SpacePyLibrary

File name:	SpacePyLibrary.doc
Date:	2016-04-25

Contents:

1 Introduction

1.1 Purpose

The *SpacePyLibrary* is a Python library that can be used for building lightweight applications in the Space domain. The implementation of the library is driven by the needs of different project in that domain.

Two applications are bundled together with the library to give examples for the library usage:

- ***SIM*:** A simulator that can be connected to ESA's Mission Control System SCOS-2000. It simulates a groundstation, the space link, and some generic functionality of a spacecraft's onboard computer.
- ***SCOE*:** A simulator that can be connected to a Central Checkout System. It simulates some generic functionality of a Frontend Checkout Equipment.

1.2 Referenced Documents

The following documents are referenced or used as standards for the implementation of the *SpacePyLibrary*:

Ref	Description	ID	Issue	Date
CLTU	Telecommand – channel service	CCSDS 201.0-B-3	3	2000-06
COP	Communicatons Operation Procedure-1	CCSDS 232.1-B-1	1	2003-09
MIB	SCOS-2000 Database Import ICD	EGOS-MCS-S2K-ICD-0001	6.9	2010-07-06
PUS	Ground systems and operations – Telemetry and telecommand packet utilization	ECSS-E-70-41A		2003-01-30
TC	Telecommand – data routing service	CCSDS 202.0-B-3	3	2000-11
TIME	Time Code Formats	CCSDS 301.0-B-3	3	2002-01
TM	Packet Telemetry	CCSDS 102.0-B-5	5	2000-11

2 Pre-requisites

2.1 Software Packages

The *SpycePyLibrary* has been developed and tested on SuSE Linux 8.2 and then ported to other operating systems. The following python packages have been used for validating the *SpycePyLibrary*:

Platform	Python package	Version
SuSE Linux 8.2	python	2.2.2
	python-tk	2.2.2
SLES 10 / SP2	ActivePython	2.7.1.4
SLES 11 / SP2	python	2.6.0
	python-tk	2.6.0
Windows XP / SP2	ActivePython	2.7.1.4
OS-X 10.11.4	Python	2.7.10
	Tkinter	8.5.9

2.2 Configuration

The *SpacePyLibrary* is implemented in pure python and does not share any source code with other systems. Even though there are some aspects that make the *SpacePyLibrary* dependant from ESA's mission control system SCOS-2000:

- The *SIM* application – which is based on the *SpacePyLibrary* – implements communication interfaces that are compatible with the SCOS-2000 TM/TC backend interface (NCTRS / NIS).
- The *SpacePyLibrary* uses some configuration variables and configuration files in SCOS-2000 format. The reason for this design concept was to simplify the installation of *SpacePyLibrary* within an existing SCOS-2000 environment.

The following configuration items are used by the *SpacePyLibrary*:

Environment variable	Description
HOST	Hostname of the SCOS-2000 runtime account
scosii_homedir	Location of the SCOS-2000 runtime directory
NCTRS_ADMIN_SERVER_PORT	NCTRS admin server port of the TC releaser. This variable is set in start script *SIM*.
NCTRS_TC_SERVER_PORT	NCTRS TC server port of the TC releaser. This variable is set in start script *SIM*.
DEF_GROUND_STATION_ID	Groundstation ID, used for the NCTRS TC connection to the TC releaser. This variable is set in start script *SIM*.
GROUND_STATION_NAME	Groundstaton name, used for the NCTRS admin connection to the TC releaser. This variable is set in start script *SIM*.

Environment variable	Description
TC_TT_PKT_BYTE_OFFSET	If the TC packet is time tagged (execution time > now), then this variable is used to determine the offset of the TC packet within the time tagged uplink packet. This variable is set in start script *SIM*.
TC_ACK_ACCEPT_SUCC_MNEMO	Mnemonics of the TM packets that are used for TC acknowledgements. This variable is set in the start scripts *SIM* and *SCOE*.
TC_ACK_ACCEPT_FAIL_MNEMO	
TC_ACK_EXESTA_SUCC_MNEMO	
TC_ACK_EXESTA_FAIL_MNEMO	
TC_ACK_EXEPRO_SUCC_MNEMO	
TC_ACK_EXEPRO_FAIL_MNEMO	
TC_ACK_EXECUT_SUCC_MNEMO	
TC_ACK_EXECUT_FAIL_MNEMO	
TC_ACK_ACCEPT_SUCC_APID_PARAM	Name of the TM parameter that contains the APID for TC acknowledgements. This variable is set in start scripts *SIM* and *SCOE*.
TC_ACK_ACCEPT_FAIL_APID_PARAM	
TC_ACK_EXESTA_SUCC_APID_PARAM	
TC_ACK_EXESTA_FAIL_APID_PARAM	
TC_ACK_EXEPRO_SUCC_APID_PARAM	
TC_ACK_EXEPRO_FAIL_APID_PARAM	
TC_ACK_EXECUT_SUCC_APID_PARAM	
TC_ACK_EXECUT_FAIL_APID_PARAM	
TC_ACK_ACCEPT_SUCC_SSC_PARAM	Name of the TM parameter that contains the source sequence count for TC acknowledgements. This variable is set in start scripts *SIM* and *SCOE*.
TC_ACK_ACCEPT_FAIL_SSC_PARAM	
TC_ACK_EXESTA_SUCC_SSC_PARAM	
TC_ACK_EXESTA_FAIL_SSC_PARAM	
TC_ACK_EXEPRO_SUCC_SSC_PARAM	
TC_ACK_EXEPRO_FAIL_SSC_PARAM	
TC_ACK_EXECUT_SUCC_SSC_PARAM	
TC_ACK_EXECUT_FAIL_SSC_PARAM	
TM_CYCLIC_MNEMO	Mnemonics of the TM packet that is used for cyclic telemetry. This variable is set in start scripts *SIM* and *SCOE*.
TM_CYCLIC_PERIOD_MS	Period in milliseconds of cyclic telemetry packets. This variable is set in start scripts *SIM* and *SCOE*.

