Class: N/A
Doc. no: UNIS-TER-MAN-001
Rev: 1.1
Date: 2017-09-25

TERMA
Space

uNIS Software User Manual

Checked: GASatt' Digitally signed by Giovanni Scotti DN: cn=Giovanni Scotti, c=DE, o=Terma, ou=Space, email=gis@terma.com Date: 2017.09.28 11:04:09 +02'00'

Giovanni Scotti Date

Software Engineer

Checked: L Gaemers Zomerdijk 2017.09.27

L. Gaemers Zomerdijk Date

Configuration Control

Checked:

Dan Søren Nielsen Date

PA Manager

Authorized: Andy Armitage Digitally signed by Andy Armitage ~~Date: 2017.09.30 13:15:48 +02~~'00'

A. Armitage Date

Director, Systems & Technology Development

This document is released for use only if signed by relevant staff or stamped "EDM Release Controlled" CM:

Record of Changes

ECO	Description	Rev	Date
	First version	1.0	2017-06-14
	Updates: RID-08 sections 5, 6 and 6.1 RID-10 section 6.1.1 RID-11 section 10 RID-12 section 5 RID-14 section 5 RID-15 section 4	1.1	2017-09-25

Contents

Figures

Tables

1 Introduction

1.1 Purpose

This document is the software user manual for the uNIS application.

It provides an high level description of the application and its external interfaces, startup options, Settings and configuration information.

1.2 Scope

The intended readership is: end users, engineers developers, testers, administrators of the CCS/TSC.

2 Documents

2.1 Applicable Documents

QuickRef	Document	Ref#
CCSDS-SLE-RAF	Space Link Extension - Return All Frames Service Specification	CCSDS 911.1-B-2
CCSDS-SLE-FCLTU	Space Link Extension - Forward CLTU Service Specification	CCSDS 912.1-B-2
CCSDS-SLE-RCF	Space Link Extension - Return Channel Frames Service Specification	CCSDS 911.2-B-1
NIS-MCS-ICD	Interface Control Document NIS/NCTRS Volume 2- Detailed interface Definition: MCS	EGOS-NIS-NCTR-ICD-0002 v. 4.0.2, date 06.11.2009
SLEAPI-RM	ESA SLE API Package Reference Manual	SL-ANS-RF-0001, Issue 4.8, 30.10.2009
SICF-ICD	Service Instance Configuration File ICD - Space Link Extension Services.	EGOS-MDW-SLES-ICD-0001-i1r0

2.2 Reference Documents

None.

3 Terms, Definitions and Abbreviated Terms

Term	Definition
CCS5	Control and Check out System version 5
GSFE	Ground Station Front End
SI	SLE Service Instance
SLE	Space Link Extension
SCID	Spacecraft Identifier
VCID	Virtual Channel Identifier
TFVN	Telemetry Frame Version Number

4 The uNIS

uNIS represents a gateway between a Mission or Checkout Control System (e.g. CCS5, TSC) and a ground station for telemetry returning and telecommand forwarding.

The uNIS interfaces the ground station equipment according to the CCSDS Space Link Extension standard, implementing in particular RAF, RCF and FCLTU SLE services (see CCSDS-SLE-RAF, CCSDS-SLE-RCF and CCSDS-SLE-FCLTU).

SLE Service Instances are operated by means of the ESA SLE API package, see [SLEAPI-RM].

The following picture shows the uNIS and its interfaces to the CCS (GSFE in particular) and the SLE Provider located at the ground station. Notice that uNIS interfaces CCS also for delivering log messages (info, warnings, errors); this interfaces is not shown in the figure.

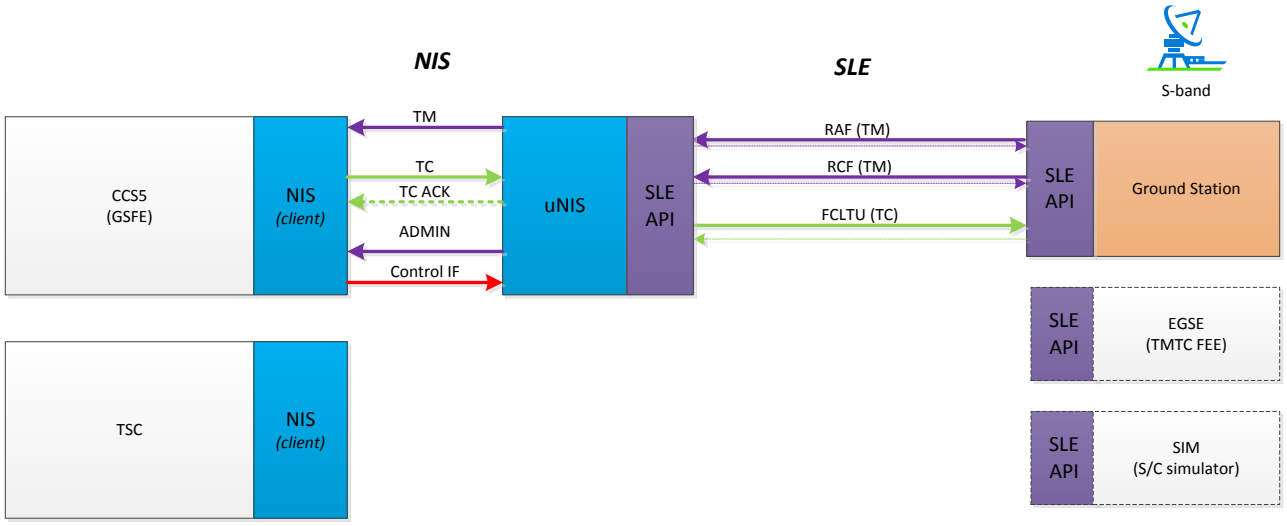

Figure 4-1 uNIS and its context

The uNIS provides the CCS/TSC client with four TCP/IP ports (the “CCS Interface”) for connecting, respectively for TM returning, CLTU forwarding/responses returning, returning of Monitoring messages (also known as Administrative message) and for Monitor & Control (Tcl Control Interface).

Multiple uNIS processes can be running at the same time, provided that they are configured not to cause clashes accessing SLE resources (they must operate different Service Instances) and local TCP/IP sockets (they can’t provide CCS TM/TC/MON/ Tcl Control services on the same ports). Each uNIS instance manages SLE service instances for a given ground station (identified by a specific identifier, in the following referred to as <<*GSID*>>).

5 Start and Stop the uNIS

A single uNIS instance handles one ground station (i.e. one SLE Responder Identifier, as defined in the SICF: *ResponderId*) and it is uniquely identified by its Ground Station Identifier: a string, in the following referred to as <<GSID>> (e.g. TestUnis1, KIRUNA).

Before starting uNIS for the specific G/S for the first time, this G/S has to be configured in the CcsServer settings as described in the next section.

uNIS for one ground station can be started from command line:

```
uNIS <<GSID>
```

(on windows the command reads: `uNIS.exe` instead of `uNIS`)

Alternatively, the uNIS can be started or stopped from TOPE procedures as follows:

```
UNIS::start <<GSID>
```

```
UNIS::stop <<GSID>
```

However, notice that uNIS can be started/stopped by the CCS5 operator directly via the GsfeView, once *NisDriver* driver(s) are defined in the CCS5 settings (MCS\GSFE\Drivers section). If the NIS driver is configured with the settings *useUNIS* = 1, then the GSFE manages the start/stop of UNIS, when :

- Driver Activate Button (GSFE View)
- autoActivate setting (GSFE API)

are changed. This is important to ensure that the UNIS is started on the CCS server, and within the driver settings useUNIS is 1. If UNIS will be used operationally, it must be started on the CCS server not on the client. For these cases a direct UNIS::start on a CCS client is discouraged.

With reference to the following picture, the *Active* checkbox starts (box checked) and stops (unchecked) the corresponding uNIS instance. The *TM*, *TC* and *MON* boxes are used to activate/deactivate the CCS5 (GSFE) connection to one uNIS, respectively for: Telemetry, Telecommand, and Administrative Messages.

Figure 5-1 GsfeView showing two uNIS drivers (two ground stations: TestUnis1, TestUnis2)

The corresponding GSFE settings are shown in : two NisDriver-type drivers are defined to enable connection with two uNIS instances (example with two ground stations: TestUnis1, TestUnis2).

