.saip.File method*), 193
to_qrcode() (*pySim.esim.ActivationCode method*), 183
to_saip_dict() (*pySim.esim.saip.SecurityDomainKey method*), 206
to_saip_dict() (*pySim.esim.saip.SecurityDomainKeyComponent method*), 206
to_string() (*pySim.esim.ActivationCode method*), 184
to_tlv() (*pySim.utils.DataObject method*), 176
to_tuples() (*pySim.esim.saip.File method*), 193
TransparentEF (*class in pySim.filesystem*), 164
TransparentEF.ShellCommands (*class in pySim.filesystem*), 164
transport_keys_from_opts() (*pySim.card_key_provider.CardKeyFieldCryptor static method*), 181
TransRecEF (*class in pySim.filesystem*), 162
try_select_path() (*pySim.commands.SimCardCommands method*), 170
TuakNumberOfKeccak (*class in pySim.esim.saip.personalization*), 220

U

unblock_chv() (*pySim.commands.SimCardCommands method*), 170
UnprotectedProfilePackage (*class in pySim.esim.es8p*), 187
update_binary() (*pySim.commands.SimCardCommands method*), 170
update_record() (*pySim.commands.SimCardCommands method*), 171

V

validate() (*pySim.esim.saip.personalization.ConfigurableParameter method*), 209
validate_val() (*pySim.esim.saip.personalization.BinaryParam class method*), 207
validate_val() (*pySim.esim.saip.personalization.ConfigurableParameter class method*), 209
validate_val() (*pySim.esim.saip.personalization.DecimalHexParam class method*), 210
validate_val() (*pySim.esim.saip.personalization.DecimalParam class method*), 210
validate_val() (*pySim.esim.saip.personalization.EnumParam class method*), 210
validate_val() (*pySim.esim.saip.personalization.Iccid class method*), 212
validate_val() (*pySim.esim.saip.personalization.IntegerParam class method*), 213
validate_val() (*pySim.esim.saip.personalization.MilenageRotationConstan class method*), 213

validate_val() (*pySim.esim.saip.personalization.SmspTpScAddr class method*), 219
validate_val() (*pySim.esim.saip.personalization.SuciCalcInfoParameter class method*), 220
verify_cert_chain() (*pySim.esim.x509_cert.CertificateSet method*), 191
verify_chv() (*pySim.commands.SimCardCommands method*), 171
verify_decoded() (*pySim.esim.http_json_api.ApiParam class method*), 190
verify_decoded() (*pySim.esim.http_json_api.ApiParamInteger class method*), 190
verify_decoded() (*pySim.esim.http_json_api.JsonRequestHeader class method*), 191
verify_decoded() (*pySim.esim.http_json_api.JsonResponseHeader class method*), 191
verify_encoded() (*pySim.esim.http_json_api.ApiParam class method*), 190
verify_encoded() (*pySim.esim.http_json_api.ApiParamFqdn class method*), 190
verify_encoded() (*pySim.esim.http_json_api.ApiParamInteger class method*), 190
verify_luhn() (*in module pySim.utils*), 179
VerifyError, 191
version_match() (*pySim.esim.saip.templates.SaipSpecVersion class method*), 224

W

wait_for_card() (*pySim.transport.calypso.CalypsoSimLink method*), 173
wait_for_card() (*pySim.transport.LinkBase method*), 172
wait_for_card() (*pySim.transport.modem_atcmd.ModemATCommandLink method*), 174
wait_for_card() (*pySim.transport.pcsc.PcscSimLink method*), 174
wait_for_card() (*pySim.transport.serial.SerialSimLink method*), 175
walk() (*pySim.esim.saip.FsNode method*), 193
walk() (*pySim.esim.saip.FsNodeDF method*), 194
wrap_as_der_tlv() (*in module pySim.esim.es8p*), 187
write() (*pySim.esim.saip.ErasedStateBytesIO method*), 192