Configuration file	Configuration item	Description
pid.dat	whole files	Information about TM packets and TM parameters are

Configuration file	Configuration item	Description
pic.dat		read from these **MIB tables** and used in *SIM* for the generation of the testdata.txt file.
tpcf.dat		
pcf.dat		
plf.dat		
testdata.sim	whole file	This file is generated by *SIM* and *SCOE* and later on used from *SIM* and *SCOE* when sending TM to SCOS-2000 or CCS.
testdata.txt	whole file	This file is deprecated and not needed anymore by *SIM* and *SCOE*. It has been replaced by the testdata.sim file. Neverless when the testdata.sim file is generated then testdata.txt is generated, too. This file can still be used with the test tool **gpgen** to inject TM packets into SCOS-2000.
TPKTconnTable.dat	TM Packetiser Port	The TM packetiser port is read from the first data line. If there are data lines, then the wrong port might be used. Note: During runtime of *SIM* it is possible to manually overwrite the port number.
TPKTconfigTable.dat	Spacecraft ID	The spacecraft ID is read from the first data line. If there are different spacecraft IDs in the file, then the wrong spacecraft ID might be used.

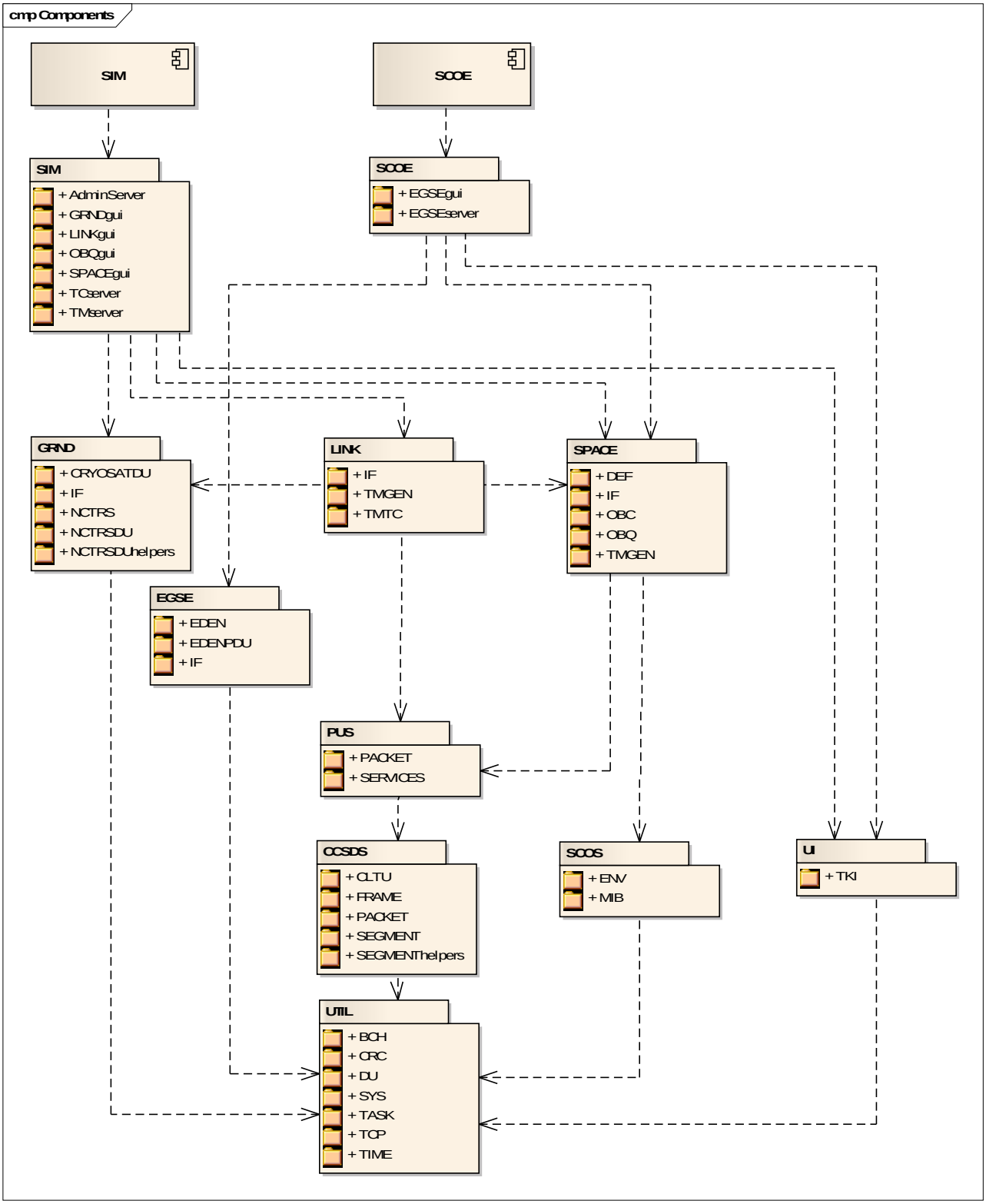
There are also hardcoded configuration items in SIM:

Source file	Configuration item	Description
ENV.py	SCOS_PACKET_HEADER_SIZE	Constants used for the generation of telemetry data.
	TPKT_PKT_IDLE_APID	
	TPKT_PKT_IDLE_SPID	
	TPKT_PKT_IDLE_FRAME_SPID	
	VPD_DATA_SPACE	
MIB.py	Testdata file name $scosii_homedir/testbin/testdata.sim	Path defined relative to scosii_homedir.
	Testdata file name $scosii_homedir/testbin/testdata.txt	
	MIB directory $scosii_homedir/data/ASCII.	
	MIB file names and TM packetiser configuration file names: $scosii_homedir/data/ASCII/pid.dat $scosii_homedir/data/ASCII/pic.dat $scosii_homedir/data/ASCII/tpcf.dat	

Source file	Configuration item	Description
	$scosii_homedir/data/ASCII/pcf.dat $scosii_homedir/data/ASCII/plf.dat $scosii_homedir/data/ASCII/TPKTconnTable.dat $scosii_homedir/data/ASCII/TPKTconfigTable.dat	

3 Library Layering

The *SpacePyLibrary* has the following structure:



The library structure is based on a layering schema:

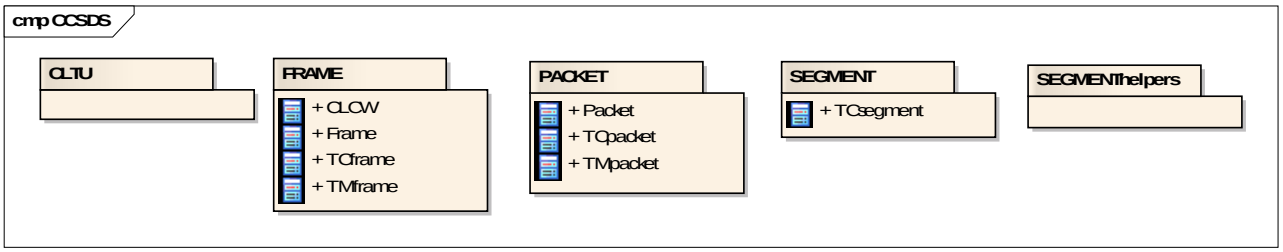
- SIM and SCOE
  - are application components that rely on packages
  - are implemented as python files SIM.py and SCOE.py
- Level 0 packages
  - are the big boxes in the figure
  - are bundling level 1 packages
  - are implemented as folders
  - the arrows in the figure show the dependencies between level 0 packages
- Level 1 packages
  - are the small boxes in the figure - inside the big boxes
  - are bundling python classes, functions and type definitions
  - are implemented as python files

4 Subsystems

This chapter describes the different subsystems in alphabetical order.

4.1 CCSDS

This level 0 package consists of the following level 1 packages:



4.1.1 CLTU

Description: CLTU Handling Module

Classes: none

4.1.2 FRAME

Description: Transfer Frame Module

Classes:

- CLCW: Command link control word
- Frame: telemetry or telecommand transfer frame
- TCframe: telecommand transfer frame
- TMframe: telemetry transfer frame

4.1.3 PACKET

Description: CCSDS Packet Module

Classes:

- Packet: telemetry or telecommand packet
- TCpacket: telecommand packet
- TMpacket: telemetry packet

4.1.4 SEGMENT

Description: Telecommand Segmentation Module

Classes:

- TCsegment: Telecommand segment

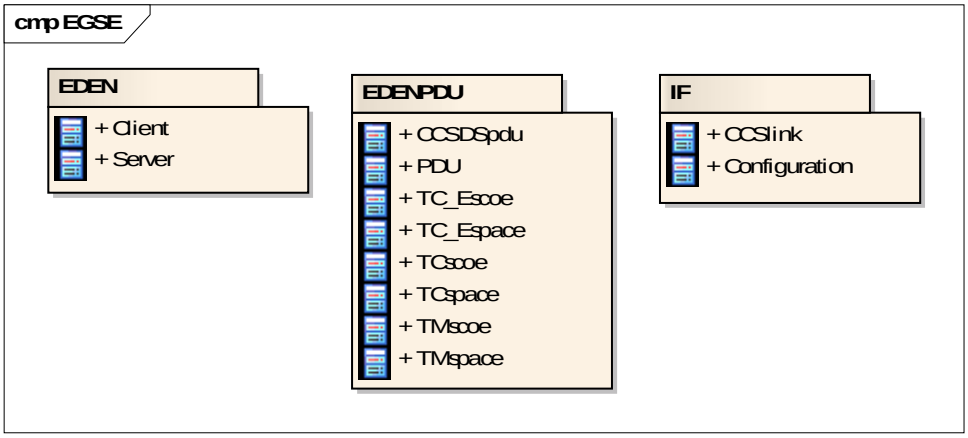
4.1.5 SEGMENThelpers

Description: Telecommand Segmentation Module helpers

Classes: none

4.2 EGSE

This level 0 package consists of the following level 1 packages:



4.2.1 EDEN

Description: EDEN protocol

Classes:

- Client: EDEN PDU interface - CCS side
- Server: EDEN PDU interface - SCOE side

4.2.2 EDENPDU

Description: EDEN protocol Data Units Module

Classes:

- CCSDSpdu: Superclass for different CCSDS PDUs
- PDU: Generic EDEN protocol data unit
- TC_Escoe: (TC_E,SCOE) PDU
- TC_Espace: (TC_E,SPACE) PDU
- TCscoe: (TC,SCOE) PDU
- TCspace: (TC,SPACE) PDU
- TMscoe: (TM,SCOE) PDU
- TMspace: (TM,SPACE) PDU

4.2.3 IF

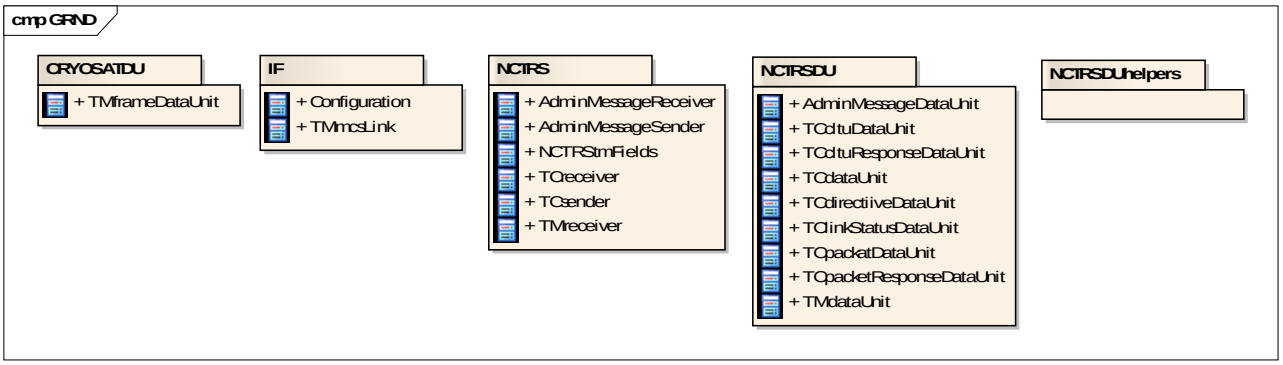
Description: EGSE Interface

Classes:

- CCSlink: Interface to the central checkout system
- Configuration: Configuration

4.3 GRND

This level 0 package consists of the following level 1 packages:



4.3.1 CRYOSATDU

Description: CRYOSAT Data Units Module

Classes:

- TMframeDataUnit: CRYOSAT telemetry frame data unit

4.3.2 IF

Description: Ground Interface

Classes:

- Configuration: Configuration
- TMmcsLink: Telemetry interface to the mission control system

4.3.3 NCTRS

Description: NCTRS Module

Classes:

- AdminMessageReceiver: NCTRS admin message receiver interface - SCOS side
- AdminMessageSender: NCTRS admin message sender interface - NCTRS side
- NCTRStmFields: Helper class that contains static initialization attributes
- TCreceiver: NCTRS telecommand receiver interface - NCTRS side

- TCsender: NCTRS telecommand sender interface - SCOS side
- TMreceiver: NCTRS telemetry receiver interface - SCOS side
- TMsender: NCTRS telemetry sender interface - NCTRS side

4.3.4 NCTRSDU

Description: NCTRS Data Units Module

Classes:

- AdminMessageDataUnit: NCTRS admin message data unit
- TCcltuDataUnit: NCTRS telecommand CLTU data unit
- TCcltuResponseDataUnit: NCTRS telecommand CLTU response data unit
- TCdataUnit: NCTRS telecommand data unit
- TCdirectivesDataUnit: NCTRS telecommand directives data unit
- TClinkStatusDataUnit: NCTRS link status data unit
- TCpacketDataUnit: NCTRS telecommand packet data unit
- TCpacketResponseDataUnit: NCTRS telecommand packet response data unit
- TMdataUnit: NCTRS telemetry data unit

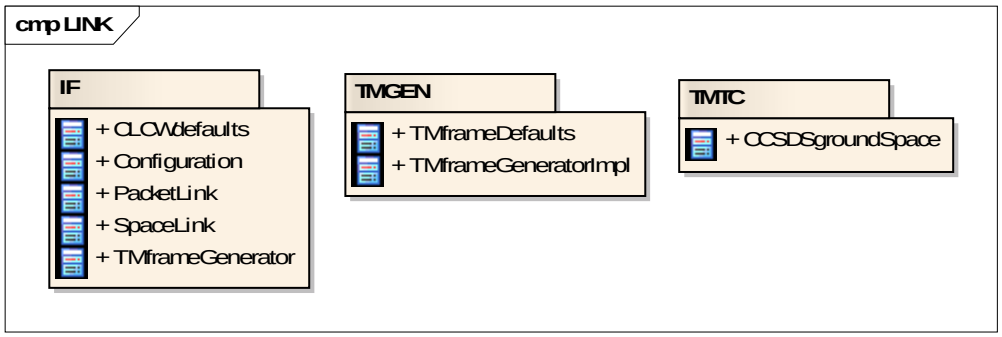
4.3.5 NCTRSDUhelpers

Description: NCTRS Data Units Module helpers

Classes: none

4.4 LINK

This level 0 package consists of the following level 1 packages:



4.4.1 IF

Description: Ground to Space Interface

Classes:

- CLCWdefaults: Default values for CLCW
- Configuration: Configuration

- PacketLink: Interface to the packet link
- SpaceLink: Interface to the spacecraft
- TMframeGenerator: Interface of the generator for telemetry packets

