SPIRE-ESA-DOC-002291

f

D O C U M E N T

mdocument title/ titre du document

EGSE BASED ON SCOS 2000

THE EGSE & MISSION CONTROL SYSTEM

prepared by/*préparé par* Serge Valera

reference/*réference* TOS-EMG/01-1029/bm/sv
issue/*édition* 4
revision/*révision* 0
date of issue/*date d'édition* 2004-03-09
status/*état* Issue
Document type/*type de document* Software User Manual
Distribution/*distribution*

a

ESTEC
Keplerlaan 1 - 2201 AZ Noordwijk - The Netherlands
Tel. (31) 71 5656565- Fax (31) 71 5656040

A P P R O V A L

Title *titre*	EGSE based on SCOS 2000	issue 4 *issue*	revision 0 *revision*

author *auteur*	Serge Valera	date *date*	2004-03-09

approved by *approuvé by*	date *date*

C H A N G E L O G

reason for change / *raison du changement*	issue / *issue*	revision / *revision*	date / *date*
SCOS 2000 – Release 2.1.1	1	0	8th June 2001
Compliance to "SCOS 2000 Release 2.3e including ESOC patches 1, 2 and 3" with EGSE modules version 2.3eP3.	2	0	10th January 2003
Finalisation of 2.3eP3 updates	2	1	24th January 2003
Language changes for compatibility with Herschel/Planck Central Checkout System (TTA WO8) Correction of SCOS/EGSE gateway interface (ESTEC patches patch 1, 2, 3)	3	0	23rd July 2003
Add-ons for 2.3e P4 updates	3	1	1st September 2003
Finalisation of 2.3e P4 installation procedure	3	2	23rd September 2003
Add-ons for 2.3e P5 updates	3	3	15th October 2003
Final P5 updates	3	5	04th March 2004

C H A N G E R E C O R D

Issue: 4 Revision: 0

reason for change/*raison du changement*	page(s)/*page(s)*
Issue 3.0 – New language grammar to facilitate transfer of procedures to the Herschel/Planck Central Checkout System. Issue 3.1 – Add-ons of P3 to P4 installation procedures Issue 3.2 – Correction of 2.3eP4 installation procedure; Logica patch TBD installation procedure. Issue 3.3 to 3.5 – Includes: • The Herschel/Planck TC history/OOL server (i.e. Logica patch 05122002) • Enhancement of the SCOS gateway for tracing incorrect messages received from EGSE routers. • Enhancement of the TCO and EGSE subsystems to allow MISCconfig definition of the EPOCH used by a mission. • Update of the algorithm used to timetag incoming TM to comply with Herschel instrument OBT timing scenario (refer to MISCconfig variable **EGSE_TM_FILING_TIME_TYPE = 3**). • Update of TC History to display the TC userrequestid. • Update of MON Alphanumeric Display to display hexadecimal values with leading “0”.	Section 1.1 bullet 3, Section. 2 and 2.3 and 2.4.2.3, All the section 5

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Scope

This User Manual details the specific EGSE aspects of the EGSE version of SCOS 2000 prepared by the Agency and delivered to the Herschel/Planck Instrument developers for support in developing their EGSE.

This manual complements the generic SCOS2000 documentation and contains:

- A recommendation for installing the LINUX SUSE 7.3 (refer to Appendix D).
- The installation procedure in a standalone mode of the SCOS EGSE Release 2.3e P5 version delivered by ESTEC/TOS-EMG (refer to section 2).
- The complementary to SCOS instructions to configure the EGSE for a given mission, a given test session (refer to section 3).
- The instructions to start the SCOS/EGSE system and the TOPE procedure execution environment (refer to section 5).
- The TOPE language reference manual (refer to 6)
- A description of the EGSE specific functions, the layout of messages sent to and received from the SCOE system (refer to section 7).

1.2 Applicable and Reference Documents

See all SCOS2000 R2.3e Documentation

[A1]	ESTEC/TOS-EMG/98.015/PSH	EGSE Gateway ICD.
[R1]	ECSS-E70-31	ECSS - Ground Systems and Operations. Monitoring and Control Data Definition Standard.
[R2]	ECSS-E70-32	ECSS - Ground Systems and Operations. Procedure Definition Language.
[R3]	ECSS-E70-41	ECSS - Ground Systems and Operations. Telemetry and Telecommand Packet Utilization.
[R4]	SOE-EMCS-SUM-0001	SCOS-2000 2.3e EMCS SUM
[R5]	S2K-MCS-ICD-0001-TOS-GCI-Issue51+MIB	SCOS-2000 Database Import ICD Issue 5.1
[R6]	S2K-TER_TN-002	S2K EGSE Extensions Technical Note, issue Draft 1B from 15.02.2002.
[R7]		TCL/TK version 8.3 Reference Manual
[R8]	S2K-MCS-SUM-0002-TOS-GCI	Scos2000 Configuration and Installation Guide.
[R9]	S2K-MCS-ICD-0005-TOS-GCI	Scos2000 Command Injection Service ICD.

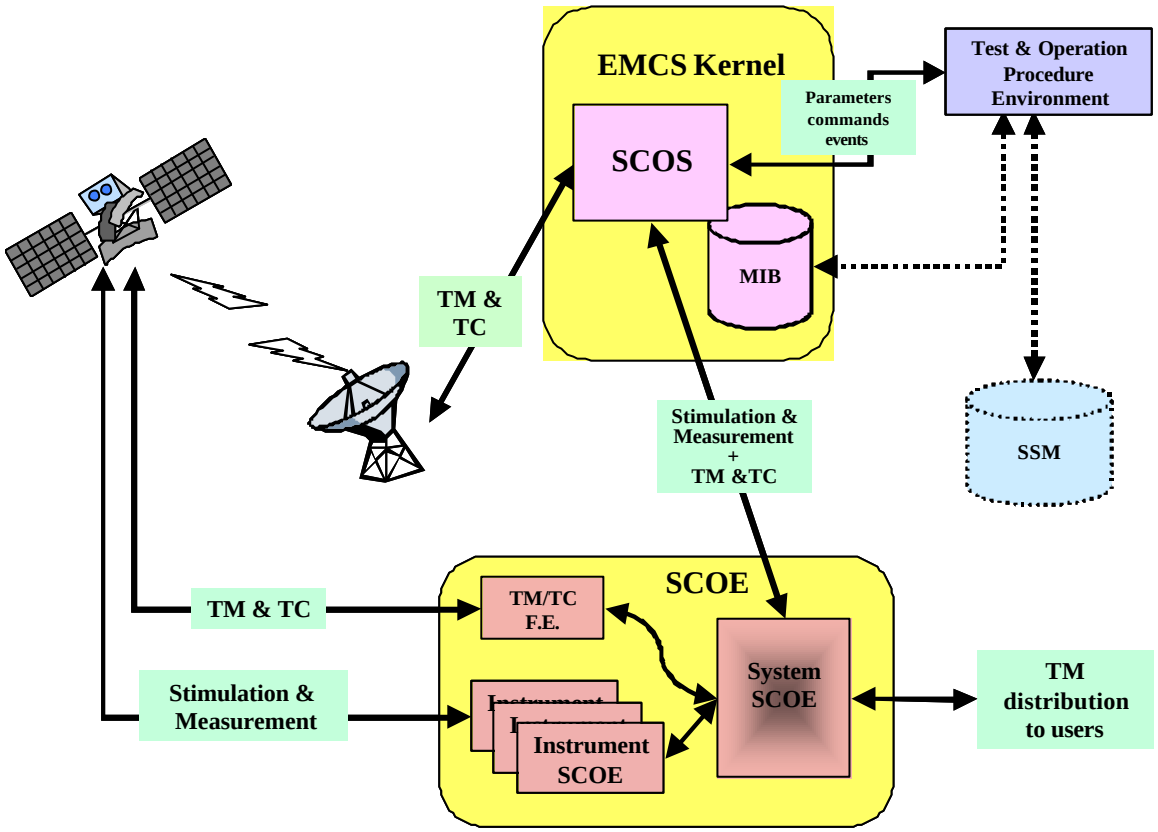
1.3 List of Acronyms

AIT	Assembly, Integration and Test
CCS	Central Check-Out System
COTS	Commercial off the shelf
EGSE	Electrical Ground-Support Equipment
EMCS	EGSE and Mission Control System
EXIF	SCOS External Interface services
MIB	Mission Information Base
PLUTO	Procedure Language for Users in Test and Operations
SCOS	Satellite Control System
SSM	Space System Model
TCL	Tool Command Language
TOPE	Test and Operations Procedure Environment
TOPE/CCS	Version of TOPE customised for the CCS.
TTA	Test Technology Activities

1.4 Overview

Within the PROBA spacecraft development, a common EGSE and Mission Control System has been designed and developed based on the SCOS2 system. This system has been used to test and to operate PROBA.

The EMCS system presented in this document is based on the PROBA approach, but uses SCOS 2000 in place of SCOS 2. The following picture gives an overview of the EMCS.



The EMCS is made of:

- Its kernel: the SCOS system and its internal Mission Information Base.
- The SCOE system.
- The Test and Operation Procedure environment.
- The Space System Model.

1.4.1 THE SPACE SYSTEM MODEL

The concept of Space System is introduced within the ECSS-E-00 standard. The Space System Model is an information system that contains mission information required by the EMCS and any user of the EMCS. The SSM will be specified within the ECSS-E-70-31 Monitoring and Control Data Definition Standard.

The remaining part of this section presents the concept of the Space System Model required to understand the changes that have been made to the SCOS system to be used as the kernel of the EMCS.

The Space System can be decomposed into a hierarchy of System Elements as for example:

HERSCHEL Space System
2→ HERSCHEL Spacecraft
3→ DHS
4→ Onboard Software
5→ Application Process 1
5→ …
2→ HERSCHEL Ground Segment
3→ EMCS
4→ SCOS
5→ …
4→ System SCOE
5→ SCOE TC Front End
5→ SCOE TC Controller
5→ …

Within the overall set of System Elements (SE), some participate actively to the monitoring and control of the Space System and need to be addressable. These SEs are the application processes of the spacecraft plus the ground processes like the TM/TC Front Ends, Ground station controllers, SCOE applications…

For these addressable SEs, a SE identifier, unique reference within the overall Space System is defined. This SE Id has an enumerated value taken within the range 0x0000 to 0xFFFF.

- Mapping the concept of Application Processes introduced in the PUS, for those SEs that represent onboard application processes, the SE Id correspond to the APID used to identify the destination of TC packets sent to the Spacecraft and the source of TM packets received from the Spacecraft. As such, the SE Ids values 0x0000 to 0x07FF are reserved for onboard application processes.
- The remaining values 0x0800 to 0xFFFF are available for identifying other SEs from the Space System.

Related to the commanding and monitoring function, Different types of SEs exist:

- SEs like onboard application processes, which are processing requests (i.e. commands) and/or generating reports (i.e. TM).
- SEs like the TC Front End, which carry requests/reports from one SE to another.

The EMCS makes use of the type of SE to determine:

- those SEs that can process requests (type = APID). Activities (like telecommands, procedures, stimuli) can be defined within the SSM and attached to the SE that will execute them.
- those SEs that are routing requests and reports to their final destination (type = Router). Activities cannot be attached to them.

Other types of SEs can be defined within the SSM, which the EMCS may use to perform specific processing.

The SCOS2000 system has been updated to control any SE that can process requests, independently of whether they are representing an onboard application process or a ground process.

The following table proposes a distribution of SE ids taking into account:

- the CCSDS constraint that 11 bits are used within the header of TM/TC packets to represent onboard application processes (i.e. 0 to 0x7FF),
- the current EMCS allocation of Ids for the existing EMCS SEs.

This table resides in the SSM.

SE Identifier			***SE Name***
Hexadecimal	***decimal***	***Type***	
Spacecraft Application Processes			
0x0000 to 0x07FF	*0 to 2047*	*Apid*	*Reserved for Spacecraft Applications Processes*
0x0800	2048	Router	Reserved (Spacecraft Identifier in MIB)
EMCS/SCOS2000 Tasks			
0x1000 to 0x1FFF	*4099 to 8191*	*Undef.*	*Reserved for SCOS2 Tasks*
EMCS Clients			
0x2000 to 0x2FFF	*8192 to 12287*	*Undef.*	*Reserved for EMCS Clients (e.g. TOPE)*
Ground Station Specific Application Processes			
0x4000 to 0x4FFF	*16384 to 20479*	*Undef.*	*Reserved for Ground Station application processes*
SCOE Application Processes			
0x8000 to 0x8FFF	*34816 to 36863*	*Undef*	*Reserved for "SCOE" related application processes.*
0x8000 to	*32768 to*	*Apid*	*Alternative routing of Spacecraft application processes via*

SE Identifier			*SE Name*
Hexadecimal	***decimal***	***Type***	
0x87FF	*34815*		*"EGSE_8000".*
0x8800 to 0x880F	*34816 to 34831*	*Undef.*	*Reserved for TC Front End.*
0x8800	34816	Router	Alternative routing of Spacecraft Application processes via TC Front End
0x8801	34817	Apid	TC Front End Controller
0x8810 to 0x881F	34832 to 34847	Undef.	Reserved for TM Front End
0x8810	34832	Apid	TM Front End > VC0
0x8811	34833	Apid	TM Front End > VC1
0x8812 to 0x8816	*34834 to* 34838	*Apid.*	*Reserved for VC2 to VC6*
0x8817	34839	Apid	TM Front End > VC7
0x8818	34840	Apid	TM Front End Controller

EGSE Gateway/Router Tasks

0xF000 to 0xFFFF	*61440 to 65535*	*Undef.*	*Reserved for EGSE Gateway/Router*

Within the SSM, activities and reports can be attached to any SE. Activities initiated from the Test and Operations procedure execution environment may:

- either be implemented as a procedure, and as such be executed by the procedure execution environment,
- or not be implemented as a procedure, and as such passed to the EMCS kernel for execution.