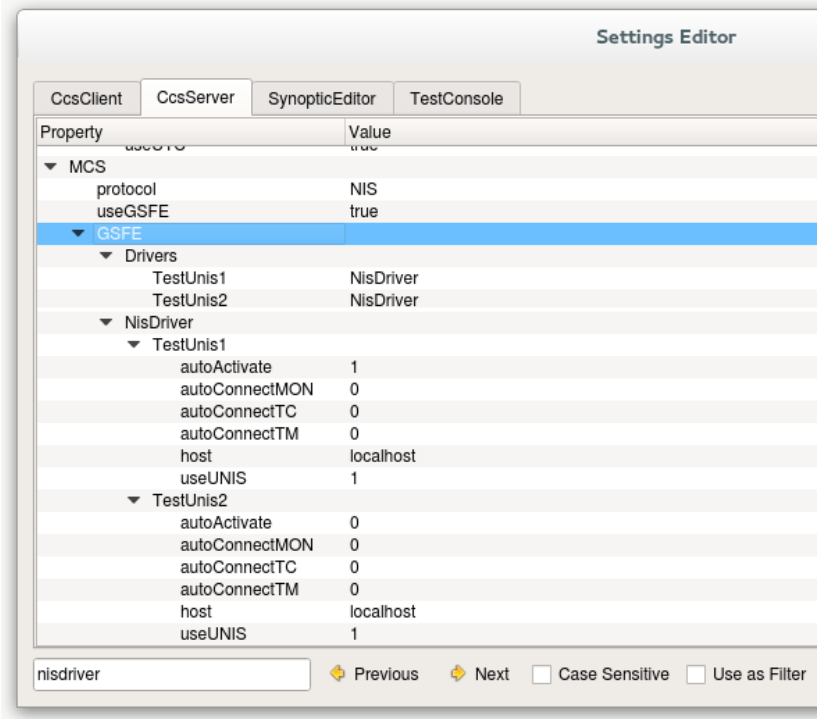

Figure 5-2 MCS\GSFE NisDriver settings

below shows the same settings defined programmatically, e.g. as part of *InitSettings.tcl.* The Tcl code below also shows the associated minimal set of uNIS settings for the two ground stations (see section 7 below for details, other uNIS settings are set to default values by uNIS at startup and might require customization, as explained in section 7).

```
# if an operations interface is used, create these two drivers
if [string is true -strict $::utope::settings(CcsServer:MCS/useGSFE)] {
    foreach app {CcsClient CcsServer} {
        set ::utope::settings(${app}:MCS/GSFE/Drivers/TestUnis1) NisDriver
        set ::utope::settings(${app}:MCS/GSFE/Drivers/TestUnis2) NisDriver
    }
    # tells the driver that it will use UNIS
    # (and tells it to look in the UNIS settings for port numbers)
    set ::utope::settings(CcsServer:MCS/GSFE/NisDriver/TestUnis1/useUNIS) 1
    set ::utope::settings(CcsServer:MCS/GSFE/NisDriver/TestUnis2/useUNIS) 1
}

# now set up UNIS settings..., the NIS driver, being told useUNIS will use these settings
# the admin port is a vestige of the ESA-NIS, so with UNIS would typically not be used
# TestUnis1 instance uses the default values
set ::utope::settings(CcsServer:uNIS/TestUnis1/useAdminPort) 0
set ::utope::settings(CcsServer:uNIS/TestUnis1/tclPort) 50019
# TestUnis2 uses different values
set ::utope::settings(CcsServer:uNIS/TestUnis2/tcPort) 20011
set ::utope::settings(CcsServer:uNIS/TestUnis2/tmPort) 20014
set ::utope::settings(CcsServer:uNIS/TestUnis2/tclPort) 50029
```

Figure 5-3 MCS\GSFE and uNIS settings defined programmatically (example)

Notice that, as for any other GSFE driver, a uNIS driver is automatically deactivated following a Session Stop and therefore uNIS process terminated. uNIS is automatically reactivated at Session Start (uNIS process restarted), if it was active before the stop.

uNIS driver state is not changed automatically upon Session Switch, therefore uNIS keeps running during the switch. The same happens when stopping (and restarting) the CCS Console.

If an instance of uNIS is started, either manually from command line or via TOPE, and there is another instance running for the same ground station, the latter will refuse to start, without affecting the status of the first one.

Notice that the CCS *abort* script terminates all CCS processes, including running uNIS's, if any.

6 The CCS interface

This section describes the interface provided by uNIS to CCS, composed of:

- *Tcl Control interface* for controlling the uNIS and its SLE Service Instances;
- *TM interface* for delivering TM frames;
- *TC interface* for reception of CLTU data and delivery of associated responses;
- *MON interface* for delivering Administrative Messages.

6.1 Tcl Control interface

SLE services running on one uNIS instance can be explicitly controlled from CCS5 (stopped, started, bound, unbound, aborted), and queried for their status and statistics via a remote Tcl channel over TCP/IP. In effect a Tcl interpreter is connected to a TCP/IP server; the client connects to this server and interacts with it in the same way it would with a Tcl console. It requests the values of variables and calls statements using Tcl syntax.

The Tcl server port number is specified as part of the uNIS settings (see *tclPort* settings below).

Inside the server, the interpreter is extended to link Tcl variables with service instance status, and to expose functions as Tcl statements. It supports normal Tcl introspection, which helps to minimize hard-coding of the interface. This gives a reliable, flexible and consistent way for TOPE procedures running on the CCS to get full remote visibility & control of the Service Instances running in the UNIS.

For a given uNIS instance (i.e. ground station), the corresponding Tcl control interface can be accessed from TOPE procedures.

First of all it is required that a connection to the specific uNIS instance is established:

```
::UNIS::connect <<GSID>
```

Where <<GSID> is the name of the GSFE NisDriver for the specific ground station, e.g. Kiruna, TestUnis1, TestUnis2, etc. Examples:

```
::UNIS::connect Kiruna
```

```
::UNIS::connect TestUnis1
```

Upon successful connection, commands and variables from the `::UNIS::<<GSID>` name space become available as described below.

Notice that each uNIS instance is able to accept multiple connections at the same time, allowing multiple Tcl clients (Tcl scripts, including the Manual Commands environment within a CcsConsole) to operate the uNIS simultaneously. Each Tcl client has to invoke ::UNIS::connect <<GSID> in order to be able to operate a uNIS instance within its scope.

Examples of uNIS Tcl commands and use of uNIS variables can be found in the uNIS Auto test procedure : TestPacks/CcsSelfTest/TSEQ/UnisAutoTest.tcl.

6.1.1 Commands

The list of available commands can be retrieved using the standard Tcl construct ‘info commands’:

```
>info commands ::UNIS::TestUnis1::*

::UNIS::TestUnis1::unbind ::UNIS::TestUnis1::bind
::UNIS::TestUnis1::stop ::UNIS::TestUnis1::start
::UNIS::TestUnis1::abort ::UNIS::TestUnis1::checkconn
```

At any time the ::UNIS:: <<GSID>::*checkconn* command is available to verify the connection with the uNIS instance. An exception is thrown if the connection is no longer active.

The following commands are used to operate the SLE service instances:

Table 6-1 UNIS SLE Control Operations

Command and arguments In the namespace `::UNIS::<<GSID>::`	**Description**	**Possible errors** *message:* description
bind *siindex*	Bind the SLE service instance identified by the *siindex* string. The *siindex* string is a simple identifier for a service instance in the uNIS instance. It has the form *sssN*, where *sss* is: “raf”, “rcf” or “cltu” (depending on the SLE service type), N is an integer, starting from 1 for each uNIS instance and SLE service type: 1, 2, 3,.. examples: `::UNIS::TestUnis1::bind raf1` `::UNIS::TestUnis1::bind rcf1` `::UNIS::TestUnis1::bind cltu1` `::UNIS::TestUnis1::bind raf2`	*Cannot bind [siid] (state is [state])*: Service instance with identifier [siid] is not currently in the applicable state for binding (unbound). *Negative Bind Return: [diagnostic]:* the operation has been rejected by the SLE provider with diagnostic [diagnostic]. Where diagnostic is according to CCSDS-SLE-RAF, CCSDS-SLE-FCLTU, CCSDS-SLE-RCF, depending on service type). *Protocol abort.:* The Bind operation failed due to SLE provider host not reachable at the IP address and port specified in the SLE Proxy configuration.
unbind *siindex*	Unbind the SLE service instance identified by the *siindex* string UNIS automatically sets the SLE UNBIND operation parameter *unbind-reason* to the value of ‘suspend’.	*Cannot unbind [siid] (state is [state]):* Service instance with identifier [siid] is not currently in the applicable state for unbinding (bound). *Negative Unbind Return: [diagnostic]:* the operation has been rejected by the SLE provider with diagnostic [diagnostic]. Where diagnostic is according to CCSDS-SLE-RAF, CCSDS-SLE-FCLTU, CCSDS-SLE-RCF, depending on service type).
start *siindex ?starttime? ?stoptime? ?TFVN? ?SCID? ?VCID?*	Start the SLE service instance identified by the *siindex* string. *starttime* and *stoptime* specify the start and stop times for the SLE START operation. Their possible values are: ‘void’ or a CCSDS	*Cannot start [siid] (state is [state]):* Service instance with identifier [siid] is not currently in the applicable state for starting (bound)