4.4.2 TMGEN

Description: Telemetry Frame Generator

Classes:

- TMframeDefaults: Default values for TM transfer frame creation
- TMframeGeneratorImpl: Generator for telemetry frames

4.4.3 TMTC

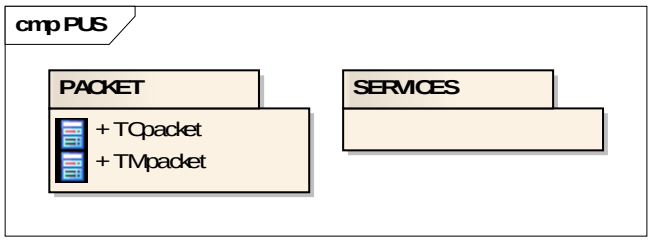
Description: Telemetry and Telecommand Channels

Classes:

- CCSDSgroundSpace: Implementation of the space and packet link, connects the ground segment with the space segment

4.5 PUS

This level 0 package consists of the following level 1 packages:



4.5.1 PACKET

Description: Packet Module

Classes:

- TCpacket: telecommand PUS packet (with datafield header)
- TMpacket: telemetry PUS packet (with datafield header)

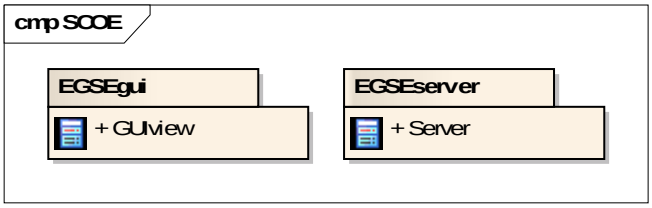
4.5.2 SERVICES

Description: Service Definition

Classes: none

4.6 SCOE

This level 0 package consists of the following level 1 packages:



4.6.1 EGSEgui

Description: EGSE server GUI

Classes:

- GUIview: Implementation of the SCOE EGSE GUI layer

4.6.2 EGSEserver

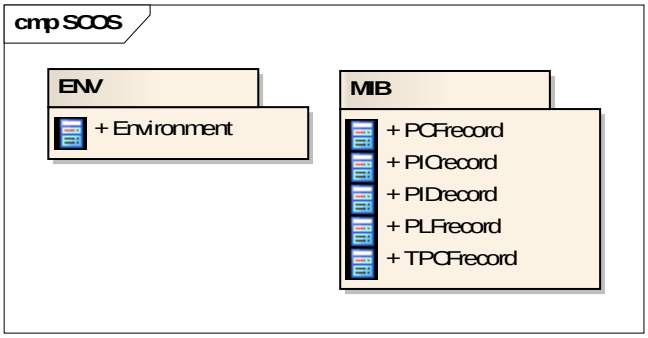
Description: EGSE server for connection to CCS

Classes:

- Server: Server interface class to the CCS

4.7 SCOS

This level 0 package consists of the following level 1 packages:



4.7.1 ENV

Description: Environment

Classes:

- Environment: Manager for environment data

4.7.2 MIB

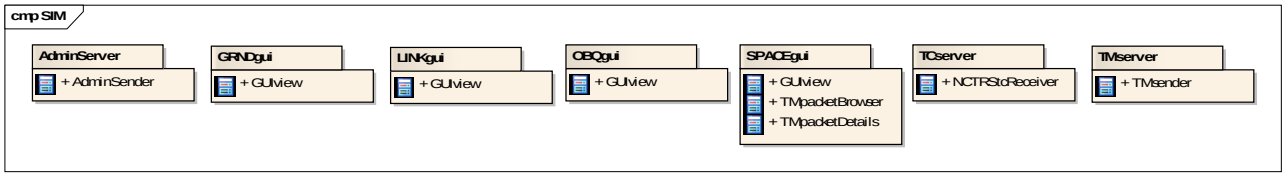
Description: Mission Database (MIB) handling

Classes:

- PCFrecord: MIB record from pcf.dat
- PICrecord: MIB record from pic.dat
- PIDrecord: MIB record from pid.dat
- PLFrecord: MIB record from plf.dat
- TPCFrecord: MIB record from pid.dat

4.8 SIM

This level 0 package consists of the following level 1 packages:



4.8.1 AdminServer

Description: NCTRS admin message server

Classes:

- AdminSender: Subclass of GRND.NCTRS.AdminMessageSender

4.8.2 GRNDgui

Description: Ground Segment Simulation GUI

Classes:

- GUIview: Implementation of the SIM Ground GUI layer

4.8.3 LINKgui

Description: Space Link Simulation GUI

Classes:

- GUIview: Implementation of the SIM Link GUI layer

4.8.4 OBQgui

Description: Onboard Queue Simulation GUI

Classes:

- GUIview: Implementation of the SIM Onboard Queue GUI layer

4.8.5 SPACEgui

Description: Space Segment Simulation GUI

Classes:

- GUIview: Implementation of the SIM Space GUI layer
- TMpacketBrowser: Browser for TM packets
- TMpacketDetails: Displays the packet details, implemented as Tkinter.Frame

4.8.6 TCserver

Description: NCTRS TC server

Classes:

- NCTRStcReceiver: Subclass of GRND.NCTRS.TCreceiver

4.8.7 TMserver

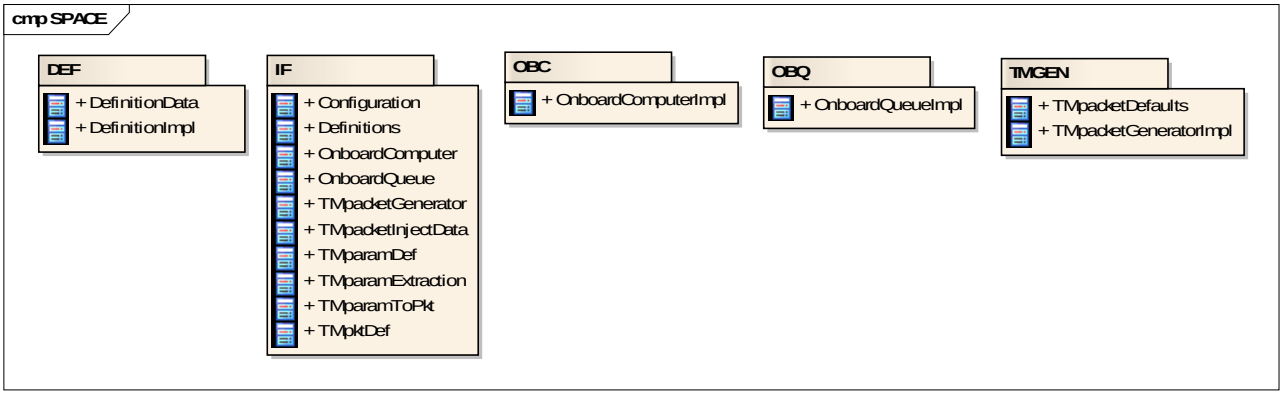
Description: NCTRS TM server

Classes:

- TMsender: Subclass of GRND.NCTRS.TMsender and SPACE.OBC.TMsender

4.9 SPACE

This level 0 package consists of the following level 1 packages:



4.9.1 DEF

Description: Space Data Definitions

Classes:

- DefinitionData: Data part of the Definitions that can be saved and loaded

- DefinitionsImpl: Manager for definition data

4.9.2 IF

Description: Space Interface

Classes:

- Configuration: Configuration
- Definitions: Interface for definition data
- OnboardComputer: Interface of the onboard computer
- OnboardQueue: Interface of the onboard queue
- TMpacketGenerator: Interface of the generator for telemetry packets
- TMpacketInjectData: Data of a TM packet that can be injected
- TMparamDef: Contains the most important definition data of a TM parameter
- TMparamExtraction: Defines a dedicated parameter extraction in a packet
- TMparamToPkt: Contains the data for a single raw value extraction
- TMpktDef: Contains the most important definition data of a TM packet

4.9.3 OBC

Description: Onboard Computer

Classes:

- OnboardComputerImpl: Interface of the onboard computer

4.9.4 OBQ

Description: Onboard Queue

Classes:

- OnboardQueueImpl: Implementation of the onboard computer

4.9.5 TMGEN

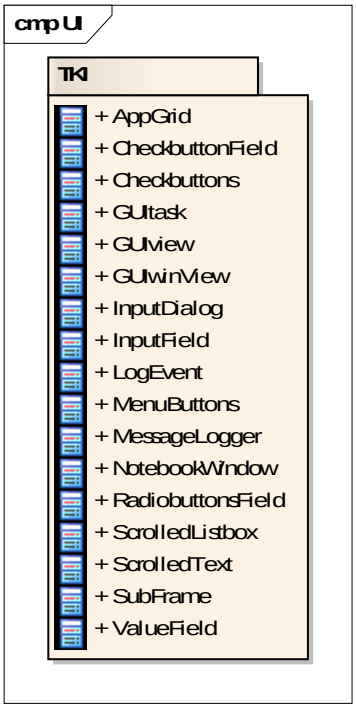
Description: Telemetry Packet Generator

Classes:

- TMpacketDefaults: Default values for TM packet creation
- TMpacketGeneratorImpl: Implementation of the generator for telemetry packets

4.10 UI

This level 0 package consists of the following level 1 packages:



4.10.1 TKI

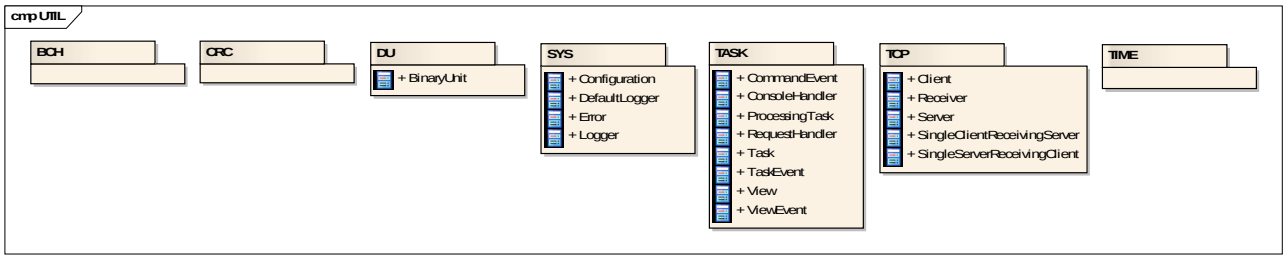
Description: Tkinter support classes

Classes:

- AppGrid: Helper class for grid layout
- CheckbuttonField: Combines a fixed label field and an checkbutton field
- Checkbuttons: Maintains a set of check buttons
- GUItask:Tkinter based task, is the parent task (in the main thread)
- GUIview: Frame with grid layout that consumes status updates
- GUIwinView: GUI window contents
- InputDialog: Input dialog with text field and checkbox entries
- InputField: Combines a fixed label field and an entry field
- LogEvent: event that forces a logging in the gui task
- MenuButtons: Maintains application buttons
- MessageLogger: Scrolled text which implements a GUI based UTIL.SYS.Logger
- NotebookWindow: Application window with a notebook for embedded views.
- RadiobuttonsField: Combines fixed label fields and an radiobutton fields
- ScrolledListbox: Tkinter.Listbox with scroll bars, implemented as Tkinter.Frame
- ScrolledText: Tkinter.Text with scroll bars, implemented as Tkinter.Frame
- SubFrame: Maintains a frame with grid layout
- ValueField: Combines a fixed label field and a dynamic value field managed by a StringVar

4.11 UTIL

This level 0 package consists of the following level 1 packages:



4.11.1 BCH

Description: BCH Encoding

Classes: none

4.11.2 CRC

Description: CRC Checksum Calculation

Classes: none

4.11.3 DU

Description: Data Unit

Classes:

- BinaryUnit: immutable binary data unit

4.11.4 SYS

Description: System Module

Classes:

- Configuration: configuration manager
- DefaultLogger: simple logger that logs via print
- Error: module specific exception to support selective catching
- Logger: interface for logger implementation

4.11.5 TASK

Description: Task Module

Classes:

- CommandEvent: command event that forces an execution in the task
- ConsoleHandler: generic keyboard handler that can be registers in the ModelTask
- ProcessingTask: A task that performs the processing of the application.

- RequestHandler: Handles the requests invoked by the ART framework
- Task: A task is an execution unit attached to a thread.
- TaskEvent: events that are executed from a task
- View: consumer of status updates
- ViewEvent: status event that automatically notifies all views

4.11.6 TCP

Description: TCP/IP Module

Classes:

- Client: TCP/IP client
- Receiver: TCP/IP receiver
- Server: TCP/IP server
- SingleClientReceivingServer: TCP/IP server that receives data from a single client
- SingleServerReceivingClient: TCP/IP client that receives data from a single server

4.11.7 TIME

Description: Time Conversions

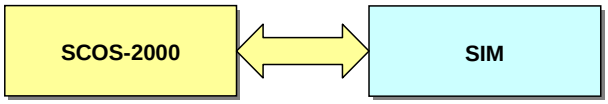
Classes: none

5 SIM

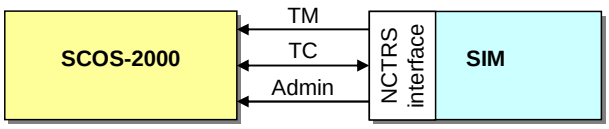
5.1 Overview

SIM is a Ground/Space Simulator implemented with the *SpacePyLibrary*. It has been developed to support system testing of ESA's spacecraft control system SCOS-2000. The major advantages of *SIM* are:

- simplicity
- support for non-nominal tests
- the generic design, which allows testing of telemetry and telecommand components of various SCOS-2000 configurations in a semi-automatic way:



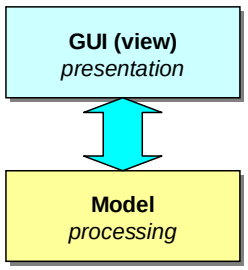
Three TCP/IP connections are used between SCOS-2000 and *SIM*:



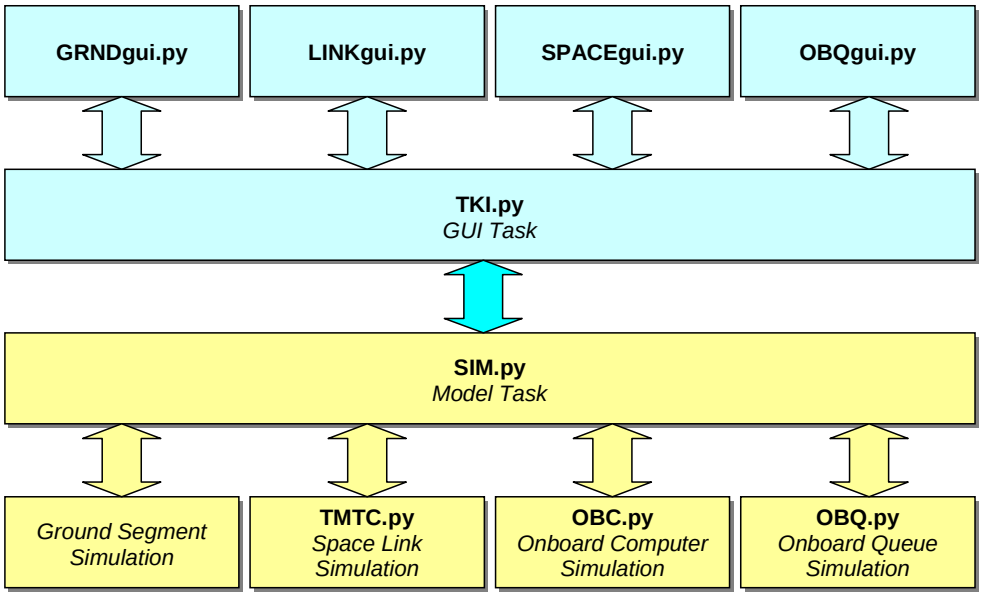
- **TM:** This connection is used to send telemetry frames from *SIM* to SCOS-2000.
- **TC:** This connection is used to send telecommand CLTUs from SCOS-2000 to *SIM* and to send groundstation replies from *SIM* back to SCOS-2000.
- **Admin:** This connection is used to send groundstation administration messages from *SIM* to SCOS-2000.