ESA has initiated the development activity for a SSM data repository based on the Oracle DBMS. This Oracle based SSM is not currently available. Users of the current EMCS have to populate the Mission data using the ASCII files interfaces provided by SCOS2000 and updated to support specific EGSE command and control mechanisms.

Within SCOS, a facility is available to load the ASCII files into the internal SCOS MIB database.

1.4.2 THE EMCS KERNEL

The EMCS kernel is an updated version of the SCOS2000 that supports functions required for use as an EGSE kernel including:

- Capability to route commands to the spacecraft using the ground station or via the SCOE.
- Capability to monitor and control ground processes (e.g. SCOE applications)
- Capability to initiate commands and to process TM that are not predefined within the database.

1.4.3 THE TEST AND OPERATION PROCEDURE ENVIRONMENT

The ECSS-E-70-32 Procedure Definition Language standard comprises the specification of a common procedure language for Test and Operations (PLUTO).

ESA has initiated the development activity of software packages supporting the PLUTO standard and offering end users the capabilities to prepare test and operation procedures, and to execute them in the context of SCOS 2000.

As the PLUTO compliant software is not yet available, a procedure environment based on the TCL-TK language has been prepared and delivered to the Herschel/Plank instruments team. This system, called TOPE, interface the EMCS kernel, initiating commands (TC or SCOE command), following their execution by the ground segment and the onboard application processes using reports sent back by SCOS and monitoring the state of the space system using TM and ground parameters.

TCL is a simple (free) programming language, made up of commands with parameters. TCL provides usual programming constructs such as:

- Variables
- Control structures
- String manipulation.
- I/O, including files on disk, network sockets, and devices such as serial ports.
- simple facilities for socket communication over the Internet.
- File management: reading and writing file attributes, copying, deleting, creating directories, etc.

  Sub-process invocation: you can run other applications with the exec command and communicate with them while they run.

  Lists: TCL makes it easy to create collections of values (lists) and manipulate them in a variety of ways.

  Time and date manipulation.

  Events: TCL allows scripts to wait for certain events to occur, such as an elapsed time or the availability of input data

  Extension mechanism to add features (e.g. get parameter, send command, retrieve database info. etc).

  TK extension to add graphical user interfaces.

The TOPE system user manual is given in section 6. For information about TCL/TK, see http://www.scriptics.com

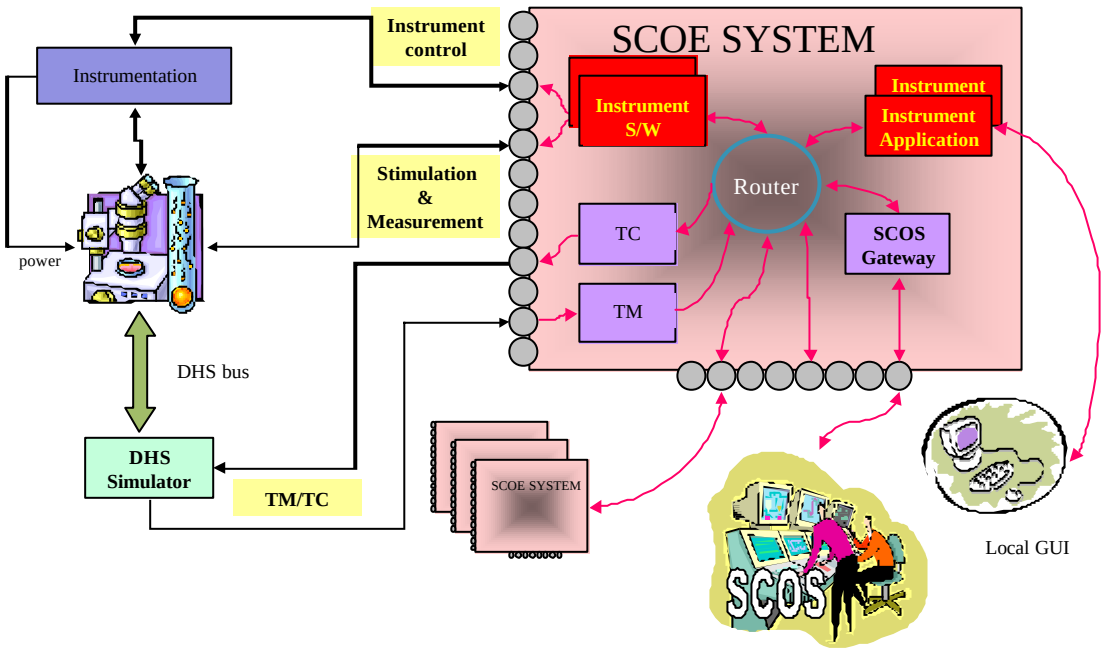
1.4.4 THE SCOE

The SCOE system complements the EMCS kernel. The EMCS Kernel (SCOS) interfaces the SCOE using TCP/IP.

For PROBA, a generic facility (called EGSE Router) has been developed by ESTEC/TOS-EMG to allow an easy integration and remote control of SCOE equipment. This facility has been developed for MS-Windows platforms. In the remaining of this section, an overview of this EGSE Router is made. The TM/TC Front Ends are also ESA developments.

Note:

1. The MS-Windows based EGSE Router and the TM/TC Front Ends are not used within the Herschel/Planck project.
2. A LINUX based EGSE Router has been developed by SRON (NL) institute for Herschel.
3. The SCOS to LINUX based EGSE Router interface is compliant to the ESTEC setup.
4. The LINUX based EGSE Router to SCOE interface does not comply to the ESTEC setup.



1.4.4.1 The MS-Windows EGSE Router

A main facility residing on the SCOE, called Router, provides a core facility for building of SCOE/EGSE equipment in which the system builder can:

- incorporate specialised hardware interfaces to the space equipment
- incorporate software applications that communicate with the hardware, with each other and with the outside world.

The Router provides:

- location transparency
- client management
- Multiple clients per connection
- Data transparency
- Message delivery via call back procedure (asynchronous) or polling (synchronous).

The Router is written in Delphi. It uses Microsoft Distributed COM, acts as an ActiveX server and supports multiple languages (e.g. Borland Delphi, C++, LabView, Java) i.e. any language supporting ActiveX. For those systems that cannot interface the Router using DCOM, another facility, called Gateway, is provided that allows connectivity using TCP/IP networks.

The EMCS Kernel (SCOS) currently makes use of the Gateway to connect to the Router environment.

1.4.4.2 The MS-Windows TM Front End

The TM Front End provides functions necessary to extract, archive and distribute (CCSDS) telemetry packets from a digital data stream connected to:

- hardware that interfaces to on-board equipment directly (as in the case of an on board data handling interface simulator)
- operational TM/TC links via base-band equipment.

Appendix A contains a definition of the Front End commands.

1.4.4.3 The MS-Windows TC Front End

The TC Front End provides transformation of requests for command packet transmission into a digital (CCSDS) data stream.

The complete ground-space protocol uses an embedded COTS product: TC ENCODER SHELL from De lande long Consultants. This product has to be purchased by users wishing to utilise the TC front end.

2 INSTALLING THE SCOS EMCS OPERATIONAL ACCOUNT.

The procedure given in this section is valid for the "**SCOS EMCS release 2.3e P5 version**" delivered by ESTEC TOS-EMG.

The SCOS EMCS releases 2.3e P"5 version has been prepared for Herschel/Planck instrument users.

The following procedure allows installing the SCOS operational environment in the standalone mode. For installing SCOS in the client/server mode, the official SCOS configuration and installation manual shall be used.

2.1 Pre-requisites

- Linux SUSE 7.3 is installed on your machine. Refer to Appendix D for installing SUSE 7.3.
- The COTS are already installed. If not, follow the SCOS2000 Configuration & Installation Guide procedure (refer to [R8]).
- The C-Tree is not anymore delivered on R2.3e COTS CDROM. Users need a license. Discussion on-going with ESOC.
- The ILOG licensing discussions are still on going between ESOC and ILOG. Temporary licenses can be obtained from ESOC waiting for the output of on-going discussion.

2.2 Installation Procedure

Login as **root**

If it is the first time you install the SCOS R2.3e, create user (using Yast2) sops23e with /home/sops23e (tcsh shell).

If you wish to install the SCOS2.3e P5 in parallel with previous P3 (or P4) version, create a directory called /home/sops23e/P3 (or P4), move all /home/sops23e into this new delivery (except the dot files and the Kdesktop directories. The P5 version is untar into a directoy P5_sops23e. Moving from one version to another can simply be achieved by moving all the content of this P3/P4 or P5 directory within the top /home/sops23e parent directory.

Another approach is to use the <automount> linux facility.

If a previous version of SCOS has already being executed on the machine, you may need to remove all SCOS related /tmp files:

```
rm /tmp/*
```

Login as **sops23e**

```
cd /home/sops23e
```

Copy the content of the EGSE 2.3eP5 CDROM into the /home/sops23e

The bin and lib directories contain binaries and libraries for LINUX. To run on SUN, you need to install the SCOS development environment and recompile (refer to clause 3

```
tar xvfz P5_sops23e.gz
rm P5_sops23e.gz
```

The tar extracts the P5 data in the P5_sops23e directory. If you are installing on top of another SCOS version, move your previous version to a safe area prior to move the full content of the P5_sops23e into /home/sops23e.

```
<don't forget to move previous version if existing>
cd P5_sops23e
mv * /home/sops23e
```

verify link CORBAcfg/omni. The default setup is pointing to /opt/omni.

verify link ILOG/views31 Replace it to point to your ilog views directory.

e.g.: rm ILOG/views31

ln -s /opt/ilog3.1/views311 ILOG/views31

Configure SCOS for your machine as follows:

```
cd /home/sops23e/Installation
```

On Linux, update the rhosts file:

```
vi rhosts
```

On Linux, however the file contains the + sign, you need to explicitly add your machine name.

Copy the cshrc file in your home directory:

```
cp cshrc ../.cshrc
```

Update the s2k.hosts:

```
vi s2k.hosts
```

1. Update all occurrences of primehostname by your machine name
2. Update all occurrences of primehostaddress by the related IP address.

Update the session.dat file. Add your machine.

```
vi session.dat
```

Update the cmd_host_file. Add your machine. You may also remove all others machines references.

```
vi CMD_HOST_FILE
```

Create a link into the admin directory for your machine, as follows :

```
cd /home/sops23e/admin
ln -s s2k.env.standalone s2k.env.<machine name>
```

The 2.3eP5 delivery is configured for H/P. For other than H/P users, don't forget to recompile the egse-ws/EGSE subsystem with the mission needs (refer to [R6]) and to install related binaries and library into /home/sops23e.

Refer to section 2.3 for known problems with the 2.3e P5 delivery.

The SCOS EMCS 2.3e P5 system should be ready for use,

logout and login again as sops23e.

In case any problem occurs when starting the SCOS tasks (e.g. NAME, PDSEV, MISC, Releaser), verify that the setup is complete using the table 5 of the section 2.2.1 of the SCOS2000 Configuration & Installation Guide, S2K-MCS-SUM-0002-TOS-GCI 3.2 rev. 2.

Prior starting using SCOS, don't forget:

- to configure the MISCconfig file (refer to clause 4.2) and
- to update the MISCconfig satellite_name, spacecraft_id.
- to import the MIB ASCII data (refer to clause 4.1).

2.3 Known problems with the SCOS EGSE 2.3 P5 version.

This list of problems is not an exhaustive list.

2.3.1 ERRORS SHOWN IN SCOS DESKTOP FOOTER WINDOW

The Desktop footer window is the window that appears on the bottom of your screen when you start the SCOS DESK task. This window shows the last few messages (information, warning or alarms).

2.3.1.1 CORBA::SystemException

When starting the FARC and DDSS tasks, the following alarm is raised:

Caught CORBA::SystemException trying to reach Name Service

The error badly states that the *redundant* MISC server cannot be reached. This message is not adequate for standalone SCOS configuration (as the default EGSE configuration proposed in this document). A software correction shall be made to remove the raising of this message. There is no further impact to expect on the SCOS system due to this error message.