Command and arguments In the namespace `::UNIS::<<GSID>::`	**Description**	**Possible errors** *message:* description
	date time string (YYYY-MM-DDThh:mm:ss.mmm) *starttime* and *stoptime* are: • optional for the RAF service (null times are assumed if they are not specified, see CCSDS-SLE-RAF for the meaning of such settings) • mandatory for RCF service • not required for CLTU services *TFVN, SCID, VCID* are required only for the RCF service to specify the *Global VCID* i.e. the combination of: transfer frame version number defined by CCSDS (values: 0 or 1), spacecraft identifier, virtual channel identifier (values: 0, 1, …, 7) or the value '*'. The '*' (i.e. 'any') indicates that a master channel, defined by the TFVN and the SCID, shall be provided by the RCF service examples: **RAF**: `::UNIS::TestUnis1::start raf1` `::UNIS::TestUnis1::start raf1 2017-05-25T14:06:06.000 2017-05-25T13:57:06.000` **RCF**: `::UNIS::TestUnis1::start rcf1 void 2017-05-25T14:07:15.000 0 530 0` `::UNIS::TestUnis1::start rcf1 2017-05-25T14:06:06.000 2017-05-25T13:57:06.000 0 530 0` `::UNIS::TestUnis1::start rcf1 void void 0 530 *` **CLTU**: `::UNIS::TestUnis1::start cltu1` Notice that, depending on the SLE service, the UNIS automatically sets some of the SLE START operation parameters, namely: **RAF**: *requested-frame-quality*: set according to uNIS settings (RequestedFrameQuality parameters, see section 7) **CLTU**: *first-cltu-identification*: set to 0	*Cannot start. Already active.* A Service instance has been required to start, but this is already in active state. *Cannot start. A [type] SI is already active: [siid].* A Return type Service instance (RAF, RCF) has been required to start, but there is another return SI (type [type], identifier [siid]) currently in active state. At any given time, only one return SI can be active. *This SI is type [type] (not RCF). Global-VCID not required:* Parameter TFVN, SCID or VCID has been specified for a RAF or CLTU service instance, that does not require such parameters at start. *RCF SI: Start operation requires parameter Global-VCID*: a RCF-START operation has been requested but the necessary TFVN, SCID, VCID triplet has not been provided. *RCF SI: Start operation rejected allowed values: TFVN: 0 or 1, ; SCID > 0; VCID: *, 0, 1..7*: a RCF-START operation has been requested but the provided Global-VCID (TFVN, SCID, VCID) is not correct. *CLTU SI: Start operation shall not have any start/stop times*: a CLTU-START operation has been requested, that includes start/stop time arguments that, in this case, are not applicable. *Negative Start Return: [diagnostic]:* the operation has been rejected by the SLE provider with diagnostic [diagnostic]. Where diagnostic is according to CCSDS-SLE-RAF, CCSDS-SLE-FCLTU, CCSDS-SLE-RCF, depending on

Command and arguments In the namespace `::UNIS::<<GSID>::`	**Description**	**Possible errors** *message:* description
		service type).
stop *siindex*	Stop the SLE service instance identified by the *siindex* string	*Cannot stop [siid] (state is [state]):* Service instance with identifier [siid] is not currently in the applicable state for stop (active). *Negative Stop Return: [diagnostic]:* the operation has been rejected by the SLE provider with diagnostic [diagnostic]. Where diagnostic is according to CCSDS-SLE-RAF, CCSDS-SLE-FCLTU, CCSDS-SLE-RCF, depending on service type).
abort *siindex*	Abort the SLE service instance identified by the *siindex* string	*Cannot abort [siid] (state is unbound):* Service instance with identifier [siid] is already in unbound state.
-	-	**Common to all operations above**: *Service Instance short index [siindex] does not exist:* the argument siindex does not correspond to an existing Service Instance within the given uNIS instance. *Request timeout* : the requested operation has timed out without receiving any confirmation from the SLE Provider, either negative or positive (note: this messages does not apply to Abort, which is an unconfirmed operation).

6.1.2 Variables

The ::UNIS:: <<GSID>:: name space includes a number of variables:

- `::UNIS:: <<GSID>::Status` : a global Status array (read/write variable to get/set the application verbose mode)
- `::UNIS::TestUnis1::rcf<<N>>, ::UNIS::TestUnis1::raf<<N>>, ::UNIS::TestUnis1::cltu<<N>>`
  For each SLE service instance (SI) loaded in a given uNIS process, a service variable is defined, i.e. a tcl array providing information on the Service instance. Depending on the type (RAF, RCF, CLTU), the variable name is: raf<<N>>, rcf<<N>>, cltu<<N>>, where <<N>> is an integer number identifying the SLE SI (<<N>> = 1, 2, etc.)

```
>info vars ::UNIS::TestUnis1::*

::UNIS::TestUnis1::rcf1 ::UNIS::TestUnis1::cltu1
::UNIS::TestUnis1::raf1 ::UNIS::TestUnis1::raf2
::UNIS::TestUnis1::raf3 ::UNIS::TestUnis1::Status
```

uNIS Status Array

The `::UNIS::` `<<GSID>::Status` array includes read/write elements:

```
>array names ::UNIS::TestUnis1::Status

verbose sletracing
```

Where:

- `verbose` : is the equivalent of the *verbose* uNIS configuration setting described in . Setting the value of this element to 1 (0) enables (disables) uNIS verbose mode dynamically at runtime. In verbose mode the application generates additional debug messages, besides the standard Errors, Warnings and Information messages.
- `sletracing` : is the equivalent of the *SLEAPI/Tracing* setting described in . Setting the value of this element to 1 (0) enables (disables) SLE API full tracing mode dynamically at runtime. Warning: enabling this option can significantly reduce application performances.

So:

```
set ::UNIS::TestUnis1::Status(verbose) 1

set ::UNIS::TestUnis1::Status(sletracing) 1
```

Enables both Verbose mode and SLE API tracing on uNIS instance “TestUnis1”. The modes are disabled with:

```
set ::UNIS::TestUnis1::Status(verbose) 0

set ::UNIS::TestUnis1::Status(sletracing) 0
```

CLTU service array

Array names and values are retrieved for instance with the standard Tcl construct ‘array get’:

```
>array get ::UNIS::TestUnis1::cltu1

n_cltus_rec 5 n_cltus_proc 4 buffer_size 100000 cltu_last_pr 0
cltu_last_ok 0 sletype FCLTU responderid sle_provider siid
sagr=SAGR.spack=SPACK.fsl-fg=FSL-FG.cltu=cltu1 productionstatus
operational cltu_status radiated status unbound n_cltus_rad 3
uplink_status nominal
```

Where:

Table 6-2 CLTU service array

Array Item name	Description
`sletype`	SLE Service type: ‘`FCLTU`’
`siid`	SLE SI identifier as defined in the SICF, e.g. sagr=SAGR.spack=SPACK.fsl-fg=FSL-FG.cltu=cltu1

responderid	SLE Responder Identifier as defined in the SICF, e.g. 'sle_provider' or 'kiruna'
status	SLE Service Instance status as defined in the SLE CCDS standard (see CCSDS-SLE-FCLTU)
productionstatus	SLE Production status as reported by CLTU-STATUS-REPORT , parameter production-status (see CCSDS-SLE-FCLTU)
cltu_status	Value as reported by the CLTU-STATUS-REPORT, parameter: cltu-status (see CCSDS-SLE-FCLTU)
uplink_status	Value as reported by the CLTU-STATUS-REPORT, parameter: uplink-status (see CCSDS-SLE-FCLTU)
n_cltus_rad	Value as reported by the CLTU-STATUS-REPORT, parameter: number-of-cltus-radiated (see CCSDS-SLE-FCLTU)
n_cltus_rec	Value as reported by the CLTU-STATUS-REPORT, parameter: number-of-cltus-received (see CCSDS-SLE-FCLTU)
n_cltus_proc	Value as reported by the CLTU-STATUS-REPORT, parameter: number-of-cltus-processed (see CCSDS-SLE-FCLTU)
cltu_last_pr	Value as reported by the CLTU-STATUS-REPORT, parameter: cltu-last-processed (see CCSDS-SLE-FCLTU)
cltu_last_ok	Value as reported by the CLTU-STATUS-REPORT, parameter: cltu-last-OK (see CCSDS-SLE-FCLTU)
buffer_size	Value as reported by the CLTU-STATUS-REPORT, parameter: cltu-buffer-available (see CCSDS-SLE-FCLTU)

RAF service array

```
>array get ::UNIS::TestUnis1::raf1
```

```
n_error_free_frames_delivered 6 symbolic_sync_lock_status in-lock
carrier_lock_status in-lock subcarrier_lock_status in-lock sletype
RAF n_frames_delivered 6 responderid sle_provider siid
sagr=SAGR.spack=SPACK.rsl-fg=RSL-FG.raf=onlc1 frame_sync_lock_status
in-lock productionstatus running status unbound
```

Where:

Table 6-3 RAF service array

Array Item name	Description
sletype	SLE Service type: 'RAF'
siid	SLE SI identifier as defined in the SICF, e.g. sagr=SAGR.spack=SPACK.rsl-fg=RSL-FG.raf=onlc1
responderid	SLE Responder Identifier as defined in the SICF, e.g. 'sle_provider'
status	SLE Service Instance status as defined in the SLE CCDS standard (see CCSDS-SLE-RAF)

productionstatus	SLE Production status as reported by RAF-STATUS-REPORT, parameter production-status (see CCSDS-SLE-RAF)
n_frames_delivered	Value as reported by the RAF-STATUS-REPORT, parameter: number-of-frames-delivered (see CCSDS-SLE-RAF)
n_error_free_frames_delivered	Value as reported by the RAF-STATUS-REPORT, parameter: number-of-error-free-frames-delivered (see CCSDS-SLE-RAF)
symbolic_sync_lock_status	Value as reported by the RAF-STATUS-REPORT, parameter: symbol-sync-lock-status (see CCSDS-SLE-RAF)
carrier_lock_status	Value as reported by the RAF-STATUS-REPORT, parameter: carrier-lock-status (see CCSDS-SLE-RAF)
subcarrier_lock_status	Value as reported by the RAF-STATUS-REPORT, parameter: subcarrier-lock-status (see CCSDS-SLE-RAF)
frame_sync_lock_status	Value as reported by the RAF-STATUS-REPORT, parameter: frame-sync-lock-status (see CCSDS-SLE-RAF)