5.2 Architecture of SIM

SIM has a layered architecture according to the model/view separation pattern:



Depending on the python platform that is used for the execution of *SIM*, GUI and Model are running in a common thread (single-threaded python) or in separate threads (multi-threaded python). The model and view layers consist of the following major components:



The model objects are:

- Ground Segment Simulation: Simulates the ground components, especially the NCTRS interface to SCOS.2000. In addition the ground station TC responses are generated here.
- TMTC.py This component simulates the CCSDS telemetry and telecommand link from the ground station to the spacecraft.
- OBC.py This component simulates the spacecraft's onboard computer: Spacecraft TC acknowledgements are generated here for received telecommands.
- OBQ.py This component simulates the spacecraft's onboard queue. Telecommands are delayed according to the related execution time.

The view objects are implemented with the python GUI packets Tkinter and Ttk:

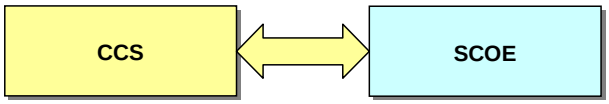
- GRNDgui.py This component visualizes the state of the ground simulation and handles the ground simulation user input commands.
- LINKgui.py This component visualizes the telemetry and telecommand link from the ground station to the spacecraft and handles the link user input commands.
- SPACEgui.py This component visualizes the spacecraft state and handles the user input of spacecraft user input commands.
- OBQgui.py This component visualizes the spacecraft's onboard queue simulation and handles the onboard queue user input commands.

6 SCOE

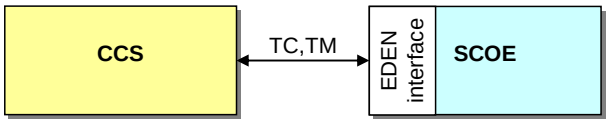
6.1 Overview

SCOE is a Frontend Checkout Equipment Simulator implemented with the *SpacePyLibrary*. It has been developed to support system testing of Central Checkout Systems (CCS) like ESA's EGSE SCOS system or TERMA's Test Sequence Controller. The major advantages of *SCOE* are:

- simplicity
- support for non-nominal tests
- the generic design, which allows testing of telemetry and telecommand components of various CCS configurations in a semi-automatic way:



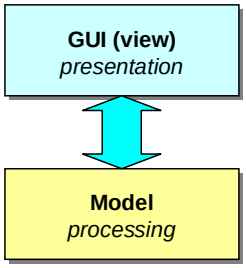
One TCP/IP connection is used between CCS and *SCOE*:



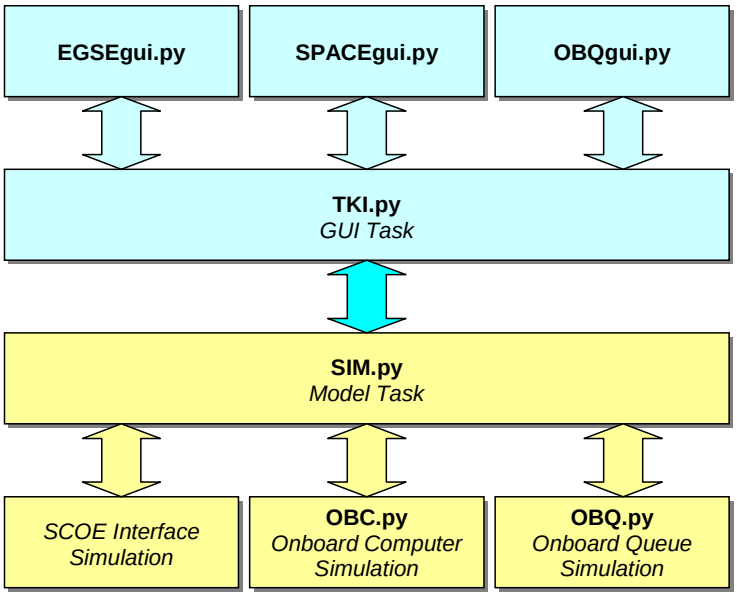
- **TC,TM:** This connection is used to send telecommand packets from the CCS to *SCOE* and to send telemetry packets from *SCOE* to the CCS.

6.2 Architecture of SCOE

SCOE has a layered architecture according to the model/view separation pattern:



Depending on the python platform that is used for the execution of *SCOE*, GUI and Model are running in a common thread (single-threaded python) or in separate threads (multi-threaded python). The model and view layers consist of the following major components:



The model objects are:

- SCOE Interface Simulation: simulates the SCOE EDEN interface for TC and TM.
- OBC.py This object simulates the spacecraft's onboard computer: Spacecraft TC acknowledgements are generated here for received telecommands.
- OBQ.py This object simulates the spacecraft's onboard queue. Telecommands are delayed according to the related execution time.

The view objects are implemented with the python GUI packets Tkinter and Ttk:

- EGSEgui.py This object visualizes the state of the SCOE EDEN interface and handles the related user input commands.
- SPACEgui.py This object visualizes the spacecraft state and handles the user input of spacecraft user input commands.
- OBQgui.py This object visualizes the spacecraft's onboard queue simulation and handles the onboard queue user input commands.

<<end of document>>