2.3.1.2 Connecting NCTRS

When the EGSE is configured to use NCTRS (i.e. MISCconfig variable EGSE_MODE = 0 or 1) and NCTRS is not running, the following alarm is raised:

Could not connect to: NCTRS_A on host …

To start the NCTRS simulator, starts the TCSIM, ADMSIM CLCWSIM.

2.3.1.3 MISCconfig variables

Some MISCconfig variables refer to environmental variables (e.g. $scosii_homedir) to define directory paths. Others make use of absolute path (e.g. /home/sops23e/…)

In case a need appears to update MISCconfig variables, it is not recommended to replace absolute paths by relative paths using environmental variables. This is due to some SCOS subsystems that do no support expansions of environment variables.

3 INSTALLING THE SCOS EMCS DEVELOPMENT WORKSPACES.

This section complements the SCOS-2000 Configuration and Installation Guide (refer to [R8]). It describes the steps to be performed to install the "**SCOS EMCS release 2.3e P5 version**" delivered by ESTEC TOS-EMG.

3.1 Pre-requisites

- Obtain from ESA/ESOC, the SCOS2000 development license.
- Receive from ESTEC TOS-EMG, the SCOS EMCS release 2.3e P3 development CDROM.
- Prepare the SCOS EMCS development account following instructions given in [R8].

3.2 Installing the SCOS EMCS release 2.3e P5 sources.

- P5_linlib.gz

  This file contains all development workspaces.

It should be noted that within the rel-ws of this 2.3e P5 version, the EGSE subsystem is obsolete. The 2.3e P5 EGSE subsystem is located within the egse-ws.

As said above, the 2.3e P5 version of SCOS EMCS is prepared for H/P. In order to configure SCOS for other missions, the EGSE subsystem needs to be configured and recompiled (refer to [R6]).

4 CONFIGURING THE SCOS EMCS.

4.1 The Mission Information

4.1.1 THE MINIMUM EGSE DATA SET

The SCOS EGSE system requires a number of generic MIB data definitions to properly function. These data can be found in the directory:

/home/sops23e/data/MIN_EGSE_v2

A given mission needs to adapt the MIN_EGSE for its mission. Especially the following files shall be updated to comply with the TM/TC packet header structures:

tcp.dat	make a new entry for the mission specific TC packet header. The STANDARD entry can be removed.
pcdf.dat	make new entries for the mission specific TC packet header parameters. The STANDARD entries can be removed.
pcpc.dat	possibly add new entries for the mission specific TC packet header parameters. Remove non-used entries.
ccf.dat	update the ZLOADTT command to make use of the mission specific TC packet
cdf.dat	update the ZBINTC command parameters to comply with the mission TC packet header
cpc.dat	Update the parameter definitions related to the ZBINTC command to comply with the mission TC packet header
dpc.dat	update the ZREJTM and ZVERIFT6 definitions to comply to the mission TM packet header.
pcf.dat	update the ZTMxxxxx and ZV6xxxxx definitions to comply to the mission TM packet header.
plf.dat	update the ZTMxxxxx and ZV6xxxxx definitions to comply to the mission TM packet header.
vdf.dat	Update the version entry.

All parameter and command names given in the MIN_EGSE ASCII files start with the letter "Z". This convention allows an easy identification of the minimum data definition. In case a mission

requires changing all these identifiers, some of these identifiers are referred in the resources/MISCconfig file and need to be changed accordingly.

Furthermore, the SCOS generic system makes use of a number of predefined TM packets. These packets are identified by their SPIDs.

The SPIDs 1 to 100 have been reserved for the SCOS EGSE generic setup. A number of these reserved SPIDs are already in use and can be found in the HFAconfig, TMDcacheSetUp, ASCII/tpcf.dat files contained within the MIN_EGSE directory. These SPID definitions are also referred in the resources/MISCconfig file.

If needed, a mission may assign other SPIDs than the reserved ones to the required generic TM packets by updating consistently the MIN_EGSE definitions and the MISCconfig ones.

4.1.2 THE MERGEDATA FACILITY

When generating its own data, it is recommended to keep separated those data being part of the MIN_EGSE from those data that are mission specific (e.g. data related to a given instrument).

The **data/mergeDATA** facility allows merging several sets of data files (e.g. MIN_EGSE and Instrument1) into a new set of data files. For this purpose, a mission data directory (e.g. Instrument1) shall comply with the structure of the MIN_EGSE directory and be located within the $scosii_homedir/data directory.

The output of the mergeDATA facility is a new directory that can be directly used by SCOS or subject to another merge with for example another instrument data set.

> **Attention**: It shall be acknowledged that using the mergeDATA facility, TOPE procedures that are defined within the two original sets of data files are **copied** into the new "merged" set of data files. When updating TOPE procedures, only the new "merged" data procedures are updated, resulting in a **need to manually copy** the new TOPE procedures that have been modified into their original data set if the need appears to use the new version of the TOPE procedures.

4.1.3 THE SETSCOSDATA FACILITY

The **data/setSCOSdata** facility allows setting SCOS for using a given MIB database defined within the data directory. This facility requires as argument the name of the new MIB directory to use and:

- sets the data/ASCII link to point to this MIB directory,
- sets the TOPE procedure directory (i.e. /home/sops23e/TCL/TC) to point to the MIB TC directory that should contain all TOPE procedures.
- copies the HFAconfig and TMDcacheSetUp into the hfiles and TMD directories.
- imports all MIB ASCII files within SCOS (i.e. performing automatically the IMPORT task of the SCOS task launcher)

When using the setSCOSdata consistently, no data is lost and switching from one database to another one is possible.

> **Attention: Prior to switch databases**, it shall be noted that new MIB definitions might conflicts previously ones resulting in erroneous SCOS archives. If this is the case, it is recommended to archive the hfiles directory and remove all archives contained within the hfiles directory prior to use the setSCOSdata facility.

In case of errors or warnings when importing the data, a log file ($scosii_homedir/Import.log) can be analysed.

4.2 MISCconfig EGSE specific variables

The following variables are those EGSE specific ones. Some of these variables need to be changed to comply with a specific test configuration.

Name	Description	Default value
EGSE_MODE	Configure the SCOS/EGSE to make use or not of ESOC ground station. 0: TC Releaser only connects to the NCTRS. 1: TC Releaser connects both to the NCTRS and to the EGSE Router 2: TC Releaser only connects to the EGSE Router.	**1**
NCTRS_EGSE_TC_SERVER	If EGSE_Mode = 1 or 2, Host name of the SCOS gateway (i.e. the SCOS application that connects using TCP/IP the EGSE gateway of the EGSE Router)	localhost
NCTRS_EGSE_TC_SERVER_PORT	Port number of the SCOS gateway (for connection to the TC Releaser)	2222
EGSE_EGW_SERVER	Host name of the EGSE gateway of the EGSE Router.	**localhost**
EGSE_EGW_SERVER_PORT	Port number of the EGSE gateway.	**9876**
EGSE_BINCMD_NAME	Name of the "binary command" that can be used with caution to upload commands that are not defined within the MIB. This command is defined within the EGSE minimum MIB. **When such command is not authorised (e.g. for a given test configuration), the command shall be removed from the MIB before importing within SCOS.**	ZBINTC
EGSE_VER_SCOE_PKTID	SPID of the SCOE Command verification report	22
EGSE_VER_SCOE_APPDATA_OFFSET	Byte offset of the Token (APID/SSC) in the SCOE command verification reports (Template 2 and Template 5)	22
EGSE_SEND_CHKSUM_ERROR_PKT	Flag identifying whether TM packets that fail the checksum verification are archived	**1**

Name	Description	Default value
	(= 1) or not (= 0). Refer to section 7.9.2.2.	
EGSE_STREAM_ID_FOR_CRC_ERROR	Only valid when EGSE_SEND_CHKSUM_ERROR_PKT = 1, Data stream containing the TM packets that failed the checksum verification. Refer to section 7.9.2.2.	**65534**
EGSE_TM_FILING_TIME_TYPE	Incoming TM data Time stamping strategy: 1: EMCS time, 2: Ground Station/ SCOE time, 3: OBT time (without any correlation, meaning the OBT field value shall be relative to the mission epoch, see TCO_MISSION_EPOCH MISCconfig). Refer to section 7.9.2.1	**1**
EGSE_TM_OBT_THRESHOLD	Incoming TM OBT sanity check threshold. If the delta between the OBT in the packet and the current time is more than the threshold it is ignored and the current time is used. In the TM Packet history the packet is shown as type PR, rather than PG. A warning is given for every “insane” packet. Refer to section 7.9.2.1	**60**
EGSE_ VER_TC_GS_PKTID	System SPID to use to store ground station request verification reports. Refer to section 7.9.2.3.	23
EGSE_VER_TC_REP_PKTID	System SPID to use to store Service 1 TC verification contained within TM packet reports. Refer to section 7.9.2.3.	24
EGSE_VER_TC_REP_APPDATA_OFFSET	Byte offset of the request token of Service 1 TC verification contained within TM packet reports.	16
EGSE_DUPLICATE_GS_UL	Flag identifying whether a UL acknowledgement flag is to be simulated when receiving UL report from the TC Front End (refer to BD mode). 0: do not generate a pseudo report. 1: For each GS_UL ack report received from the EGSE, inject a GS_UL and a GS_OB ack into SCOS.	1
EGSE_WRAPPER_PKTID_3	System SPID value used by SCOS to file unknown SCOE observation messages. Refer to section 7.9.2.4.	20
EGSE_WRAPPER_PKTID_6	System SPID value used by SCOS to file unknown TM packets. Refer to section 7.9.2.5.	21
EGSE_PIC_FILE_3	Location and name of the Template 3 specific MIB PIC file	$SCOSII_ASCII_MIB/ pic3.dat

Name	Description	Default value
EGSE_PIC_FILE_3	Location and name of the Template 3 specific MIB PID file	$SCOSII_ASCII_MIB/ pid3.dat
EGSE_PIC_FILE_6	Location and name of the Template 6 specific MIB PIC file.	$SCOSII_ASCII_MIB/ pic.dat
EGSE_PIC_FILE_6	Location and name of the Template 6 specific MIB PID file	$SCOSII_ASCII_MIB/ pid.dat
IMPT_DST_FILE	Name of the ASCII file containing the System Element routing configuration data. Refer to section 6.1, VIA argument.	dst.dat
OOL_PACKET_APID	System SPID to use to store OOL event packets (default value is 3000). Note that this value might be changed to comply with H/P SPID allocation strategy (TBC)	3000
PDS_SERVER_TC	PDS server for TC packets. Default is localhost	localhost
PDS_SERVER_EV	PDS server for EV packets. Default is localhost	localhost
HPR_TC_DIR	Herschel packet retrieval configuration	/home/sops23e/HPR
HPR_OL_DIR	Herschel packet retrieval configuration	/home/sops23e/HPR
HPR_IPC_PORT	Herschel packet retrieval configuration	2209
TCO_MISSION_EPOCH	Mission Epoch (default is 1/1/1970)	1958-01-01T00:00:00.000000

4.3 Environment variables for the TCL scripts of TOPE

The following variables do not need to be updated. If really needed for a given mission setup, the EXIF_runtime.env and s2k.env file shall be revised.

Variable	Description	Default Value
EXIF_ML_DIR	Directory where ML looks for its components.	Local directory
EXIF_TOPE_EDITOR	Path and filename of the editor which is used by TOPE	./ML.TCL
EXIF_POLL_TIME	Interval between two polls for Telemetry data in milliseconds	100
TOPE_DIR	Directory where TOPE looks for Testcase scripts	../TC
TOPE_LOG	Directory where TOPE creates Log files	$TOPE_DIR

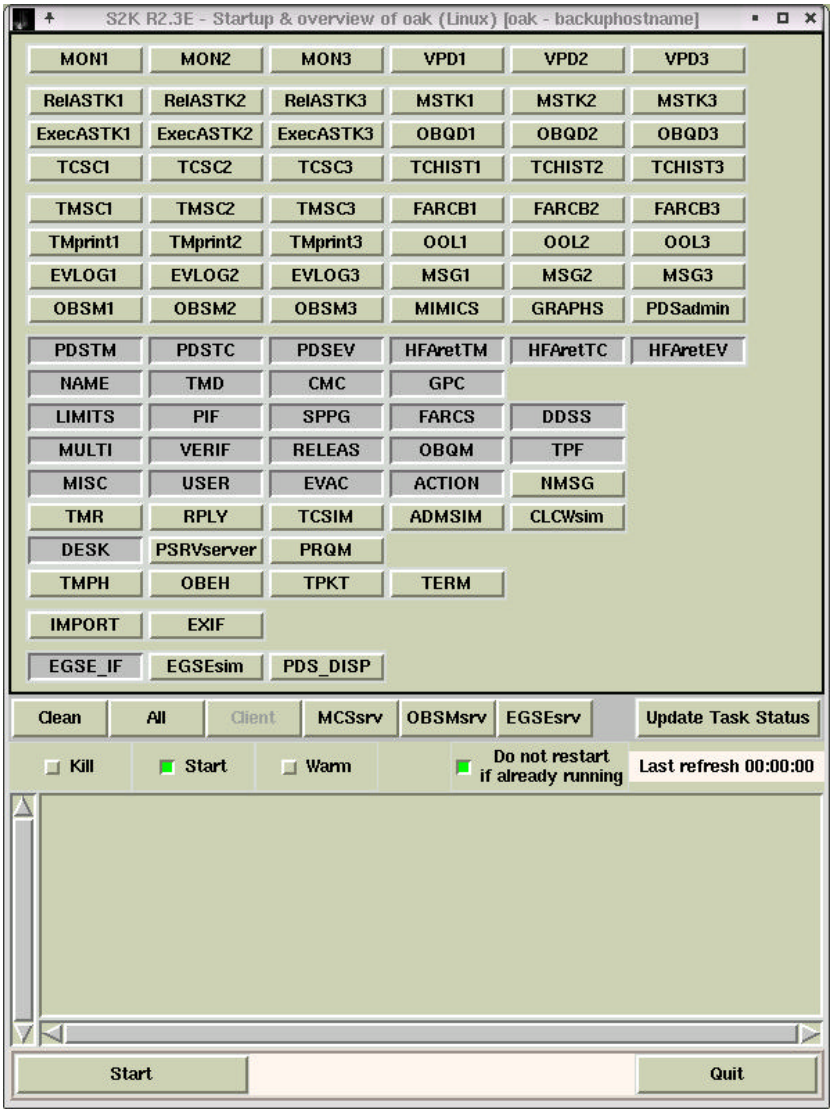
5 STARTING THE EMCS

Prior to using the EMCS, the system has to be configured, mission data has to be prepared and the corresponding MIB ASCII files be imported (refer to section 0).