RCF service array

```
>array get ::UNIS::TestUnis1::rcf1
```

```
n_error_free_frames_delivered na symbolic_sync_lock_status in-lock
carrier_lock_status in-lock subcarrier_lock_status in-lock sletype
RCF n_frames_delivered 6 responderid sle_provider siid
sagr=SAGR.spack=SPACK.rsl-fg=RSF-FG.rcf=onlc1 frame_sync_lock_status
in-lock productionstatus running status unbound
```

Where:

Table 6-4 RCF service array

Array Item name	Description
sletype	SLE Service type: 'RCF'
siid	SLE SI identifier as defined in the SICF, e.g. sagr=SAGR.spack=SPACK.rsl-fg=RSF-FG.rcf=onlc1
responderid	SLE Responder Identifier as defined in the SICF, e.g. 'sle_provider'
status	SLE Service Instance status as defined in the SLE CCDS standard (see CCSDS-SLE-RCF)
productionstatus	SLE Production status as reported by RCF-STATUS-REPORT, parameter production-status (see CCSDS-SLE-RCF)

n_frames_delivered	Value as reported by the RCF-STATUS-REPORT, parameter: number-of-frames-delivered (see CCSDS-SLE-RCF)
symbolic_sync_lock_status	Value as reported by the RCF-STATUS-REPORT, parameter: symbol-sync-lock-status (see CCSDS-SLE-RCF)
carrier_lock_status	Value as reported by the RCF-STATUS-REPORT, parameter: carrier-lock-status (see CCSDS-SLE-RCF)
subcarrier_lock_status	Value as reported by the RCF-STATUS-REPORT, parameter: subcarrier-lock-status (see CCSDS-SLE-RCF)
frame_sync_lock_status	Value as reported by the RCF-STATUS-REPORT, parameter: frame-sync-lock-status (see CCSDS-SLE-RCF)

6.2 TM interface

For a given uNIS instance, Telemetry Data Units (uNIS TMDU) generated by any active Return SLE Service instance (RAF or RCF) are transmitted from the uNIS to the CCS over a TCP/IP connection. The uNIS acts as a server. The CCS initiates the connection. No authentication of clients is performed by the uNIS.

The TM port number is specified as part of the uNIS settings (see *tmPort* settings below).

There is no limit to the number of clients that can connect to the TM port.

If no client is connected on a port and data is received from a GS, data are discarded.

When the TCP/IP link is disconnected or lost, the GS link is not affected.

The Table below gives the TM DU layout[1]. The user data is the TM Transfer Frame (TF). A specific header is added. Notice that Integers mentioned in the table are unsigned integers.

Table 6-5 TM Data Unit

octets	Field name	Size (octets)	Comment
0-3	**packet size**	4	integer; total length of the DU (in bytes)
4-5	**S/C ID**	2	integer; satellite code or S/C identifier
6	**data stream type**	1	integer; identifies the channel and the type of data: SLC (RAF): Onl-TIMELY(0) Onl-COMPLETE (4) MC (RCF): Onl-TIMELY (1) Onl-COMPLETE(5) VC (RCF): Onl-TIMELY (2) Onl-COMPLETE (6) Bad fr (RAF): Onl-TIMELY (3) Onl-COMPLETE (7)
7	**VC ID**	1	integer; {[0;63]}
8-9	**route ID**	2	integer; GS id: site code
10-17	**ERT**	8	CDS time format (CCSDS Day Segmented); earth reception time

[1] The format of the uNIS TMDU follows NIS-MCS-ICD, version 0 of the TM format is applied (see section 2.2.2. of NIS-MCS-ICD).

18	**sequence flag**	1	Integer; sequence annotation {in sequence (0); out of sequence (1)}
19	**quality flag**	1	integer; data quality {good (0); bad/incorrect GS Id/incorrect data stream Id (1)}
20-(size-1)	**user data**		octet string; TM Transfer frame NB: RS symbols are included if delivered by the GS equipment.

6.3 TC interface (CLTU)

uNIS Telecommand Data Units (uNIS TCDU[2]) are transmitted over a full-duplex TCP/IP connection. The uNIS acts as a server. One TC port is defined per uNIS process and only one connection is accepted on this port.

The TC port number is specified as part of the uNIS settings (see *tcPort* settings below).

The TCDU consists successively of:

- a header (see Table 6-6)
- a variable part, which can be made of one of the following:
  - TC CLTU sent by CCS5 to uNIS (see Table 6-7);
  - TC CLTU responses sent back by uNIS (see Table 6-8).

[2] The format of the uNIS TCDU follows NIS-MCS-ICD, TC CLTU definition (see section 3.3. of NIS-MCS-ICD) with the exception of the value set by uNIS in field ‘reason’ in the TC CLTU response.

Table 6-6 TC DU header

octets	Field name	Size (octets)	Comment
0-3	**packet size**	4	integer; total length of the DU (in bytes)
4-5	**type**	2	integer; type of data unit: { 7 = CLTU 8 = CLTU response }
6-7	**S/C id**	2	integer; S/C identifier

Table 6-7 TC CLTU

octets	Field name	Size (octets)	Comment
0	**spare**	1	not used
1-4	**TC Id**	4	integer; CLTU Id incremented by one
5-10	**spare**	6	not used
11-14	**earliest production time flag**	4	integer; {time not used(0); time used(1)}
15-22	**earliest production time**	8	CDS time (CCSDS Day Segmented) used in the SLE CLTU IF
23-26	**latest production time flag**	4	integer; {time not used(0); time used(1)}
27-34	**latest production time**	8	CDS time used in the SLE CLTU IF
35-38	**spare**	4	not used
39 - size -1	**TC**	Up to 3K	CLTU

Table 6-8 TC CLTU response

octets	Field name	Size (octets)	Comment
0-7	**time**	8	CDS time format; time tag generated by the GS equipment if present; time of reception of the response otherwise.
8-10	**spare**	3	not used
11-14	**TC Id**	4	integer; CLTU Id

15	**acknowledgement**	1	integer; processing stage and result: { 0 = confirm accept 1 = confirm transmit 3 = failure accept 4 = failure transmit 9 = rejected by uNIS }
16	**reason**	1	integer; reject reason if failure; { 1 = rejected by SLE provider 2 = rejected by uNIS (SI not active) 3 = rejected by uNIS for other reasons. (SLE SI status, SLE API error, etc. Details are reported by uNIS as log messages.) }
17	**spare**	1	not used
18-21	**next TC Id**	4	integer
22-25	**spare**	4	not used

6.4 MON Interface (Administrative messages)

For a given uNIS instance, Administrative Data Units (uNIS ADMINDU) generated by any active SLE Service instance are transmitted from the uNIS to the CCS over a TCP/IP connection. The uNIS acts as a server. The CCS initiates the connection. No authentication of clients is performed by the uNIS.

The Mon port number is specified as part of the uNIS settings (see *monPort* settings below).

There is no limit to the number of clients that can connect to the TM port.

If no client is connected on a port and data is received from a GS, data are discarded.

It is possible to configure whether the Mon port is active or not (*useAdminPort* settings).

The ADMINDU[3] consists successively of a header and a message text (see the following table).

Table 6-9 Administrative Messages (header + text)

octets	Field name	size (octets)	Comment
0-3	**packet size**	4	integer; total length of the DU (in bytes)
4-11	**time**	8	CCSDS time format
12-13	**type**	2	integer; { 0 = TM info; 1 = TC info}
14-15	**severity**	2	integer; { 0 = info ;1 = warning; 2 = alarm }

[3] The format of the uNIS ADMINDU follows NIS-MCS-ICD (Section 4. Administrative Messages). The uNIS interface handles a subset of the messages foreseen by the NIS-MCS-ICD, namely: TM (with reference to NIS-MCS-ICD table 2.15): Event Ids: 1, 2, 3, 4, 5, 15; TC (with reference to NIS-MCS-ICD table 2.18): Event Ids: 1, 2, 3, 4, 5.

