EGSE TestControl
User Guide

Reference: PICC-ME-MN-002
Version: Issue 1
Date: July 12, 2004
Author: Erich Wiezorrek, MPE

Table of Contents

1 Introduction

1.1 Scope

This user manual describes how to install, start and run TestControl. In addition it specifies the structure of test procedures.

1.2 Applicable and Reference Documents

Id	*Reference*	*Title*
R1	TOS-EMG/01-1029/bm/sv	EGSE Based on SCOS 2000 The EGSE & Mission Control System Software User Manual

2 Installation

2.1 Installing Thirdparty Software

TestControl uses the Extended Tcl (TclX) which provides additional interfaces to the native operating (Unix kind) system.

2.1.1 SuseLinux

TclX is part of the Suse software distribution set.

Start YaST2, select the module Software --- Install/Remove software. Search for the package "tclx" and install it.

2.1.2 Solaris

Fetch the compressed TAR file "tclx8.3.5-src.tar.gz".

```
% tar -xzf tclx8.3.5-src.tar.gz
% cd tclx8.3.5
% ./configure
% make
% su
% make install
% exit
```

2.2 Installing TestControl

TestControl is an extension of the TOPE system delivered with SCOS 2000. The installation will add some files, most of them in subdirectories. In addition the symbolic links of the TOPE "images" are redirected to point to "wishx", the extended wish image. The original link information is saved so the original TOPE installation can be reinstalled.

TestControl is distributed via a compressed TAR file which shall be placed in the

directory where the original TOPE TCL files are located[1]. To find this directory the command

```
% find ~ -name TOPE.tcl
```

can be used, the file TOPE.tcl is located in this directory. In the following this directory will be called TOPE installation directory.

2.2.1 Installing a TestControl release

Go to the TOPE installation directory and unpack the TestControl TAR file. Go into the TestControl installation directory (for this release) and run the installation script. In the following example script the strings "*m.n*" must be replaced with the major/minor release numbers of the release you want to install.

```
% cd ~/tcl/TOPE   # or wherever your TOPE installation is
% tar -xzf TestControl-m.n.tar.gz
% cd tch-vm.n
% ./installTestControl.csh
```

The compressed TestControl TAR file can be removed.

2.2.2 Removing TestControl, restoring original TOPE

Go to any TestControl installation directory and run the uninstall script.

```
% cd ~/tcl/TOPE   # or wherever your TOPE installation is
% cd tch-vm.n     # where "m.n" is any installed release
% ./uninstallTestControl.csh
```

3 TestControl Setup

3.1 Environment Variables

Some environment variables must be set in order to get TestControl running.

INSTRUMENT_MODEL: Due the fact that the HCSS associates a lot of things (eg test procedures, test executions) with a specific instrument model which currently be used (in this case under test) this model must be specified with the environment variable INSTRUMENT_MODEL. The model name must match one known by the HCSS.

With the new MissionConfiguration concept of HCSS only the instrument name is needed.

TCH_TCL_DISPLAY: This environment variable must point to the same X11 display the HCSS TestControl server is using. For example if the DISPLAY environment variable of the shell which start the HCSS TestControl server is pointing to "pacs1:0" the TCH_TCL_DISPLAY environment variable must point to the same X11 display ("pacs1:0").

TCH_TCL_SERVER: In the standard setup this environment variable needs not to be set. It can be used to connect to an HCSS TestControl server with a different application name other than TCH_TCL_SERVER.

1 In the standard SCOS 2000 V2.3e release this directory is "~/tcl/TOPE".

3.2 X11 Server Access

In addition to the setting of the environment variables the TestControl system needs access to the X11 server which is set by the TCH_TCL_DISPLAY environment variable and which is shared with the HCSS TestControl server. Because of security settings enforced by TCL/TK standard "xhost" security settings are not sufficient. Instead the "xauth" application must be used to allow access to the X11 server. See your local "Xsecurity" man page for further documentation.

Example (used during PACS AVM testing):

For the PACS AVM test the PACS ICC team is using a dedicated X11 terminal with the hostname "irmultia" to display the console terminal windows for the HCSS testcontrol application and the EGSE router/gateway applications. Using a terminal window on "irmultia" the following command line can be used to retrieve the proper xauth keys:

irmultia> **xauth list**

The output may look similar to

```
irmultia.mpe-garching.mpg.de:0 MIT-MAGIC-COOKIE-1 043d1a1a0c057f3f20052114457e296f
irmultia/unix:0 MIT-MAGIC-COOKIE-1 043d1a1a0c057f3f20052114457e296f
```

The important line is the one starting with "irmultia.mpe-garching.mpg.de:0".