The MISCconfig file has to be updated to reflect the mission computer setup (EGSE Router address, Printers, etc.)

5.1 Starting the SCOS EGSE tasks

To start the SCOS system, run the s2.start facility. This facility starts the SCOS tasks launcher (see below). Select the EGSRsrv button and start the associated tasks.



5.2 *Starting the External Interfaces and TOPE procedure environment*

To start the external interfaces and TOPE task launcher, select the **EXIF** button (from the SCOS task launcher) and start the task.

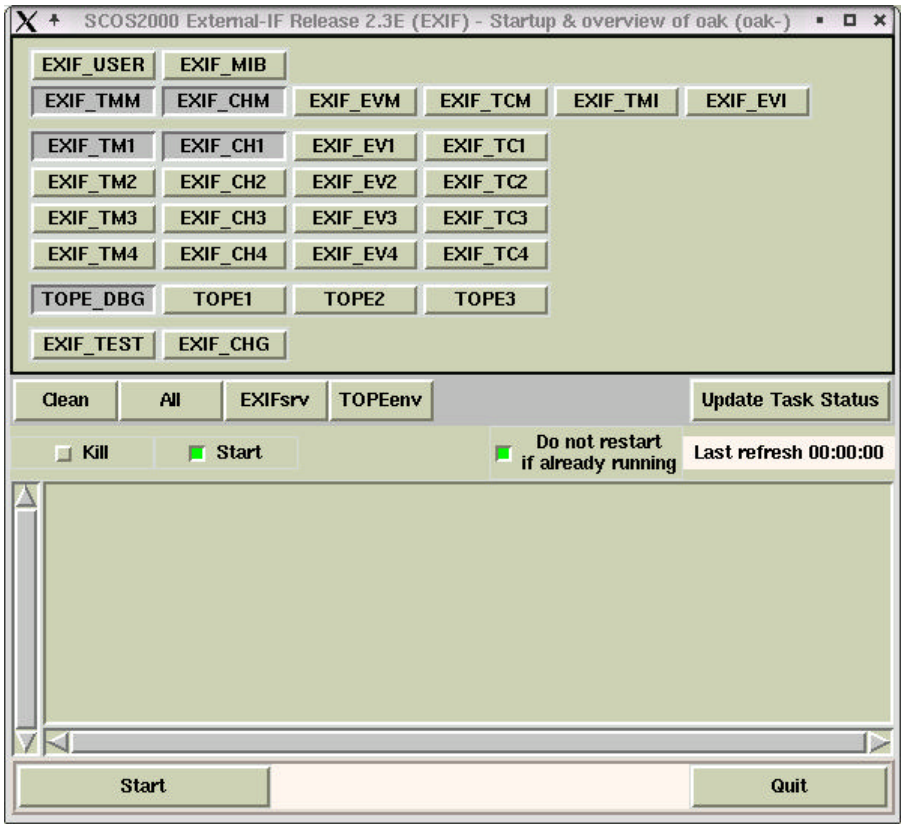
To start the TOPE debugger procedure environment, select the **TOPEenv** button and start the task.

NOTE:

*In case other TOPE clients are required, for each new client, you need to start a new **EXIF_Chi** task.*

*The **TOPEi** buttons allow starting instances of the TOPE execution environment in normal mode;*

The TOPE_DBG button allows starting the TUBA TCL debugger.



6 TOPE LANGUAGE DEFINITION

The TOPE system is based on TCL/TK. The SCOS interfacing functions have been developed on top of the TCL/TK. The TOPE language offers to the end users, the capability to define their test procedures.

Users may make use of any TCL/TK low-level functions.

The section only introduces the SCOS related functions of the TOPE language.

It shall be noted that the language constructs presented in this section are not compatible with the TOPE language used prior to Issue 3.0 of this document.

6.1 Sending Commands to the Unit Under Test and/or SCOEs.

tcsend <CommandName> <Arguments> <Parameters>

<CommandName>		
<Arguments>		all arguments clauses are optional. If not given, the EMCS kernel default value is used.
	via <string> **This option is not supported in TOPE/CCS**	Encoding and Routing information. Up to 3 values separated by "." can be given as following: **Server name**: EGSE or NCTRS **EGSE encoding style:** TC or SCOE (has to be consistent with the MIB definition of the command header) **EGSE Destination:** a value from 0x0000 to 0xFFFF. If no value is given, the Default for the Via is taken from the MIB definition (DST table) Note that if the Server name is NCTRS, the Via 2nd and 3rd values have no meaning. Note that if the EGSE encoding style is SCOE, the EGSE destination value is optional; the EGSE destination is retrieved from MIB by SCOS (the field APID of the command is used). If no value is given, the command will be sent using the MIB information related to the APID of the command (refer to DST table). Note that for a TC routed to the TCFE, the coding style has to be TC and the EGSE Destination has to be set to the ID of the TCFE: EGSE.TC.0x8800
	mapid <number> **This option is not supported in TOPE/CCS**	If this argument is not given, the default value calculated by SCOS-2000 is used (MISCconfig variable CMD_MAP_ID, CCF_MAPID for high-priority commands). The MAPID value is relevant only in case of a command that is sent to NCTRS or EGSE using the TC encoding style (Template 4).

	vcid <number> **This option is not supported in TOPE/CCS**	Virtual Channel ID. If this argument is not given, SCOS-2000 will use its default value (MISCconfig variable CMD_VC_ID). VCID makes sense only in case of a command which is sent to NCTRS or EGSE using TC encoding style (Template 4).

	checks {
	<staticPTVflag>
	<dynamicPTVflag>
	<CEVflag>
	}
		Flag to indicate if the PTV and or the CEV checks are required. Allowed values are:
		Static PTV Flags: SPTV / SPTV_OFF
		Dynamic PTC Flags: DPTV / DPTV_OFF
		CEV Flags: CEV / CEV_OFF
		ALL (default value) meaning {SPTV DPTV CEV}
		NONE meaning {SPTV_OFF DPTV_OFF CEV_OFF}

	ack {<ACKflags>}	List of flags for reporting command execution verification by application. Allowed values are: ACCEPT START PROGRESS COMPLETE ALL meaning {ACCEPT START PROGRESS COMPLETE} NONE meaning no acknowledgement reports. If the Ack clause is not present, the default acknowledgement flags defined in the MIB (CCF_ACK and PCDF_VALUE where PCDF_TYPE="K") are used.
	referby <Varname>	Name of the TCL variable, which is used for acknowledgement report notification. Whenever an acknowledgment report arrives, this variable is assigned to a TCL list structure. The attributes of this list are extracted with a family of access functions, which are described in 6.1.1. If the variable <Varname> is first used within a tcsend statement. it is initialised with an empty TCL list when the tcsend is invoked.
	userrequestid <UserreqID>	End user identifier of the TC request. If the userrequestid is not used, the default value is 0.
	releasetime <*time*>	The absolute time when the command should be released in the form YYYY.DDD.HH.MM.SS.UUUUUU If the releasetime clause is not present, the command is released by the SCOS system ASAP, where the Command Injection Handler defines the release time.
	executiontime <time>	When this clause is added to the sendTC, an onboard schedule command is build (i.e. LOADTT, PUS service 11,4). The TC to load within the onboard schedule is associated the <time> given as a parameter corresponding to the required start of execution of the related TC by the onboard schedule. If the executiontime clause is not present, the command is not time-tagged.

	ilocktype<interlockType> **This option is not supported in TOPE/CCS**	The interlock type of the command. Allowed values are: NONE (default value) LOCAL GLOBAL SUBSYS_LOCAL SUBSYS_GLOBAL
	ilockstage <stage> **This option is not supported in TOPE/CCS**	Stage for the interlock. Allowed values are: UV_GS_ACCEPT (default value) UV_GS_UPLINK UV_SC_ONB_ACCEPT EV_ACCEPT EV_COMPLETE
	nowait	By default, tcsend waits automatically until the stage UV_GS_RECEIVE is reached. If an error occurs in any of these stages, the tcsend command fails with a TCL exception. Using the nowait option allows disabling the automatic wait. The use of the nowait option allows reproducing the behaviour of previous to P4 version of the TOPE system.

Parameters	The command parameters are listed at the end of the command with a list for each parameter: Each command parameter has the following form: { <name> <value> [<format>] [RAW | ENG | DEFAULT] } <name> Parameter name <value> Parameter value <format> Format of the value The default <format> is **ST** (string). The string format can be used to pass any parameter to SCOS. The SCOS system automatically translates strings into the expected format and generates an error in case of incompatibility (refer to [R9]) The following list of formats is given for completeness with EXIF interfaces. Using specific EXIF types implies type compatibility check performed at TOPE level. LO long (32 bits signed) UL unsigned long FL float IEEE standard single precision floating point BO boolean ST string BS binary string TI time (refer to section 6.1.1) *SH short (16 bits signed integer)* *US unsigned short* *CH char (length 1 char only)* *OC octet (length 1 octet only)* Values of parameters defined as Binary String within the MIB can be passed according to the following convention: 1. {BINPARAM "xyz" "BS"} The binary string consists of 3 bytes containing the ASCII character codes of the x, y, and z characters). 2. {BINPARAM "00FF10" "ST"} The string "00FF10" is treated as a sequence of 2-digit hexadecimal numbers. 3 bytes with the values 0, 255, 16 will be encoded in the command. RAW/ENG ENGineering Parameter is defined as RAW (the default) or value. DEFAULT Use MIB default for this parameter. If this option is given, a dummy value (preferably "" (i.e. null)) must be specified, and the <format> must be omitted. Example usage: `tcsend TC1 {PARAM1 "" DEFAULT}` This syntax can be used to make the usage of the MIB default explicit, and resolve ambiguities if the command has several parameters with the same name.
Return value:	On success, the return value of tcsend is a unique id of the request, which is assigned by the system.

On failure	On failure, a TCL exception is thrown (which may be catched using the standard TCL command `catch` if desired). E.g.:
	`if {[catch {tcsend TC001 referby tc1}]} {`
	`  putlog "tcsend has failed"`
	`}`

When a command is initiated by TOPE, the command with all its arguments and parameters is traced within the log.

When the command is received by SCOS, a first message is immediately returned and logged as following:

<date&time> TC accepted by SCOS.

or

<date&time> TC `rejected` by SCOS : Error code <*SCOS returned code* >

In case of error, the string message passed in the exception is also logged.

Other command execution reports may arrive from SCOS. Each report is logged using the following syntax:

```
<date&time> TC <Varname> <requestId> multiplexId <multiplexId> - <stage>
<status> [completed] <updateTime>
```

6.1.1 TIME FORMAT PARAMETER VALUES.

Parameter values of type **Absolute time** shall comply with the following format:

"YYYY.DDD.HH.MM.SS.mmmmmm".

Parameter values of type **Relative** *time* shall comply with the following format:

"[+|-]HH.MM.SS.mmmmmm".

The relative time format may be abbreviated as follows:

- The leading "HH" and "MM" fields may be omitted. Thus "MM.SS.mmmmmm" and "SS.mmmmmm" are allowed.
- If the "HH" and "MM" fields have been omitted, the microseconds field may also be omitted. A plain number (containing no ".") is parsed as a number of seconds.
- Trailing zeros in the microseconds field may be omitted. Thus "+10.20.30.1", "+10.20.30.100", and "+10.20.30.100000" are the same (10 hours, 20 minutes, 30 seconds, and 100 milliseconds).