16-19	**event id**	4	integer identifying the event; see (TM) and (TC)
20	**message text**		Character string ASCII text and parameters. See the two following tables, for TM and TC related messages.
(size -1)	**end**	1	\0

Table 6-10 Messages Related to TM

Event id	Severity	Message
1	Info	`Established TM link <channel> <data type> to <GS equipment name>`
2	Warning	`Failed to establish TM link <channel> <data type> to <GS equipment name>: <connection reason>`
3	Info	`Closed TM link <channel> <data type> to <GS equipment name>`
4	Info	`Aborted TM link <channel> <data type> to <GS equipment name>`
5	Warning	`<GS equipment name> aborted TM link <channel> <data type>`
6	Info	`First data received on TM link <channel> ONLT from <GS equipment name>`
7	Warning	`Data gap on TM link <channel> ONLT from <GS equipment name>. Reason: <DG reason> Size: <DG Size>`
8	Info	`End of data on TM link <channel> ONLT from <GS equipment name>: end of pass`
9	Info	`End of data on TM link <channel> <data type> from <GS equipment name>: <end of data reason>`
10	Info	`<channel> is <status> reported from <GS equipment name>`
11	Warning	`Drop on TM link <channel> <data type> from <GS equipment name>`
12	Warning	`Bad data on TM link <channel> <data type> from <GS equipment name>`
13	Info	`Good data on TM link <channel> <data type> from <GS equipment name>`
14	Warning	`TM <channel> <data type> from <GS equipment name> cannot be delivered to MCS`
15	Info	`<status report>`

Table 6-11 TM Message Parameters

parameters	size (octets)	comment
channel	9	ASCII characters space link channel (‘SLC’); bad frames (‘BadFrames’); master channel xxx, where xxx is the MC id (‘MC xxx’); virtual channel xxx/y, where xxx is the MCid, and y the VCid (‘VC xxx/y’)

data type	7	ASCII characters, one of : • 'Onl-TIM' for on-line timely data streams • 'Onl-COM' for on-line complete data streams • 'Offline' for Off-line data streams;
GS equipment name	8	ASCII characters
connection reason	25	ASCII characters {'cannot connect control ch'; 'cannot connect data ch'; 'cannot start tx'} ch: channel; tx: transfer
status	10	ASCII characters {'not-locked'; 'inactive'; 'bad'; 'active'; 'idle'}
end of data reason	18	ASCII characters {'end of criteria'; 'end of stored data'}
status report	146	For RAF data streams, this field is as follows (as one single string): " #fr: <numFrames>; #err-free fr: <numErrorFreeFrames>; fr.synch: <frameSyncLock>; carr.: <carrierDemodLock>; subcarr.: <subCarrierDemodLock>; sym.synch: <symbolSyncLock>; prod.status: <productionStatus>;"[3]
	125	For RCF data streams, this field is as follows (as one single string): " #fr: <numFrames>; fr.synch: <frameSyncLock>; carr.: <carrierDemodLock>; subcarr.: <subCarrierDemodLock>; sym.synch: <symbolSyncLock>; prod.status: <productionStatus>;" where: <numFrames> is a 5 char integer value <numErrorFreeFrames> is a 5 char integer value <frameSyncLock> is a 11 char string <carrierDemodLock> is a 11 char string <subCarrierDemodLock> is a 11 char string <symbolSyncLock> is a 11 char string <productionStatus> is a 8 char string The values for the 4 <….Lock> fields are one of the following string (11 characters left justified, padded with spaces): "in lock" "out of lock" "not in use" "unknown" The values for the <productionStatus> field are either 'active' or inactive' (8 characters left justified, padded with spaces)

Table 6-12 Messages Related to TC

event id	Severity	Messages
1	Info	`Established TC link to <GS equipment name>` Where< GS equipment name> is a 8 ASCII characters string
2	Warning	`Failed to establish TC link to <GS equipment name>`
3	Info	`Closed TC link to <GS equipment name>`
4	Info	`Aborted TC link to <GS equipment name>`

7

7 Configure the uNIS

The default uNIS properties for a given G/S <<GSID>> (e.g. “UnisTest1”) have to be customised according to the local environment using the CCS Settings editor as described below.

Table 7-1 uNIS Settings

Property	Description	Format
verbose	When value is 1 (0), it enables (disables) production of resource consuming debug messages (e.g. TM frame dumps). To be used only for trouble shooting activities. This affects the uNIS application logging and not the SLE API component underneath (controlled by the Tracing property, see below).	Unsigned integer: 0 or 1
SLEAPI section This section includes the properties required by the SLE API component, as follows:		
Tracing	Specifies whether the SLE API shall be started with tracing option fully enabled (value 1) or not (value 0). In such cases all SLE related actions will be logged to standard output. Warning: enabling this option can significantly reduce application performances. To be used only for trouble shooting activities. SLEAPI tracing is activated/deactivate with this flag independently of the verbose setting above.	Unsigned integer: 0 or 1
ground station sections (named with user defined strings) These sections (one for each G/S) define the SLE service specific properties for a uNIS instance. Each G/S subsection is named <<GSID>>, i.e. a unique string that identifies the G/S (e.g. KIRUNA). Such a subsection requires a corresponding GSFE driver section to be configured with the same <<GSID>> name and matching tmPort, tcPort, monPort (if used) values. The section includes a number of properties:		
SICFDirectory	path to the directory that contains SICF file(s) to be processed by the uNIS for the given G/S	string
SleProxyUser	The location of the SLE API proxy definition file (file path relative to $CCSTESTENV. Example: USER/SleProxyUser.txt) An example of such file can be found under TestPacks\CcsSelfTest\USER The same file is reported below (SLE API Proxy File section).	string

ResponderId	Used by uNIS at RAF/RCF/CLTU-BIND to identify the SLE Provider See responder-identifier in [CCSDS-SLE-RAF/RCF/CLTU]. This value has to match one of the REMOTE_PEERS/ID specified in the SLE API Proxy file (see below). This value is used by the uNIS instance to identify the SICF blocks applicable to its G/S	string
SleProtocolVersionRAF	Parameter that identifies the version number of the RAF service specification that is to govern the association if RAF-BIND succeeds. Values: 1-4. See version-number in [CCSDS-SLE-RAF].	Unsigned integer
SleProtocolVersionRCF	As above for RCF. Values: 1-4. See version-number in [CCSDS-SLE-RCF].	Unsigned integer
SleProtocolVersion_CLTU	As above for CLTU. Values: 1-3. See version-number in [CCSDS-SLE- CLTU].	Unsigned integer
tmPort	TCP/IP port number for CCS/TSC to connect for reception of TM Data from uNIS.	Unsigned integer, 16 bits.
tcPort	TCP/IP port number for CCS/TSC to connect for delivery of CLTU requests and reception of CLTU responses from uNIS.	Unsigned integer, 16 bits.
monPort	TCP/IP port number for CCS/TSC to connect for delivery of Administrative messages from uNIS.	Unsigned integer, 16 bits.
useAdminPort	This parameter specifies whether the Mon (Administrative) port shall be activated or not. Notice that, as the Tcl control interface was added to uNIS, the Mon interface is no longer used by CCS5, however this interface has been kept for backward compatibility	Bool: true/false.
tclPort	TCP/IP port number for CCS/TSC to connect for uNIS control operations via Tcl commands.	Unsigned integer, 16 bits.
RouteID	Ground Station code, set in the TM Data Unit returned to CCS/TSC	Unsigned integer, 32 bits size.
RequestedFrameQuality	Requested Frame Quality (it applies to RAF service): the frame quality criteria, set by the RAF-START operation, used to determine which frames are selected for delivery. See requested-frame-quality parameter description in [CCSDS-SLE-RAF] Possible values: ALL, GOOD, ERRED.	string

ProduceNotification	This parameter applies to CLTU service and specifies whether the SLE provider shall invoke the CLTU-ASYNCNOTIFY operation upon completion of the radiation of the CLTU. Such a CLTU-ASYNC-NOTIFY operation turns out into a CLTU response produced by the uNIS and sent to CCS (UV2). See 'report' parameter of CLTU-TRANSFER-DATA operation in [CCSDS-SLE- FCLTU].	Bool: true/false.
CltuDelay	This parameter applies to CLTU service. It shall contain the minimum radiation delay, in microseconds, between the CLTU transferred in this operation and the next CLTU . See 'delay-time' parameter of CLTU-TRANSFER-DATA operation in [CCSDS-SLE- FCLTU].	Unsigned integer
StatusReportReportingCycle	The parameter specifies the requested interval between RAF/RCF/CLTU-STATUS-REPORT status reports in seconds. If the parameter value is set to 0 no status report is requested to the SLE Provider. Notice that the production of this report is vital to the proper update of the service arrays (see section 6.1.2). Notice that the value set for *StatusReportReportingCycle* has to be in the value range accepted by the SLE Provider (this is part of the SLE API configuration on the Provider side: System Element file, parameters: *MIN_REPORTINGCYCLE*, *MAX_REPORTINGCYCLE*).	Unsigned integer

An example of the CCS Setting Editor for the uNIS section is given in the following picture (the example shows two G/S subsections : TestUnis1 and TestUnis2).

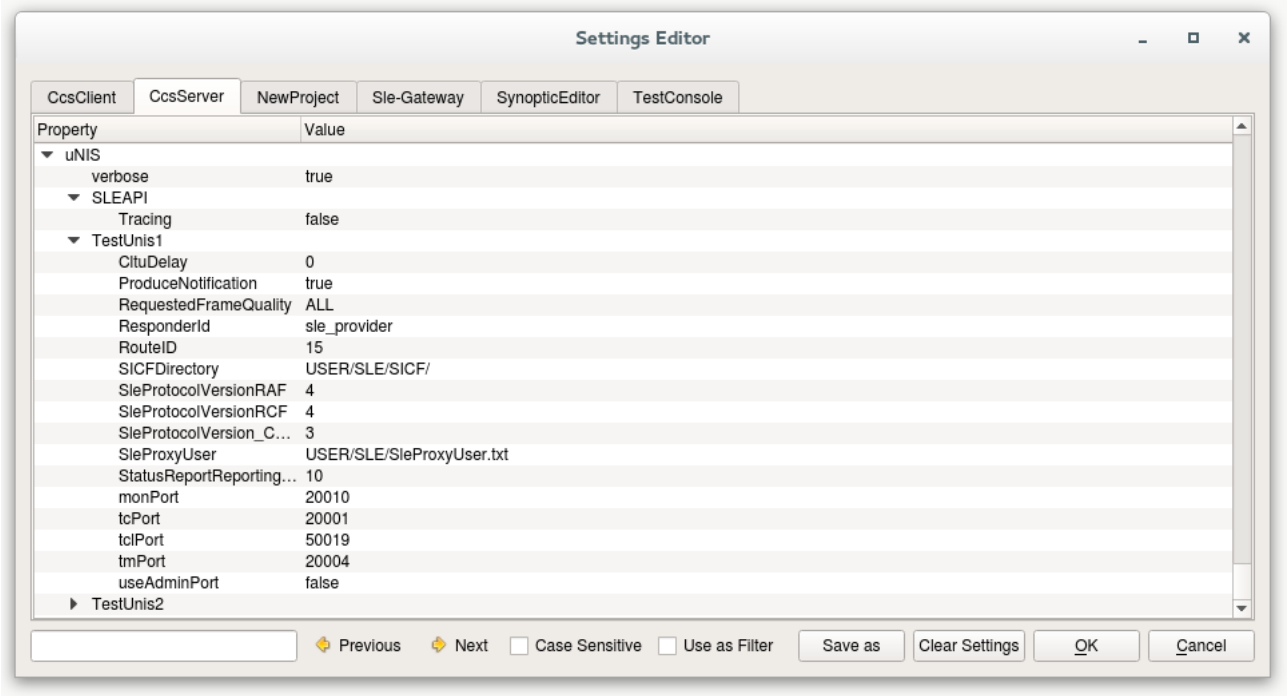

Figure 7-1 example of uNIS settings for two ground stations

8 SLE API configuration

8.1 Proxy File

The Proxy File provides the SLE API with the necessary configuration information, like:

- The SLE BIND Role used by the application (initiator in case of uNIS);
- The Local Application Identification (authority identifier, password);
- The SLE Service Type List (supported SLE services, RAF, RCF and CLTU in the case of uNIS);
- The list of Peer Applications (i.e. SLE Providers) with specification of:
  - Authority identifier
  - Authentication mode (None, Bind, All)
  - Password
- The list of Foreign Responder Ports:
  - Logical port identifier
  - Socket list (list of IP address and TCP port number)
  - TCP buffer (receive and transmit buffer sizes)
  - Heartbeat Interval and Dead-Factor
- Output Queue Length (number of data units PDUs to queue)
- Maximum Authentication Delay
- Maximum Trace Length (length of dumps in traces, if SLE API tracing is active)

Most of this data does not need to change, however, for every project some values have to be customized. In particular:

- The Local Application Identification
- The list of Peer Applications
- The list of Foreign Responder Ports

The following text reports an example of Proxy File (SleProxyUser.txt). Sections that require project specific settings are indicated with the tag <<Project Specific>>.

```
# ------------------------------------------
# ESA SLE API Package
# Proxy Configuration File - USER
# Version: $Id:   $
# ------------------------------------------

PROXY_ROLE = INITIATOR
#
# local application identifier.
# It has to match what expected by SLE Provider (REMOTE_PEERS: ID) <<Project
Specific>>
LOCAL_ID = SLE_USER
# local password (in hex nibbles)
LOCAL_PASSWORD = ffdf01449809e4e5e677818892
#
# list of the supported services/versions
SERVER_TYPES ={
   SRV_ID = RAF
```

```
      SRV_VERSION = {
           1
           2
           3
           4
      }

      SRV_ID =  RCF
      SRV_VERSION = {
             1
             2
             3
             4
        }


      #
      SRV_ID = CLTU
      SRV_VERSION = {
         1
         2
      }

}
#
# list of remote peers. Depends on the available SLE Provider(s) <<Project
Specific>>
REMOTE_PEERS = {
   # peer username
   ID = SLE_PROVIDER
   # peer password (in hex nibbles, to allow all characters)
   PASSWORD = 000102030405060708090a0b0c0d0e0f
   # authentication mode (NONE or BIND or ALL)
   AUTHENTICATION_MODE = NONE
   #
   ID = SLE_PROVIDER1
   PASSWORD = ff23a67babcd01
   AUTHENTICATION_MODE = NONE
   #
}
#
# list of foreign responder ports. It depends on the available SLE Provider(s)
<<Project Specific>>
FOREIGN_LOGICAL_PORTS = {
   # logical port name
   PORT_NAME = PORT_TM1
   # heartbeat timer value
   PORT_HEARTBEAT_TIMER = 0
   # heartbeat dead factor
   PORT_DEAD_FACTOR  = 5
   # IP address (host:port)
   IP_ADDRESS = {
#        [ip-address:port-number] has to match the host:port where the SLE provider
makes available PORT_TM1 <<Project Specific>>
     127.0.0.1:5008
   }
   TCP_XMIT_BUFFER_SIZE = 32768
   TCP_RECV_BUFFER_SIZE = 32768
#
   PORT_NAME = PORT_TC1
   PORT_HEARTBEAT_TIMER = 0
   PORT_DEAD_FACTOR  = 5
   IP_ADDRESS = {
```

```
#      [ip-address:port-number] has to match the host:port where the SLE provider
makes available PORT_TC1 <<Project Specific>>
     127.0.0.1:5010
    }
   TCP_XMIT_BUFFER_SIZE = 16384
   TCP_RECV_BUFFER_SIZE = 16384
}
#
# maximum transmit queue size (PDUs)
TRANSMIT_QUEUE_SIZE = 1000
#
# maximum authentication delay (seconds)
AUTHENTICATION_DELAY = 180
#
# maximum length of trace output for strings (characters)
MAX_TRACE_LENGTH     = 256
#
# ----------------- end of file ----------------
```

9 SICF configuration

The uNIS application requires one or more SICF file(s) to be located in a directory on the disk. The SICF format is described in [SICF-ICD]. Notice that, although it is in general allowed to have for each configured G/S (i.e. uNIS instance) a separate SICF folder, it is possible to store SICF files for all configured G/S's (i.e for all uNIS instances) in the same directory. Each uNIS instance is capable of filtering, from each file in the SICF folder, only the sections applicable to the Responder Id the uNIS instance is configured for.

SICF files are read by uNIS at startup.

Examples of SICF file are reported below.

```
BEGIN_OBJECT = SLE-SI-CONFIGURATION;
DESCRIPTION = "Example Service Instance Configuration file - RAF";
REQUESTER_NAME = "Admin";
CREATION_DATE = 2017-101T12:00;
VERSION = 01;

BEGIN_GROUP = R-AF-TS-P;
          service-instance-id = "sagr=SAGR.spack=SPACK.rsl-fg=RSL-FG.raf=onlc1";
          service-instance-start-time = VOID;
          service-instance-stop-time = VOID;
          initiator-id = "sle_user";
          responder-id = "sle_provider";
          responder-port-id = "Harness_Port_onlc1";
          return-timeout-period = 30;
          delivery-mode = COMPLETE_ONLINE;
          initiator = USER;
          permitted-frame-quality = "allFrames.goodFramesOnly.erredFramesOnly";
          latency-limit = 1;
          transfer-buffer-size = 10;
     END_GROUP = R-AF-TS-P;


END_OBJECT = SLE-SI-CONFIGURATION;
```

```
BEGIN_OBJECT = SLE-SI-CONFIGURATION;
DESCRIPTION = "Example Service Instance Configuration file - CLTU";
REQUESTER_NAME = "Admin";
CREATION_DATE = 2016-300T12:00;
VERSION = 01;

   BEGIN_GROUP = F-CLTU-TS-P;
          service-instance-id = "sagr=SAGR.spack=SPACK.fsl-fg=FSL-FG.cltu=cltu1";
          service-instance-start-time = VOID;
          service-instance-stop-time = VOID;
          initiator-id = "sle_user";
          responder-id = "sle_provider";
          responder-port-id = "Harness_Port_Cltu_1";
          return-timeout-period = 20;
```

```
          maximum-cltu-length = 2000;
          minimum-cltu-delay = 0;
          maximum-cltu-delay = 5000000;
          bit-lock-required = FALSE;
          rf-available-required = FALSE;
          protocol-abort-clear-enabled = TRUE;
     END_GROUP = F-CLTU-TS-P;


   BEGIN_GROUP = F-CLTU-TS-P;
          service-instance-id = "sagr=SAGR.spack=SPACK.fsl-fg=FSL-FG.cltu=cltu2";
          service-instance-start-time = VOID;
          service-instance-stop-time = VOID;
          initiator-id = "sle_user";
          responder-id = "sle_provider_2";
          responder-port-id = "Harness_Port_Cltu_2";
          return-timeout-period = 20;
          maximum-cltu-length = 2000;
          minimum-cltu-delay = 0;
          maximum-cltu-delay = 5000000;
          bit-lock-required = FALSE;
          rf-available-required = FALSE;
          protocol-abort-clear-enabled = TRUE;
     END_GROUP = F-CLTU-TS-P;

END_OBJECT = SLE-SI-CONFIGURATION;
```

```
BEGIN_OBJECT = SLE-SI-CONFIGURATION;
DESCRIPTION = "Example Service Instance Configuration file - RCF";
REQUESTER_NAME = "Admin";
CREATION_DATE = 2016-300T12:00;
VERSION = 01;

     BEGIN_GROUP = R-CF-TS-P;
          service-instance-id = "sagr=SAGR.spack=SPACK.rsl-fg=RSF-FG.rcf=onlc1";
          service-instance-start-time = VOID;
          service-instance-stop-time = VOID;
          initiator-id = "sle_user";
          responder-id = "sle_provider";
          responder-port-id = "Harness_Port_RCF_onlc1";
          return-timeout-period = 30;
          delivery-mode = COMPLETE_ONLINE;
          initiator = USER;
          permitted-vcids = "(3,0,7).(3,0,0).(3,0,*)";
          latency-limit = 1;
          transfer-buffer-size = 10;
     END_GROUP = R-CF-TS-P;

END_OBJECT = SLE-SI-CONFIGURATION;
```

10 Critical Errors

This section lists major errors that can occur at uNIS startup, due to inconsistent configuration.