Note: Time values are handled as strings in TCL. As such, they cannot be used within arithmetic expressions.

6.1.2 ACKNOWLEDGEMENT REPORT ACCESS FUNCTIONS

The variable specified in the "referby" clause of tcsend will be updated every time an acknowledgement report is received from the EMCS.

The value of the updated variable is a TCL list containing the various properties of the report, which can be accessed using the following functions:

getrequestid <report>	Request Id (number) for identifying the command request, which matches the value returned by tcsend.
getmulitplexerid <report> **This function is not supported in TOPE/CCS**	Unique Id (number), generated by the multiplexer, set in first call of acknowledgement callback
getstage <report>	Stage of the verification. Updated in every call of the acknowledgement callback. The possible values are: PTV_DYNAMIC PTV_STATIC MCS_RELEASE UV_GS_RECEIVE UV_GS_UPLINK UV_ONB_ACCEPT EV_APP_ACCEPT EV_START_EXEC EV_PROGRESS_0 EV_PROGRESS_1 EV_PROGRESS_2 EV_PROGRESS_3 EV_PROGRESS_4 EV_PROGRESS_5 EV_PROGRESS_6 EV_PROGRESS_7 EV_PROGRESS_8 EV_PROGRESS_9 EV_END_EXEC
getstatus <report>	Status of the verification. Possible values are: IDLE PENDING PASSED FAILED UNVERIFIED UNKNOWN TIMEOUT SUPERSEDED UNCERTAIN_FAILED UNCERTAIN_PASSED AFFECTED SCC NOT_APPLICABLE

getstagehistory <report>	This is a character array (string) where the index is the value of verification stage (getstage). Executing the tcsend statement, all stages are set to "-", so the whole array is "----------". At each call of the acknowledgement call-back the corresponding array element in the stagehistory is set with a shorthand character for the status: IDLE I PENDING P PASSED S FAILED F UNVERIFIED U UNKNOWN X TIMEOUT T SUPERSEDED E UNCERTAIN_FAILED N UNCERTAIN_PASSED V AFFECTED A SCC "" (i.e. null value) NOT_APPLICABLE ? getstage returns at any time the current status of all of the verification stages, in a representation, which matches the TC history display of SCOS. Example: tcsend cmd1 referby vCmd1 waitfor vCmd1 –until {[getcompleted $vCmd1]} putlog [getstagehistory $vCmd1]
getcompleted <report>	Flag for indicating if the command has been completed (1) or not (0). This means that no more reports will be sent from SCOS. If <report> is empty, then 0 is returned.
getupdatetime <report>	Time of the report.
getuserrequestid <report>	Returns the user-defined request id, which has been specified with the "userrequestid" option of tcsend.
tostring <report>	Returns a printable summary of the report. Example usage: putlog [tostring $myRequest]

6.1.3 GET/SET THE TRANSFER MODE (AD/BD)

Note: *The H/P CCS system only operates in BD mode. The get/set transfer mode functions are not supported by the TOPE/CCS.*

The default transfer mode used by the TOPE/EXIF system is given by the MISCconfig variable:

CMD_PKT_UPLN_MODE.

The following settransfermode and gettransfermode commands allow changing this MISCconfig default behavior.

settransfermode <mode>

<mode>	Allowed values are “AD” and “BD”.

The success of this operation can be verified using gettransfermode (see below).

gettransfermode

Returns the current transfer mode of the external command handler. Possible return values are “AD” and “BD”.

6.2 Fetch a parameter

fetch <ParameterName>

<ParameterName>	Telemetry Parameter Name
Return value	TCL list containing the properties of the parameter. These can be extracted using the access functions (see below). On failure, a TCL exception is raised. e.g. set vP001 [fetch P001]

A set of access functions is provided for extracting the parameter’s attributes:

Function name	Return value
getname <parval>	Returns the name of the parameter or null if <parval> is empty.
getrawvalue <parval>	Returns the raw (i.e. uncalibrated) value of the parameter or null if <parval> is empty. If the rawvalue is invalid (function israwvaluevalid returns false and getrawvalidity returns non-zero), “” (i.e. null) is returned.
getextractedvalue <parval>	Returns the raw (i.e. uncalibrated) value of the parameter or null if <parval> is empty. Unlike getrawvalue, this function returns the raw value independent of its validity.
israwvaluevalid <parval>	Returns 1 (logical true) it the raw value of the parameter is valid. It returns 0 (logical false) if the parameter is invalid or <parval> is empty.
getrawvalidity <parval>	Returns the validity of the raw value of the parameter. This is a bitset represented by an integer number. Any non-zero value indicates that the value is not valid. Predefined constants are provided reflecting the meaning of each individual bit: **VAL_STATE_OFF** State validity evaluates to 0 **VAL_POWER_OFF** System element is not powered **VAL_ROUTE_OFF** Transport device route is off **VAL_MISC** Spare (not used) **VAL_TRANSIENT** Telecommand scheduled for param **VAL_TRANSPORT_ERROR_1..** **VAL_TRANSPORT_ERROR_8** Transport error **VAL_EXPIRED** Value has expired(packet not arrived) **VAL_UNKNOWN_STATE** State parameter is invalid

Function name	Return value

	VAL_UNKNOWN_COMMAND Unknown condition
	VAL_UNKNOWN_CRITERIA Unknown applicability criteria
	VAL_CALIBRATION Calibration failed
	VAL_TOO_EARLY Time predates mission
	VAL_UNKNOWN_PKT Packet not configured in the cache
	VAL_UNINIT Value has not been initialized
	VAL_PKT_RETRV Packet retrieval error
	VAL_MIB_ERROR Error in MIB (SDB) data
	VAL_SYSTEM_ERROR General system error
	VAL_FIELD_UNKNOWN Field unknown in packet definition
	VAL_FIELD_ABSENT Field absent from given packet
	VAL_UNKNOWN_TYPE Unknown type name
	VAL_UNKNOWN_OP Unknown operator
	VAL_UNKNOWN_CONV Unknown conversion
	VAL_OVERFLOW An overflow occurred - data may have been lost
	VAL_DIVIDE_BY_ZERO Divide by zero
	VAL_OL_PARSE An error occurred parsing OL expression.
	
	Example usage:
	`set g123 [fetch G123]`
	`if {[getrawvalidity $g123] & $VAL_TOO_EARLY}…`
	
	Notes:
	1) This function and the VAL_xx constants are intended for in-depth investigations. The israwvaluevalid function can be used for general purpose.
	2) The “validity” panel of SCOS’ TQD display shows a checkbox for each of these possible invalidity conditions.

getengvalue <parval>	Returns the engineering (i.e. calibrated) value of the parameter. It returns “” (i.e. null) if <parval> does not have an engineering value. If the engineering value is invalid (function isengvaluevalid returns false and getengvalidity returns non-zero), “” (i.e. null) is returned.
isengvaluevalid <parval>	Returns 1 (logical true) it the enginnering value of the parameter is valid. It returns 0 (logical false) if the value is invalid or <parval> is empty.
getengvalidity <parval>	Returns the validity of the engineering value of the parameter as a bit-set. For details, see getrawvalidity.
getdefaultvalue <parval>	Returns the engineering (i.e. calibrated) value of the parameter if it exists, otherwise the raw value. If the default alue is invalid (function isdefaultvaluevalid returns false and getdefaultvalidity returns non-zero), “” (i.e. null) is returned.
isdefaultvaluevalid <parval>	Returns 1 (logical true) it the default value of the parameter is valid. It returns 0 (logical false) if the value is invalid or <parval> is empty.
getdefaultvalidity <parval>	Returns the validity of the default value of the parameter as a bit-set. For details, see getrawvalidity.

Function name	Return value
getoolstate <parval>	Returns the OOL state of the parameter. Possible return values are: NOMINAL WARNING ALARM SCC
getsccstate <parval>	Returns the SCC (Status Consistency Check) of the parameter. Possible return values are: SCC_ UNINIT SCC_INIT SCC_DISABLE SCC_OFF
gettimestamp <parval>	Returns the time stamp of the parameter.
isvalid <parval> **This function is not supported in TOPE/CCS**	returns 1 (true) if the raw and the engineering value are valid.[1]
tostring <parval>	Returns a human-readable representation on the parameter attributes
validitystr <validity>	Converts a validity value (as returned by getrawvalidity,..) into a human-readable list of strings.

Example:

```
set varP1 [fetch P1]
if {[israwvaluevalid $varP1] && [getrawvalue $varP1] > 10} …
```

6.3 Subscribe to a parameter

subscribe <ParameterName> referby <Varname> ?whenever <UpdateMode>?

<ParameterName>	Telemetry Parameter Name
<Varname>	Name of a TCL variable, which is used for carrying all attribute values of the Parameter as returned by SCOS. See the definition of fetch.
<UpdateMode>	UPDATE: parameter has been updated due to TM packet arrival, or because a

[1] Thus this function is only useful with parameters, which have an associated engineering value.
(Rationale: If this function would ignore the validity of the engineering value if there is no engineering value, it would return true in a scenario where the engineering value is invalid but has never arrived due to a “whenever CHANGE” subscription.)

This function is not supported in TOPE/CCS	contributing parameter has been updated. The latter can be • A source parameter if the parameter is synthetic • A validity parameter • A limit applicability parameter • A calibration selection parameter CHANGE: value has changed The default is UPDATE.

An error will be raised if the <ParameterName> was already subscribed in a previous call of subscribe.

6.4 Subscribe to a set of parameters

subscribeset <ParameterNames> referby <Varname> ?whenever <UpdateMode>?

<ParameterNames>	TCL list expression containing the names of the parameters. E.g. {G123 G124} or [list G123 G124] are possible ways to construct such a list.
<Varname>	Name of a TCL variable, which is used for set notification.
<UpdateMode> **The whenever clause is not supported by the TOPE/CCS**	See subscribe (single parameter).

subscribeset is an advanced command, allowing subscribing multiple parameters at one time. Whenever one or several parameters from the specifed set are updated, the following happens:

1. For each updated parameter, the TCL variable <Varname>_<ParameterName> will be updated with a parameter value. The same access functions as with fetch and subscribe are applicable.
2. <Varname> will be set to a TCL list containing the names of the parameters from previous step.

An error will be raised if any of the parameters was already subscribed in a previous call of subscribe(set).

Example:

```
subscribeset {ZZ001 ZZ002 ZZ003} referby vSet
waitfor vSet
foreach x $vSet {
     putlog “$x has been updated!”
```

```
        set varname "vSet_$x"
        # TCL trick: [set x] returns value of x !
        set parval [set $varname]
        putlog [tostring $parval]
    }
```

6.5 Unsubscribe to parameters

unsubscribe <ParameterList> |-all

<ParameterList>	List with names of previously subscribed parameter(s)
-all	Cancel all currently active parameter subscriptions.

Example:

```
subscribe ZZ00A referby vP
waitfor vP
unsubsribe ZZ00A
# set example:
subscribeset {ZZ001 ZZ002 ZZ003} referby vSet
waitfor vSet
unsubscribe {ZZ001 ZZ002 ZZ003}
```

6.6 Wait For a condition to become true

waitfor <Varname> ?-until <Expression>?

<Varname>	The name of a TCL Variable.
<Expression>	TCL code fragment, which is evaluated every time <Varname> is updated.

Waits until <Varname> is set, then evaluates the <Expression>. If the <expression> is True, go to next statement, else wait for a new value to be set to <Varname> is evaluated.

If the –expression keyword and the expression are omitted, waitfor waits exactly once.

Examples:

```
tcsend cmd1 referby vCmd1
waitfor vCmd1 –until {[getcompleted $vCmd1]}
subscribe P1 referby vP1
waitfor vP1 –until {[israwvaluevalid $vP1] && [getrawvalue $vP1] > 10}
```

Waits until a valid raw value of P1 is greater than 10.

6.7 Wait For Time Interval

waittime <time>

<time>	The amount of relative time to wait.
	<time> must be a positive delta time. (The syntax of time values is described in 6.1.1.)
	Examples for waiting 1 second:
	`waittime 1`
	`waittime 1.000000`
	`waittime +00.00.01.000000`

`waittime` waits until the specified amount of time has elapsed.

Note: processing of events will take place while waittime is executed.

6.8 Watchdog

watchdog <wdName> triggeredby<Var-List> <condition> <Action>

<wdName>	Name of the Watchdog
<Var-List>	List of Variables to be waited for
<Condition>	Pre-Condition of the Watchdog
<Action>	Action to be evaluated

The watchdog will be triggered when any of the variables that are listed in <Var-List> changes. Then, if the condition evaluates to true, the action will be executed.

The watchdog function behaves as following:

```
IF <wdName>_lock THEN exit ;# avoid watchdog code being called twice
ELSE    SET <wdName>_lock 1
IF <Condition> THEN <Action>
SET <wdName>_lock 0
```

Example:

```
subscribe P1 referby vP1 whenever CHANGE
subscribe P2 referby vP2 whenever CHANGE
watchdog wd1 triggeredby {vP1 vP2} {[getrawvalue $vP1] == 0xFF} \
{puts "WD1: P1 Eng [getengvalue $vP1]"}
```

Notes:

- **The `watchdog` function is not available on TOPE/CCS!**
- **As an alternative to watchdog, the standard TCL command `trace` can be used to reach the same goal[2]. The trace command is described in the TCL manual pages.**
  Example:

```
proc mywatchdog {n n2 op} {
     # there must be 3 formal parameters, see manual page!
     set paramVal [set ::$n]
     if {! [israwvaluevalid $paramVal]}{
         putlog "WD: [getname $paramVal]: Raw is now
invalid!"
     }
}

subscribe P010 referby p1
subscribe P010 referby p2
trace variable p1 w mywatchdog
trace variable p2 w mywatchdog
# other actions
```

6.9 Log a message

putlog <string>

<String>	String to be logged. A time stamp is added to the log line

Example:

putlog OK.

Results in output

2001.015.16:59.59.000: OK.

If no user-defined logfile is currently active (see openlog, closelog below), the text is written to the standard logfile.

6.10 Open log file

openlog <filename> [w | a]

<filename>	Opens the Log file <filename> where the output of all subsequent putlog calls will be written (in addition to the Logging Window).

[2] In fact, trace is more general. The implementation of watchdog uses trace internally.

[w / a]	w: Open a new file or an existing file in overwrite mode. [default] a: Open an existing file in append mode

This function is not available in TOPE/CCS !

6.11 Close log file

closelog

The log file is closed. After the log file has been closed, putlog output will be written to the standard log again.

This function is not available in TOPE/CCS !

6.12 Miscellaneous functions

getstageorder <Stage>

This function returns the order number for a command stage:

PTV_DYNAMIC	0
PTV_STATIC	1
MCS_RELEASE	2
UV_GS_RECEIVE	3
UV_GS_UPLINK	4
UV_ONB_ACCEPT	5
EV_APP_ACCEPT	6
EV_START_EXEC	7
EV_PROGRESS_0	8
EV_PROGRESS_1	9
EV_PROGRESS_2	10
EV_PROGRESS_3	11
EV_PROGRESS_4	12
EV_PROGRESS_5	13
EV_PROGRESS_6	14
EV_PROGRESS_7	15
EV_PROGRESS_8	16
EV_PROGRESS_9	17

EV_END_EXEC	18
any other argument	-1

It can be used to wait for a minimum stage:
Example:

```
tcsend Cmd13 referby vCmd13
waitfor vCmd13 {[getstageorder [getstage $vCmd13]] >=
                        [getstageorder EV_APP_ACCEPT]}
```

7 EGSE SPECIFIC FUNCTIONS / DATA STRUCTURES

7.1 EGSE Message Format

The EGSE Messages are those messages exchanged between two ground applications using the ESA EGSE Router protocol. These messages have the structure compliant to [A1] as described below:

Reference	Field	Octets/Bits size
EGSERouter Byte Array Message Format [A1]		
	DstID	2 octets
	SrcID	2 octets
	Token	4 octets
	Time	8 octets
	Msg_Type	1 octet
	Spare	1 octet
	S/C id	2 octets = 0x66 for PROBA
	Application Data	Any

The DstID field contains the unique identifier of the EGSE ground application to which the message is sent (The Destination Application).

The SrcId field contains the unique identifier of the EGSE ground application, which has sent the message (The Source Application).

The Token field contains the unique identifier of the message given by the sending application and used whenever required, to report on the processing of the message.

The Time field contains the time in which the message has been issued. The format of the Time field is:

> The time field format is compliant to the SUN OS time format. It is of 8 bytes lengths (2 long unsigned integers) encoded as:

Byte 0, 1, 2, 3	Unsigned integer equal to the cumulative seconds from the 1970-01-01T00:00:00 epoch.
Byte 4, 5, 6, 7	Remaining Microseconds.

The Msg_Type field contains a unique identifier of the type of messages and identifies the structure of the Application Data part of the message.

The following messages types are used in the context of exchange of request and/or report messages required for the control and monitoring of any EGSE ground applications:

Msg_ Type	***Description***	***Definition***
1	SCOE Command	Template 1
2	SCOE Command Verification Report	Template 2
3	SCOE Observation Report	Template 3
4	Sending TC Packet Request	Template 4
5	Sending TC Packet Verification Report	Template 5
6	TM Packet Report	Template 6
7	TM Frame Reports	Template 7

Other Msg_Types may be defined to allow other types of messages to be exchanged between applications.

The following sections give the complete definition of the Application Data structures for the Msg_Types 1 to 6.

7.2 The EMCS Request acknowledgement Policy

The Packet Utilisation Standard introduces 4 types of acknowledgements that users can request to follow the execution of the command by the onboard application processes:

- the Acceptance of the command by the Application Process,
- the Starting of execution of the command,
- the Progress of stages of execution of the command,
- the Completion of execution of the command.

The EMCS makes use of command verification reports allowing the processing of commands to be followed not only when they are executed by the onboard application processes but also when they are processed by the EMCS kernel, the NCTRS and the SCOE.

- Any SE (onboard application process and ground process) that can process requests can make use of the acknowledgement facility to report on the status of the execution of request sent to them. The 4 types of acknowledgements introduced by PUS can be used. PUS service type 1 subtypes 1 to 8 can be generated using Template 2 for SCOE commands and Template 6 for TC packets reporting.
- 3 acknowledgement requests (GS_Ack "ground station acceptance", UL_Ack "uplinked", ONB_Ack "onboard acceptance") and 6 associated acknowledgement reports (service type 9 to 14) have been added to follow the ground TC front ends progress in sending telecommands. The ground station and the EGSE TC Front End applications use these new types of acknowledgements.
  - The GS Ack is returned by the Ground Station/SCOE TC FrontEnd when a telecommand is received.
  - The UL_Ack is issued by the Ground Station/SCOE TC FrontEnd when a telecommand issued in BD mode has been sent to the spacecraft.
  - The ONB_Ack is issued by the Ground Station/SCOE TC FrontEnd when the Spacecraft confirms the reception of a telecommand issued in AD mode.

The resulting acknowledgement policy is as following.
When defining commands the MIB, users may request the following verifications to be performed and acknowledgement reports to be issued:

Check	***Meaning***	***Comment***
Static PTV condition	Static Pre-Transmission verification.	
Dynamic PTV condition	Dynamic Pre-Transmission verification.	
GS_Ack requirement	These 3 acks are system acknowledgments. The EMCS kernel requests them. Users can only use the associated reports.	*A pre-requisite to this requirement is that the routing application process (e.g. TC Front End) is able*
UL_Ack requirement		

Check	*Meaning*	*Comment*
ONB_Ack requirement		*to issue these reports.*
Acceptance_Ack requirement	The user may require the Acceptance_Ack related reports (Acceptance ack/nack) to be issued by the application process.	*A pre-requisite to this requirement is that the application process is able to issue these reports.*
Start_Ack requirement	The user may require the Start_Ack related reports (Start ack/nack) to be issued by the application process.	
Progress_Ack requirement	The user may require the Progress_Ack related reports (Progress ack/nack) to be issued by the application process.	
Completion_Ack requirement	The user may require the Completion_Ack related reports (Completion ack/nack) to be issued by the application process.	
CEV condition	Confirmation Execution verification. The users may require a condition to be verified to declare the command successfully executed. This condition is verified by SCOS.	

Note: SCOS2000 implements the Acceptance, Start, Progress, Completion reporting mechanism using the PUS service 1 or using TM conditions. The CEV is either computed using the Completion report or via TM. Both should be available.

When commands are initiated by the TOPE the following reports are generated according to the final destination of the commands and the capability of the application process to report on the state of execution of commands,

TOPE		*EMCS*	*Conditional Reporting*
Initiate Command	→		
	←	ECH ack/nack	Always generated by the External Command Handler.
	←	SPTV ack/Nack	User decision according to MIB definition and TCSPACON setting.
	←	DPTV ack/Nack	User decision according to TCSPACON setting.
	←	GS ack/Nack	Always delivered.
	←	UL ack/Nack	Always delivered
	←	ONB ack/Nack	Always delivered (Only in AD mode)
	←	Acceptance ack/Nack	User decision according to the application process capabilities.

	←	Start ack/Nack	User decision according to the application process capabilities.
	←	Progress ack/Nack	User decision according to the application process capabilities.
	←	Completion ack/Nack	User decision according to the application process capabilities.
	←	Completed Notification	Sent by the ECH after Completion Ack or after any Fail condition

Note: the order in which the acknowledgement reports are received by the EMCS kernel cannot be guaranteed.

7.3 SCOE Request

The SCOE command template is used to send commands to a SCOE. The request format is similar to the PUS format where:

- The CCSDS and PUS Secondary header is removed and replaced by specific EGSE headers.
- The PUS Application Data structure remains (see below the Template 1 Parameters structure).
- The Parameters structure format depends of the Service Type/Subtype compliant to PUS. The MIB database contains the definition of the Parameter structure.

TEMPLATE 1: SCOE command

Field	Bit size	Value
→		
DstID	*2 octets*	
SrcID	*2 octets*	
Token	*4 octets*	
Time	*8 bytes*	
Msg_Type	*1 octet*	*= 1*
Spare	*1 octet*	*= 0*
S/C id	*2 octet*	*= 0x66 – PROBA*
→		
Service Type	8 bits	
Service Subtype	8 bits	
Ack	8 bits	Bit 0,1,2,3 (not used)
		Bit 4 = Completion_Ack requirement
		Bit 5 = Progress_Ack requirement
		Bit 6 = Start_Ack requirement
		Bit 7 = Acceptance_Ack requirement
Parameters	any	

7.4 SCOE command Verification Report

Senders of SCOE commands may require the destination application to return a set of verification reports to the sender.

For this purpose, the "ack" field of the request message expresses the requirements for verification reports.

For each required report, the destination application will return a verification report as described below.

TEMPLATE 2: SCOE command Verification Report

Field	Bit size	Value
→		
DstID	*2 octets*	
SrcID	*2 octets*	
Token	*4 octets*	
Time	*8 bytes*	
Msg_Type	*1 octet*	*= 2*
Spare	*1 octet*	*= 0*
S/C id	*2 octet*	
→		
Service Type	8 bits	1
Service Subtype	8 bits	1 to 8 depending of type of acknowledgement
SCOE command Token	4 octets	Value of the related "SCOE command Token"
~~*Step Number*~~[3]		~~*Only for Service subtype = 5 or 6*~~
~~Code~~[4]	~~8 bits~~	
Parameters		

The meaning of the different Service subtype is as following:

Subtype	***Meaning***

[3] Reporting progress at step level is currently not supported by the EMCS server.
[4] This code is currently not supported by the EMCS server.

1	Successful Acceptance
2	Failed Acceptance
3	Successful Start
4	Failed Start
5	Successful Progress
6	Failed Progress
7	Successful Completion
8	Failed Completion

7.5 SCOE Observation Report

The EGSE Application Process can send any sort of feedback report message to SCOS following an approach similar to the PUS Telemetry packet.

An EGSE Application Process can send to SCOS observation report messages using the [A1] service “smData”.

A smData message has the following structure:

TEMPLATE 3: SCOE Observation Report

Field	Bit size	Value
→		
DstID	*2 octets*	
SrcID	*2 octets*	
Token	*4 octets*	
Time	*8 bytes*	
Msg_Type	*1 octet*	*= 3*
Spare	*1 octet*	*= 0*
S/C id	*2 octet*	
→		
Service Type	8 bits	See PUS
Service Subtype	8 bits	See PUS
Parameters	any	See Note 1.

Note: When creating an observation report message, the MsgData→Parameters structure is similar to the PUS source data structure of report packets.

7.6 Sending TC Packets

To send TC Packets to the Spacecraft using the EGSE requires the use of an EGSE TC Front End. The following template is used.

TEMPLATE 4: SendingTC Packet

Field	Bit size	Value
→		
DstID	*2 octets*	*ID of the TC Front End SE*
SrcID	*2 octets*	
Token	*4 octets*	
Time	*8 bytes*	
Msg_Type	*1 octet*	*= 4*
Spare	*1 octet*	*= 0*
S/C id	*2 octet*	
→		
Ack[5], [6]	8 bits	Bit 0 = 0 (not used)
		Bit 1 = ONB_Ack requirement (AD mode only)
		Bit 2 = GS_Ack requirement
		Bit 3 = UL_Ack requirement
		Bit 4 to 7 (not used.)
Channel	8 bits	VC number
MAPID[7]	8 bits	Multiplexed Access Point Identifier
Service	8 bits	AD = 0; BD = 2;
TC Packet	any	Full CCSDS/PUS TC Packet.

[5] When commands are sent in BD mode, the UL_Ack requirement asks for a report to be issued when the TC packet has been completely sent to the Spacecraft (ack report) or partially or not sent (nack report).
[6] When commands are sent in AD mode, the UL_Ack requirement asks for a report to be issued when the TC packet is confirmed (ack report) or not confirmed (nack report) in accordance to the TC packet protocol (i.e. as result of the processing of the CLCW).
[7] Currently, the correspondence (as made for PROBA) between the Service type/subtype of commands and MAPID is hardcoded within the SCOS to SCOE interface.