Table 10-1 Critical errors

Error Message	Description
Error Parsing SICF Files	The Service Instance Configuration File (SICF) contains syntax errors. SICF shall follow the format specified in SICF-ICD.
Cannot Find any Service Instances in [sicf] for ResponderId: [rid]	A uNIS instance (for a given G/S Identifier) has processed all SICF files in folder [sicf] (as specified in uNIS settings: SICFDirectory) but did not find any Responder Identifier applicable to such uNIS instance: [rid] (defined in uNIS settings as ResponderId).
Cannot find SLE API PROXY file under: [path]	The SLE API Proxy configuration file could not be found at the path [path] specified in settings (SleProxyUser).
Configuration of SLE Proxy failed. File: [filename] - result: [diagn]	The SLE API Proxy configuration file contains errors See section 8.1. Details are shown in [diagn], as reported by the SLE API, responsible for processing such file.
[servicename] service listen failed	TM, TC or MON interface to CCS (specified in [servicename]) cannot be started. This is normally due to TM, TC and/or MON ports (specified respectively by uNIS settings: tmPort, tcPort and monPort) being not available for setting up a TCP/IP server. This can happen, for instance if another uNIS instance is already running for the same ground station, if any of the service ports are taken by other applications on the same workstation, or if the specified port number is outside the range allowed by the Operating System.
Cannot set up CCS Tcl control interface	The Tcl Control interface to CCS cannot be started. This is normally due to port tclPort (specified in uNIS settings) being not available for setting up a TCP/IP server. See note in similar case above.

Annex A - Thapi test tool

Annex A Thapi test tool

The SLE API Test harness (Thapi) is an interactive tool that can be used for testing the SLE API in provider or in user role.

Usage on Windows:

```
Thapi.exe -x <proxy database> -s <service element database> [-a <command file>] [-t <tracelevel>] [-d <data file>]
```

Usage on Linux:

```
Thapi -x <proxy database> -s <service element database> [-a <command file>] [-t <tracelevel>] [-d <data file>]
```

By convention, square braces enclose optional arguments, other arguments are mandatory.

Where:

- `<proxy database>`: SLE API proxy definition file path
- `<service element database>`: SLE Service Element definition file
- `<command file>`: test script file path. If a test script is specified the tool runs in batch mode and it is script driven, otherwise the tool works in interactive mode;
- `<data file>`: path to an input data file containing either TM frame (RAF, RCF services) or CLTU definitions (CLTU service). If no data file is specified the tool will create pre-defined TM frames / CLTUs;
- `<tracelevel>`: trace level (0=low, 1=medium, 2=high, 3=full) (-t is optional);

Concerning the option `-d <data file>`: a data file is an ASCII file containing a list of data elements (hexadecimal format). Comments are allowed, beginning with "#". Examples:

```
### Example of CLTU data file for THAPI testing
### Each CLTU frame is written in hex form on separate line
eb90fc099e4e61281ebc39ac802fa7285df20fb8558da96dc294c5c5c5c5c5c5
eb90fc099e4e61281ebc39ac802fa7285df20fb8558da96dc294c5c5c5c5c5c5
## comment
eb90fc099e4e62281e4a39ac832fa7285d460fb85545dc8ee752c5c5c5c5c5c5
eb90fc099e4e62281e4a39ac832fa7285d460fb85545dc8ee752c5c5c5c5c5c5

### Example of TM data file for THAPI testing
### Each TM frame is written in hex form on separate line
0102030405060708090A0B0C0E0F10110102030405060708090A0B0C0E0F101101020304
05060708090A0B0C0E0F10110102030405060708090A0B0C0E0F1011
02AABBCCDD02AABBCCDD02AABBCCDD02AABBCCDD02AABBCCDD02AABBCCDD02AABBCCDD02
AABBCCDD02AABBCCDD02AABBCCDD02AABBCCDD02AABBCCDD02AABBCCDD02AABBCCDD
## Comment
03EEFF0102030405
```

The hexadecimal definition of TM frames and CLTU is case insensitive, e.g.: both 0a0b0c and 0A0B0C are valid values.

Note: when executing the Thapi in SLE Provider role, the **Communication Server** process (SLECS) has to be started beforehand:

Windows:
```
slecs.exe –d <proxy database>
```

Linux:
```
slecs –d <proxy database>
```

An error message reported by the Thapi such as: `Result: SLE_E_PORT ######`
is a symptom that the SLECS is not running.

Optionally the Default Logger process (SLEDFL) can be started to inspect trace/log messages:

Windows:
```
sledfl.exe -d <proxy database> -t 0
```

Linux:
```
sledfl –d <proxy database> -t 0
```

A.1 Using the Test Harness

After starting the application, Thapi can be controlled by command line inputs (or in batch mode, if selected at startup). The supported commands are listed below. After given a command further inputs may be required. In this case the Test Harness displays a short help about the details of each command.
For example after initiating the sending of a BIND Operation with the command “bind” further setup of the operation is required. The possible setup commands are shown by the Test Harness directly after the bind-command.

In the following: (u), (p) specify that the command is only applicable to the tool running in User or Provider role, respectively.

A.1.1 General Commands

initialise	initialises the Builder
start	starts the Builder
terminate	terminates the Builder
create_si	creates a service instance
destroy_si	destroys a service instance
list_si	lists all created service instances
shutdown	performs shutdown of the Builder
wait_event_all_si	Wait for the next event from DCW
help	this help text
exit	exits the test program
down	down to service instance commanding
up	up to service element commanding
sii_base_rtn	set base sii for return SI
sii_base_fwd	set base sii for forward SI

start_rec	starts recording the commands to a file
stop_rec	stops recording commands
playback	starts playback of previously recorded commands
wait	Wait for n seconds
wait_event	Wait for the next event from DCW
wait_selected_op	Wait for a given operation from DCW
start_loop_sequence	Start recording the loop sequence
stop_loop_sequence	Stop the recording of the loop sequence
play_loop_sequence	Play the recorded loop sequence
test_td	Test Transfer Data and Resume Data Transfer
print	print contents of the SI

SI Admin

set_siid	set service instance identifier
set_peer_id	set peer identifier
set_pp	set provision period
set_bind_ini	set bind initiative
set_rsp_port_id	set responder port identifier
set_rtn_to	set return timeout
config_completed	config completed
send_a_return	Send a return operation from the list
send_all_return	Send all stored return operations
timeoffset	Set offset for Timesource
help	this help text
suspend	suspends the reception of operations other than PEER-ABORT
resume	resumes the reception of operations

A.1.2 Commands for CLTU Service Instances

SI Admin

set_blr	set bit lock required
set_maxl	set max Sldu length
set_mf	set modulation frequency
set_mi	set modulation index
set_plop	set plop in effect
set_rfr	set RF lock required
set_scbrr	set subcarr. to bitrate ratio
set_mbs	set max buffer size
set_init_ps	set initial production status
set_init_uls	set initial uplink status
set_nm	Set_NotificationMode
set_acqsl	set acquisition sequence length
set_pam	set protocol abort mode
set_mdt	set minimum delay time
set_cpc	set clcw physical chanel
set_cgv	set clcw global vcid
set_plop1_idle_sl	set plop1 idle sequence length

SI Update

cltu_started	Cltu started
cltu_ns	Cltu not started
cltu_rad	Cltu radiated
cltu_aborted	Cltu aborted
set_ps	Production status change
set_uls	set uplink status
buffer_empty	Buffer empty
evt_proc_compl	Event proc completed