7.7 Sending TC Packet Verification Report

The TC Front End reports to SCOS the status of the processing of commands sent to the Spacecraft using the GS_Ack, UL_Ack and ONB_Ack reports (**see also [8]**).

The EMCS kernel injects the Template5 message as Telemetry packets (using the SPID 10005 / EGSE_VER_TC_GS_PKTID)

The TC Front End reports have the following structure.

TEMPLATE 5: Sending TC Packet Verification Report

Field	Bit size	Value
→		
DstID	*2 octets*	
SrcID	*2 octets*	
Token	*4 octets*	
Time	*8 bytes*	
Msg_Type	*1 octet*	*= 5*
Spare	*1 octet*	*= 0*
S/C id	*2 octet*	
→		
Service Type	8 bits	1
Service Subtype	8 bits	**9 to 14 depending of the required acknowledgement and its state.**
EGSE Command Token	4 octets	**As set with the corresponding sending TC packet request.**
Code	8 bits	
Parameters	any	

Subtype	***Meaning***
9	Successful GS
10	Failed GS

[8] SCOS2000 can also require the spacecraft to send telecommand verification reports as defined in the PUS. These TC verification reports will come directly from the Ground Segment TM Front End (or Ground Station) and be processed in a conventional way.

11	Successful UL
12	Failed UL
13	Successful Onboard Acceptance
14	Failed Onboard Acceptance

7.8 TM Packet Report.

When the EGSE TM Front End receives TM from the spacecraft, it encapsulates the TM Packet into a report message as following.

TEMPLATE 6: TM Packet Report

Field	Bit size	Value
→		
DstID	*2 octets*	
SrcID	*2 octets*	
Token	*4 octets*	
Time	*8 bytes*	
Msg_Type	*1 octet*	*= 6*
Spare	*1 octet*	*= 0*
S/C id	*2 octet*	
→		
TM Packet	any	**Full CCSDS/PUS TM Packet.**

Nota: *TM packets received by the TM Front END and sent to the EMCS Kernel may contain any housekeeping or scientific reports. It may also contain Service Type 1 reports.*

The Service Type 1 reports are injected with the Packet ID 10006 (EGSE_VER_TC_REP_PKTID) known by the SCOS-2000 verifier.

7.9 Processing data messages coming from the EGSE Router.

7.9.1 SCOS INTERNAL ARCHIVES

The SCOS/EGSE system stores any received message (i.e. Templates 2, 3, 5, 6) into SCOS internal packets. Each internal packet is identified by a given SPID.

Some SPID numbers (from 0 to 100) are reserved for SCOS kernel specific needs. This number allocation can be changed for a given mission by update of the MISCconfig file and the minimum EGSE data set (refer to section 4.1.1).

The SCOS/EGSE determines what SPID to use according to a given combination of the value of key fields extracted from the received messages:

- the Msg_Type field of the EGSERouter Byte Array Message Format (i.e. Template number)
- the APID,
- the service type and subtype,
- the PI1 and PI2 fields.

  The PI1 and PI2 are fields that can be located anywhere within the packet for a given Template number, service type and subtype (refer to [R5], section 3.3.2.4.2 Packets identification criteria).

All SCOS archives are stored within the $scosii_homedir/hfiles directory. A given SPID archive is made of an index file and a *data file*. The structure of the *data file* is made of

- a header that is used by SCOS for retrieval purpose, and
- a body that contain the incoming data report (refer to MIB ASCII files plf.dat and vpd.dat).

7.9.2 SPECIFIC PROCESSING PERFORMED ON INCOMING DATA

The SCOS/EGSE system process any message received from the EGSE Router. Specific functions are performed according to the Msg_Type, Service Type and Subtype of the received messages.

7.9.2.1 Time Stamping.

The SCOS system time stamps any message prior to store it within its internal archives. The EGSE is configurable to allow time stamping according to a given AIT test configuration.

Time stamping is made according to the value of the MISCconfig variable EGSE_TM_FILING_TIME_TYPE as following:

- When EGSE_TM_FILING_TIME_TYPE = **1**, the time stamping is made using the time of reception of the message on the local machine on which SCOS is running (i.e. EMCS time).

  When EGSE_TM_FILING_TIME_TYPE = **2**, the time stamping is made using the time field value of the EGSERouter Byte Array Message Format (refer to section 7.1)

- When EGSE_TM_FILING_TIME_TYPE = **3**,
  - For all TM packet messages that contain a time field, the time stamping is calculated by correlating the related OBT time.

    Refer to [R6] to configure SCOS for a given mission dependent time correlation algorithm.
  - For any other messages, the time stamping is made using the time field value of the EGSERouter Byte Array Message Format (refer to section 7.1).
  - An sanity check is performed on the calculated OBT If the delta between the OBT in the packet and the current time is more than the threshold (given by MISCconfig variable EGSE_TM_OBT_THRESHOLD) it is ignored and the current time is used. In the TM Packet history the packet is shown as type PR, rather than PG. A warning is given for every “insane” packet

7.9.2.2 Checksum Verification

The EGSE can be configured to verify the checksum of received TM packets, using the MISCconfig variables:

- EGSE_SEND_CHKSUM_ERROR_PKT
  - When EGSE_SEND_CHKSUM_ERROR_PKT = **0**, a TM packet failing the checksum verification is not archived.
  - When EGSE_SEND_CHKSUM_ERROR_PKT = **1**, a TM packet failing the checksum verification is archived within the SCOS data stream given by the EGSE_STREAM_ID_FOR_CRC_ERROR variable.
- EGSE_STREAM_ID_FOR_CRC_ERROR: the SCOS data stream to use to archive TM packets that fail the checksum verification.

Note 1: When archiving TM packets that fail the checksum verification, the SCOS EGSE tries to retrieve the SPID key according to the procedure given in section 7.9.2.5. If the SPID is found, the TM packet is archived within this SPID. When archiving such packet, the packet is flagged as non-valid resulting in all contained parameters are as well flagged non-valid.

Note 2: For missions that do not use the CRC checksum and the related PEC field at the end of the CCSDS TM packets, the SCOS EGSE system can be configured accordingly. Refer to [R6].

7.9.2.3 Request Verification processing

This function is called for the following message types:

- "SCOE Command verification", i.e. Msg_Type = 2
- "Sending TC Packet verification", i.e. Msg_Type = 5
- "PUS Service 1 TM Packet", i.e. Msg_Type = 6, Service Type = 1 and Service Subtype is between 1 and 8.

The Request verification function performs the following:

1. Call the SCOS TC verifier related function.
2. Store the received message within SCOS internal packets using System specific SPIDs:

   System-SPID = EGSE_VER_SCOE_PKTID for SCOE for messages of type "Command verification".

   System-SPID = EGSE_VER_TC_GS_PKTID for messages of type "Sending TC Packet verification".

   - System-SPID = EGSE_VER_TC_REP_PKTID for messages of type "PUS Service 1 TM Packet".

   Within the EGSE Minimum database:

   - a set of parameters are defined related to these system specific SPIDs. They are prefixed by one of "ZV2", "ZV5" and "ZV6".
   - Within the tpcf.dat file, these SPIDs are identified as EGSE_T2_VER, EGSE_T5_VER, EGSE_T6_VER.
3. Search within the MIB ASCII pid.dat and pid3.dat files whether a given request verification report message having a specific structure to process is also required to be stored within a mission dependent SPID.

   Note 1: This function is new in the SCOS Release 2.3e p3 version and allows processing reports that for a given APID, Service Type, Service subtype makes use of a Code and parameters fields.

   Note 2: This function does not cover all capabilities covered by the PUS standard for uniquely identify the presence and structure of the Service 1 related code and parameters.

7.9.2.4 SCOE observation messages processing

This function is called for the "SCOE Observation" type of messages, i.e. Msg_Type = 3.

For such message, the following key fields are used to identify the mission specific SPID to use:

- SrcID, i.e. the APID of the application that has issued the message.
- Service Type and Subtype.

- Depending of the content of the pic3.dat ASCII file, the PI1 and PI2 fields.

If an entry can be found for these key fields within the pid3.dat ASCII file, the message is stored within the related SPID.

If no entry can be found within the pid3.dat table, the message is "rejected" and filed within the system specific SPID called EGSE_WRAPPER_PKTID_3.

Within the EGSE Minimum database, parameters identified with a prefix "ZO", are defined to support a first analysis of these rejected messages.

When the cause of the reject is that the MIB database does not contain the declaration of the SPID related to the received message, users may update the MIB with the new definition and run the SCOS task launcher PDS_DISP (refer to section 5.1) to read the EGSE_WRAPPER_PKTID_3 archive and copy the rejected report into its own SPID archive (refer to section 7.9.2.6).

Within the tpcf.dat file, the rejected SCOE observation SPID is identified by EGSE_T3_UNKN.

Note: When storing the SCOE observation message within the SCOS SPID related archive, the body of the internal packet is filled with the full SCOE observation message as defined within section 7.5.

According to the record length of the SPID related file (refer to its definition within the HFAconfig file), it may happen that padding bits are added at the end of the body.

7.9.2.5 TM Packets processing

This function is called for the (other than service 1 related) "TM Packet" type of messages, i.e. Msg_Type = 6.

For such message, the SCOS conventional key fields of TM Packets are used to identify the mission specific SPID to use:

- The APID contained within the CCSDS header of the TM Packet.
- The Service Type and Subtype of PUS data field header
- Depending of the content of the pic.dat ASCII file, the PI1 and PI2 fields.

If an entry can be found for these key fields within the pid.dat ASCII file, the message is stored within the related SPID.

If no entry can be found within the pid.dat table, the message is "rejected" and filed within the system specific SPID called EGSE_WRAPPER_PKTID_6.

Within the EGSE Minimum database, parameters identified with a prefix "ZTM", are defined to support a first analysis of these rejected messages.

When the cause of the reject is that the MIB database does not contain the declaration of the SPID related to the received packet, users may update the MIB with the new definition and run the SCOS task launcher PDS_DISP (refer to section 5.1) to read the EGSE_WRAPPER_PKTID_6 archive and copy the rejected report into its own SPID archive (refer to section 7.9.2.6).

Within the tpcf.dat ASCII file, the rejected TM packet SPID is identified by EGSE_T6_UNKN.

Note: When storing the TM Packets within the SCOS SPID related archive, the body of the internal packet is filled with the full TM Packet including CCSDS and PUS headers and PEC fields.

According to the record length of the SPID related file (refer to its definition within the HFAconfig file), it may happen that padding bits are added at the end of the body.

7.9.2.6 Re-distributing unknown reports into proper SPID archives

When valid SCOE observation messages or TM packets are received by SCOS but do not have any correspondence with a MIB defined SPID, they are "rejected" according to the scenario introduced in sections 7.9.2.4 and 7.9.2.5.

A specific function is available within the EMCS that allows unknown messages that have been filed in the EGSE_WRAPPER_PKTIDs area to be re-distributed within their proper archive files after new entries have been added to the MIB and loaded within SCOS.

The procedure for such re-distribution of archive data is the following:

1. Shutdown the EMCS kernel (i.e. SCOS)
2. Create new MIB report definitions (i.e. new SPID entries for SCOE Observation Messages or TM Packets). Don't forget updating the HFA and TMD config files.
3. Import the updated MIB into SCOS.
4. Start the server and DESK processes of SCOS.
5. Start the PDS_DISP task.

The PDS_DISP task verifies whether the original SPIDs of the messages that have been filed under the EGSE_WRAPPER_PKTID_3 and EGSE_WRAPPER_PKTID_6 SPIDs archives have been properly defined within the MIB. If this is the case, the corresponding messages are copied within their proper archive.

A log is made within the Event log and within the $HOME/PDSdispatch.log.

Note: in the current version, the unknown message archives are not cleaned. The PDSadmin tool can be used to delete messages from the unknown message archives.

APPENDIX A MS-WINDOWS TM FRONT END USER MANUAL

For the purpose of demonstrating the functioning of the EMCS, a demonstrator of a simple TM Front End SCOE has been delivered together with the EGSE Router. This demonstrator allows replay of archived PROBA TM frames. The following tables detail the commanding and control of this EGSE TM Front End. The MIB ASCII files delivered with the EMCS contains the data required for operating this front end.

The TM Front End is made of up to 8 system elements. 3 of them have been configured for PROBA:

0x8810	TM Front End : VC0
0x8811	TM Front End : VC1
…	…
0x8817	TM Front End : VC7

Commands can be sent to any TM Front End Virtual Channel Controller to require TM packets to be sent to any client. Filtering mechanisms have been encoded to allow filtering by APID.

The TM Front End Controllers commands are:

TMFE	VC<n>APON	Activate sending of VC<n> TM Packets related to APID <apid>	
		Destination	0x881<n>
		Type, Subtype	8, 1
		TMr_Id	SE Id that shall receive the TM (Uint 16 bits)
		Apid	A valid S/C APID (Uint 16 bits)
TMFE	VC<n>APOF	Deactivate sending of VC<n> TM Packets related to APID <apid>	
		Destination	0x881<n>
		Type, Subtype	8, 2
		Apid	A valid S/C APID (Uint 16 bits)
TMFE	VC<n>ALLON	Activate sending of all VC<n> TM Packets	
		(same as VC<n>APON with APID = 0xFFFF)	
TMFE	VC<n>ALLOFF	Deactivate sending of all VC<n> TM Packets	
		(same as VC<n>APOF with APID = 0xFFFF)	

APPENDIX B CONVENTIONS

B.1 Unsigned Integer values format convention

Hexadecimal values are always prefixed by the two characters "0x". Example 0x8000 is equal to the decimal value 32768

B.2 Bit/Octet Numbering Convention

The following convention is used to identify each bit in a forward-justified N-bit field.

The first bit in the field to be transmitted (i.e. the most left-justified bit when drawing a figure) is defined to be “BIT 0”; the following bit is called “Bit 1” and so on up to “Bit N-1”.

When the field is used to express a binary value (such as an integer), the Most Significant Bit (MSB) shall be the first transmitted bit of the field (i.e. “Bit 0”).

An **octet** (i.e. a byte) is 8-bits length.

A short word is 16-bits length (i.e. 2 octets).

A word is 32-bits length (i.e. 4 octets).

A long word is 64-bits length (i.e. 8 octets)

The above convention for identifying a bit is also used for identifying each octet in a forward-ordered N-octet field.

APPENDIX C LIMITATIONS AND KNOWN BUGS

C.1 resolved.

C.2 TC verification across TOPE sessions

Usually, the state of TOPE is not preserved across sessions. However, when the TOPE is stopped after sending a request and before receiving all acknowledgements; restarting TOPE may imply that previous request acknowledgment are received. This behaviour is not consider faulty.

C.3 Tope TCL/Tk and MISCconfig variables

The TOPE environment must not override MISCconfig variables else TOPE crashes. This is due to the fact that the MISCcfgBase issues warnings via the C++ cerr stream when a MISCconfig variable is overridden. The use of the C++ I/O streams (cin, cout, cerr) is not compatible with the C language I/O used in TCL/Tk.

C.4 Known Limitations of TUBA

The co-operation of TUBA with TOPE is implemented as follows.

- Starting several procedures inside TUBA does not mean that they run in parallel. The first procedure will be stopped until the second one terminates. If the second procedure is executed step by step and the last statement is reached, actually two statements will be executed: the last one of the second procedure and the next statement of the first procedure.
- *TUBA patches the TCL command "source". When the user selects a file in the TOPE windows and clicks on the run button, the modified source command looks for the first statement in this file and sets a breakpoint. Then the original source is executed. This behaviour does not work properly if the first command is the definition of a TCL procedure:*
  (proc procname { parameter[s] } { body })
  because TUBA jumps over procedure definitions. Thus the current TOPE implementation shows a message dialog with the information that a TCL command should be inserted as the first command in the file.
  - Under certain circumstances TUBA performs a step into operation although the user has selected step over.

TUBA debugging sessions cannot be restarted. This feature has been disabled because restarting a debugging sessions causes trouble on the CORBA communication layer with SCOS.

C. 5 Display of Time Parameters

Time parameters make use of the defined epoch to convert the raw value into a meaningful time. The algorithm used works correctly for all raw values which result in a time after 1970 (the UNIX epoch). For times before 1970 the date is shown as very large (e.g. for 1958 epoch and a raw value

of 0 the date is shown as 2094 rather than 1958). This problem will only occur for uninitialised parameters; once the parameter is initialised to current time the display will be correct.

APPENDIX D HOW TO INSTALL SUSE FOR SCOS EMCS.

Installing SUSE 7.3 at ESTEC – For internal use only.
This procedure deviates from the ESOC internal procedure given below.

SuSE 7.3 Professional version
Select "Installation" (as opposed to "Manual Installation") → option write lilo to boot)
Language: English (GB)
Keyboard: English (US)
Time Zone: Global/GMT
Hardware Clock: GMT
New Installation
Software selection: "Default" "Default" includes the software categories "KDE Desktop" and "Help & Support" (as opposed to "Default with office")
+ category "Development"
+ acroread
- fetchmail
+ acct
- iproute2
- dhcpcd
- finger-server
- i4l
- i4lfirm
- yast2-config-adsl
- yast2-trans-adsl
- talk-server
- SuSEfirewall
- personal-firewall
- apmd
+ xrpm + prerequisites for xrpm: python (e.g. python-tkinter) libraries and blt
+ lincvs
+ pcl-cvs
+ pdksh
+ tcl-devel

(We leave the packages ppp, sunpppd, wvdial and fam in, though not needed nor wanted, due to dependencies with other packages)

Installing SUSE 7.3 at ESOC – For internal use only.

SuSE 7.3 Professional version
Select "Installation" (as opposed to "Manual Installation")
Language: English (GB)
Keyboard: English (US)
Time Zone: Global/GMT
Hardware Clock: GMT
New Installation
Discard suggested partitioning
Expert disk partitioning: ReiserFS for new file systems
we leave unmodified the existing ext2 file systems for /home and /local

/dev/sda1	root	6GB	ext2	RedHat 6.2
/dev/sda2	root	6GB	ReiserFS	SuSE 7.3
/dev/sda3	Extended	22GB		
/dev/sda5	/home	20GB	ext2	
/dev/sda6	/local	1.4GB	ext2	
/dev/sda7	swap	512MB	swap	

Software selection: "Default"[9] (as opposed to "Default with office")
+ category "Development"
+ acroread

- fetchmail
+ acct
- iproute2
- dhcpcd
- finger-server
- i4l
- i4lfirm
- yast2-config-adsl
- yast2-trans-adsl
- talk-server
- SuSEfirewall
- personal-firewall
- apmd
+ xrpm + prerequisites for xrpm: python libraries and blt
+ linkcvs
+ pcl-cvs
+ pdksh
+ tcl-devel
[10]

Monitor configuration: 1280x1024 24-bit 75 Hz
Ethernet and Sound card recognized and configured by the installer
DHCP disabled in the configuration of the Ethernet Interface
Standard DEVLAN IP and DNS parameters
Lilo configuration for dual boot SuSE 7.3 – RedHat 6.2:

[9] "Default" includes the software categories "KDE Desktop" and "Help & Support"
[10] We leave the packages ppp, sunpppd, wvdial and fam in, though not needed nor wanted, due to dependencies with other packages

```
disk=/dev/sda
   bios=0x80
boot = /dev/sda
vga = 791
read-only
menu-scheme = Wg:kw:Wg:Wg
lba32
prompt
timeout = 80
message = /boot/message

   image = /boot/vmlinuz
      label = suse
      root = /dev/sda2
      initrd = /boot/initrd
      append = "enableapic vga=0x0317"

   image = /boot/vmlinuz.suse
      label = failsafe
      root = /dev/sda2
      initrd = /boot/initrd.suse
      append = "disableapic ide=nodma apm=off"
      optional

   image = /boot/memtest.bin
      label = memtest86

   other = /dev/sda1
      label = redhat
      table = /dev/sda
```

Disable the display of user names on the KDE login screen:

KDE Control Center / System / Log-in Message: show users “none”

Configure NTP daemon and initial ntpdate command during boot:

Yast2 / System / rc_Config_Editor / Base_Admin / SuSE Configuration / Time Synchronisation:

XNTPD_INITIAL_NTPDATE = “devntp1a devntp1b”

Yast2 / System / rc_Config_Editor / Base_Admin / Start Variables / Start Network / start_xntpd

START_XNTPD = “yes”

Edit /etc/ntp.conf with the ntp server names

System Start-up customization:

Yast2 / System / rc_Config_Editor / Start Variables / Start_Administration

START_GPM = “no”
START_ACCT = “yes”
START_INETD = “yes”
START_SSHD = “no”

Other options in rc_config_editor submenus[11]:

Beautify_ETC_HOSTS = “no”
Check_ETC_HOSTS = “no”
Create_YP_CONF = “no”
Use_NIS_FOR_AUTOFS = ”no”

Inetd Configuration:

Disable “time” for both TCP and UDP
Disable “talk” and “ntalk”
Disable “finger”
Enable “in.ftp”
Enable shell “in.rshd –L”

Standard Maintenance accounts

Maintenance trusted host entry

[11] The tree organization of the Yast2 registry is too complicated. Use the search function to located the following options

Known Problems:

Mouse Pointer

The mouse pointer tended to disappear within the X screen test and tune dialog and, occasionally, on the desktop after login. Problems disappear after re-running Sax2 and letting the auto-detection of the mouse work. This gave

protocol = IMPS/2
device = /dev/psaux
Z axis mapping = "4 5"

for a Microsoft wheel-mouse.

Halt & Reboot accounts

Halt and reboot accounts were installed with the halt and reboot commands as login shells. The behavior of these accounts differs from Solaris in that the specified login shell is only executed, and consequently the machine shutdown, when the user opens a shell after having obtained a normal KDE desktop. This is not acceptable for a shutdown procedure and some investigation is still needed on how to implement this function perhaps customizing the Shutdown button on the login screen.

The halt and reboot accounts do work via switch user "su".

APPENDIX E CCS MIGRATION GUIDELINE

E.1 Use common subset of TOPE commands

Due to the H/P CCS user requirements, the set of commands supported by the TOPE/CCS differs from the set described in this manual.

The TOPE instrument language statements and clauses that are not available on the CCS are flagged within this document (i.e. "not supported by TOPE/CCS").

When preparing instrument procedures that need to run on the H/P CCS during system tests, the following differences should be acknowledged.

E.2 Parallel/Multiple Procedures

Inside an instance of TOPE, TOPE/IEGSE uses a kind of "cooperative" multitasking model with a single event loop (this is a consequence of the architecture). This has several consequences:

- TOPE/IEGSE procedures can use global variables to communicate with each other. It is also possible that one procedure sets a variable, which is used by the next procedure, after the first has terminated.
  These "spillovers" do not (and shall not) work on TOPE/CCS. Thus they are to be avoided. (It is also considered bad programming style if one procedure depends on variables set by another procedure, because it may lead to errors, which are hard to find.)
- Since there is only one event loop, updates of variables might be lost by a procedure. For example:
  procedure A executes waitfor. Then procedure B is started, until B hits another waitfor. Now all incoming TM/TC verification traffic will be visible to procedure B only. In the worst case, this may have the consequence that procedure A waits forever, even after B has terminated, because A has missed some updated it is waiting for.

The TOPE/CCS uses individual processes for each procedure. This implies that procedures run truly parallel.

Recommendation ➔ Use different instances of TOPE/IEGSE for procedures, which shall run in parallel on the CCS!

E.3 connect SCOE

In the SCOS-based IEGSE system, the connection between SCOS and the SCOE is established automatically. This is a feature of the protocol used.

On the HP/CCS system, connections must be established explicitly. Therefore the connect/disconnect command pair is provided:

```
connect SCOE1

# … do the job

disconnect SCOE1
```

E.4 attach/detach SCOE

The CCS requires that a test procedure *attach* to a SCOE before commanding it. This attaching is an exclusive lock, i.e. at most one procedure can be attached to a SCOE at a given point in time. Thus the command "attach <SCOEname>" must be inserted before the first tcsend to a SCOE:

```
attach SCOE1

# … do the job

detach SCOE1 ; # optional
```

Note: attach is required for SCOE commands but not for S/C commands.

E.5 Telemetry update mode

1. The TM update policy used on the CCS is similar, but not identical to "whenever UPDATE":
   - For parameters originating from a packet, the "referby" variable will only be updated by TOPE/CCS if a source packet has arrived. It will not be updated due to an update of a validity parameter, SCC state, limit applicability, or calibration selection parameter.
   - For synthetic parameters, the update policy is identical: An update notification will be received whenever one of the parameters in the OL expression is updated, or the validity parameter, SCC state, limit applicability, or calibration selection parameter is updated.
2. The "whenever CHANGE" is currently not available on the TOPE/CCS. However, waiting for a change of the raw value, for instance, is straight-forward:

```
set oldraw [getrawvalue $param]
waitfor param –until {[getrawvalue $param] != $oldraw}
```

APPENDIX F HOW TO UPDATE THE TOPE SYSTEM TO AVOID THE TUBA DEBUGGER TO STEP INTO CERTAIN PROCEDURES.

1. Identify what procedures are to be excluded from Stepping-In.
2. Edit $scosii_homedir/tcl/tuba/TOPE.ses adding the procedure names to the section "_exclusions".

   The wildcard character "*" can be used; e.g. "Fire*" will exclude all procedures which name starts with "Fire".

   If the procedures are in a TCL namespace and the complete namespace should be excluded, the "::<namespace>::*" syntax can be used; e.g. "::TOPE::*" will exclude all procedures defined in the top-level TOPE namespace.