CLTU Operations

bind	(u) send CLTU-BIND operation
unbind	(u) send CLTU-UNBIND operation
start	(u) send CLTU-START operation
stop	(u) send CLTU-STOP operation
td	(u) send CLTU-TRANSFER-DATA operation
an	(p) send CLTU-ASYNC-NOTIFY operation
ssr	(u) send CLTU-SCHEDULE-STATUS-REPORT operation
gp	(u) send CLTU-GET-PARAMETER operation
te	(u) send CLTU-THROW-EVENT operation
peer_abort	(u/p) send CLTU-PEER-ABORT operation
auto_gen_td	(u) send CLTU-TRANSFER-DATA operations automatically

upon td the following options are available:

ert	set-up earliest radiation time. Further argument: time (format YYYT-MM-DDTHH:MM:SS, e.g. 2015-05-01T10:00:00)
lrt	set-up latest radiation time. Further argument: time (format YYYT-MM-DDTHH:MM:SS, e.g. 2015-05-01T10:00:00)
dt	set-up delay time. Further argument: delay in milliseconds (integer)
rn	set-up radiation notification. Further argument: values: 0=produce notif., 1=do not produce notif., -1=invalid.
id	set-up CLTU id for the CLTU to be transferred. Further argument: CLTU ID (integer)
eid	set-up next expected CLTU id (for return-PDU). Further argument: next CLTU ID (integer)
sp	set-up positive result
sd	set-up diagnostic
loa	load CLTUs from ASCII file (requires program argument -d <data-filename> to be set at startup)
dl	set-up data length (only if ‘loa’ is not selected, see above). Further argument: length (integer). In such case a pre-defined CLTU with the given length is created by the tool
ok	set-up completed

A.1.3 Commands for FSP Service Instances

SI Admin

set_mbs	set max buffer size
set_mpl	set max packet length
set_mfl	set max frame length
set_init_ps	set initial production status

set_apidl	set APID list
set_bto	set blocking timeout
set_bu	set blocking usage
set_die	set directive invoc. enabled
set_init_dio	set initial directive invoc. online
set_mapl	set MAP list
set_shp	set segment header present
set_vcms	set VC mux scheme
set_vcpv	set VC polling vector
set_vcpril	set VC priority list
set_vc	set VC
set_ptm	set perm transmission mode

FOP monitor

set_slw	Set FOP sliding window
set_tot	Set timeout type
set_tinit	Set timer initial
set_trl	Set transmission limit
set_tfsc	Set transmitter frame sequence count
set_fops	Set FOP state
set_muxs	Set MAP multiplex scheme
set_muxpv	Set MAP polling vector
set_muxpril	Set MAP priority list

SI Update

pkt_started	Packet started
pkt_ns	Packet not started
pkt_rad	Packet radiated
pkt_ack	Packet acknowledged
set_ps	Production status change
vc_abort	VC aborted
no_dir	No directive capability
buffer_empty	Buffer empty
evt_proc_compl	Event proc completed
dir_compl	Directive completed
dir_online	Directive capability online

FSP Operations

bind	(u) send FSP-BIND operation
unbind	(u) send FSP-UNBIND operation
start	(u) send FSP-START operation
stop	(u) send FSP-STOP operation
td	(u) send FSP-TRANSFER-DATA operation
an	(p) send FSP-ASYNC-NOTIFY operation
ssr	(u) send FSP-SCHEDULE-STATUS-REPORT operation
gp	(u) send FSP-GET-PARAMETER operation
te	(u) send FSP-THROW-EVENT operation
dir	(u) send FSP-INVOKE-DIRECTIVE operation
peer_abort	(u/p) send FSP-PEER-ABORT operation
auto_gen_td	(u) send FSP-TRANSFER-DATA operations automatically

A.1.4 Commands for RAF Service Instances

SI Admin

set_dm	set delivery mode
set_ll	set latency limit
set_buffer_size	set transfer buffer size
set_init_ps	set initial production status
set_init_fsl	set initial frame sync lock
set_init_cdml	set initial carrier demod lock
set_init_scdl	set initial sub carrier demod lock
set_init_ssl	set initial symbol sync lock

SI Update

set_ps	set production status
set_fsl	set frame sync lock
set_cdml	set carrier demod lock
set_scdl	set subcarrier demod lock
set_ssl	set symbol sync lock

RAF Operations

bind	(u) send RAF-BIND operation
unbind	(u) send RAF-UNBIND operation
start	(u) send RAF-START operation
stop	(u) send RAF-STOP operation
td	(p) send RAF-TRANSFER-DATA operation
sb	(p) force sending of Transfer Buffer
sn	(p) send RAF-SYNC-NOTIFY
ssr	(u) send RAF-SCHEDULE-STATUS-REPORT operation
gp	(u) send RAF-GET-PARAMETER operation
peer_abort	(u/p) send RAF-PEER-ABORT operation
auto_send_td	(p) send RAF-TRANSFER-DATA operations automatically
auto_recv_td	(u) receive RAF-TRANSFER-DATA operations automatically

upon td the following options are available:

fq	set-up frame quality. Further parameter: 0=good, 1=erred, 2=undetermined, -1=invalid.
dlc	set-up data-link-continuity. Further parameter: integer value > -2, as per CCSDS standard.
loa	load TM Frames from ASCII file (requires program argument: -d <data-filename> to be set at startup)
dl	set-up data length. (only if 'loa' is not selected, see above). Further argument: length (integer). In such case a fixed TM frame with the given length is created by the tool (0x303132….)
panno	set-up private annotation. Further parameter: annotation length in bytes (integer value)
pico	set-up picosecond resolution
ok	set-up completed

A.1.5 Commands for RCF Service Instances

SI Admin

set_dm	set delivery mode
set_ll	set latency limit
set_buffer_size	set transfer buffer size
set_init_ps	set initial production status
set_init_fsl	set initial frame sync lock
set_init_cdml	set initial carrier demod lock
set_init_scdl	set initial sub carrier demod lock
set_init_ssl	set initial symbol sync lock
set_pgvcid	set permitted GVCID list

SI Update

set_ps	set production status
set_fsl	set frame sync lock
set_cdml	set carrier demod lock
set_scdl	set subcarrier demod lock
set_ssl	set symbol sync lock

RCF Operations

bind	(u) send RCF-BIND operation
unbind	(u) send RCF-UNBIND operation
start	(u) send RCF-START operation
stop	(u) send RCF-STOP operation
td	(p) send RCF-TRANSFER-DATA operation
sn	(p) send RCF-SYNC-NOTIFY
ssr	(u) send RCF-SCHEDULE-STATUS-REPORT operation
gp	(u) send RCF-GET-PARAMETER operation
peer_abort	(u/p) send RCF-PEER-ABORT operation
auto_gen_td	(p) send RCF-TRANSFER-DATA operations automatically

upon td the following options are available:

dlc	set-up data-link-continuity. Further parameter: integer value > -2, as per CCSDS standard.
loa	load TM Frames from ASCII file (requires program argument: -d <data-filename> to be set at startup)
dl	set-up data length. (only if 'loa' is not selected, see above). Further argument: length (integer). In such case a fixed TM frame with the given length is created by the tool (0x303132….)
pico	set-up picosecond resolution
ok	set-up completed

A.1.6 Commands for ROCF Service Instances

SI Admin

set_dm	set delivery mode
set_ll	set latency limit
set_buffer_size	set transfer buffer size
set_init_ps	set initial production status
set_init_fsl	set initial frame sync lock
set_init_cdml	set initial carrier demod lock
set_init_scdl	set initial sub carrier demod lock
set_init_ssl	set initial symbol sync lock

set_pgvcid	set permitted GVCID list
set_pcwts	set permitted control word type list
set_ptcvcid	set permitted TC VcId list
set_pums	set permitted update mode list

SI Update

set_ps	set production status
set_fsl	set frame sync lock
set_cdml	set carrier demod lock
set_scdl	set subcarrier demod lock
set_ssl	set symbol sync lock
set_nfp	set number of frames processed

ROCF Operations

bind	(u) send ROCF-BIND operation
unbind	(u) send ROCF-UNBIND operation
start	(u) send ROCF-START operation
stop	(u) send ROCF-STOP operation
td	(u) send ROCF-TRANSFER-DATA operation
sn	(p) send ROCF-SYNC-NOTIFY operation
ssr	(u) send ROCF-SCHEDULE-STATUS-REPORT operation
gp	(u) send ROCF-GET-PARAMETER operation
peer_abort	(u/p) send ROCF-PEER-ABORT operation
auto_gen_td	(u) send ROCF-TRANSFER-DATA operations automatically

A.2 Thapi validation scripts

The uNIS installation folder includes a Thapi test bed directory structure (thapi_testdata). Its absolute path is, for instance:

- /opt/uNIS/thapi_testdata (Linux)
- C:\Program Files\UNIS\thapi_testdata (Windows)

The bin folder includes two sets of scripts, which automatically execute a number of test procedures exercising various test scenarios:

- `test_suite.sh|bat` : exercises all SLE services
- `test_suite_2.sh|bat` : exercises RAF, RCF, CLTU only services

The usage of such scripts is:

```
[script name] [uNIS installation path] [test bed path] [cs]
```

Where:

`[script name]` is one of the above mentioned test scripts. The .sh suffix is used on Linux, and the .bat suffix is used on Windows.

`[uNIS installation path]`: uNIS installation folder, e.g:

LINUX: /opt/uNIS/;

Windows: C:\Program Files\UNIS\

`[test bed path]` : path to the Thapi test bed directory e.g:

LINUX: /opt/uNIS/thapi_testdata

Windows: C:\Program Files\UNIS\thapi_testdata

The option *cs* is used to start the communication server and default logger only (to be executed before the actual test procedure). The actual test is started by omitting the *cs* option.

Test log files reporting the result of the test for both user and provides sides are stored under `thapi_log` in the current working directory where the scripts were launched).