Now log-in to to the EGSE system (for PACS AVM it is "irsun01") using the SCOS 2000 operational account (the one you are also using with TOPE) and enter

```
irsun01> xauth add irmultia.mpe-garching.mpg.de:0 \
           MIT-MAGIC-COOKIE-1 043d1a1a0c057f3f20052114457e296f
```

You can use the mouse to cut and paste the key from the previous list command.

Finally repeat this step for the HCSS system.

When starting the TestControl server at the HCSS system make sure the DISPLAY environment variable is pointing to "irmultia:0".

4 Starting TestControl

TestControl is using the same task launcher as TOPE to start and monitor the necessary TOPE/TestControl subsystems. To execute a test procedure the usual TOPE procedure environment window can be used. In addition it is planned that test procedures can also be started from a shell command prompt (once the necessary TOPE subsystems are started already).

The TOPE/TestControl task launcher can be started either by pressing the "EXIF" button of the SCOS 2000 task launcher or by typing "exif.start" on the shell command line.

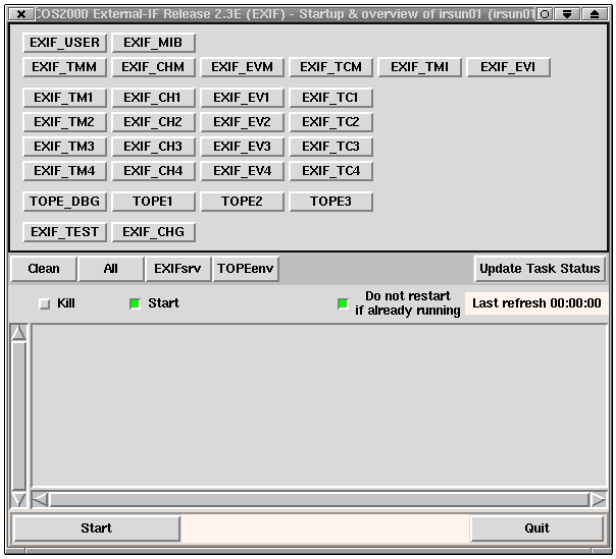

Illustration 1: The TOPE/TestControl task launcher

Following tasks must be activated:

EXIF_TMM
EXIF_TM1
EXIF_CHM
EXIF_CH1
EXIF_CH2
EXIF_CH3

To get a TOPE/TestControl procedure execution environment press "TOPE1", "TOPE2" or "TOPE3".

5 Using TOPE/TestControl

The TOPE/TestControl procedure execution is controlled by the TOPE command window which appears after starting "TOPE1", "TOPE2" or "TOPE3".

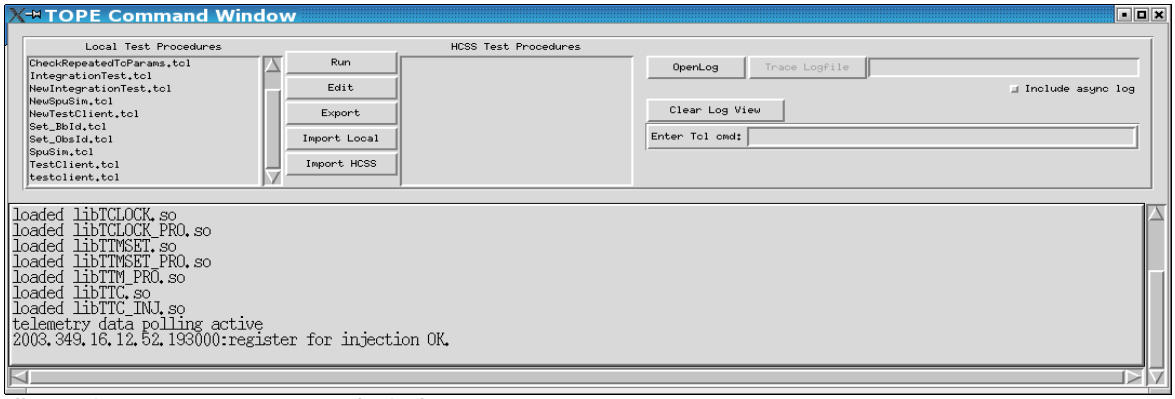

Illustration 2: TOPE command window

The TOPE command window can be destroyed by either using the window decoration frame or by killing it via the TOPE/TestControl task launcher.

5.1 Executing a Test Procedure

There are two list boxes in the TOPE command window to select a test procedure for execution. The left one labeled "Local Test Procedures" lists local test procedure files located in the directory pointed by the environment variable TOPE_DIR[2]. The right list box labeled "HCSS Test Procedures" shows the test procedures stored in the HCSS registry. See also section “TestProcedure Objects” on page 9.

After a test procedure is selected in one of the list boxes it can be started by pressing the "Run" button. When there are parameters defined for the test procedure (see page 10) a dialog window will pop up to inquire the parameter values.

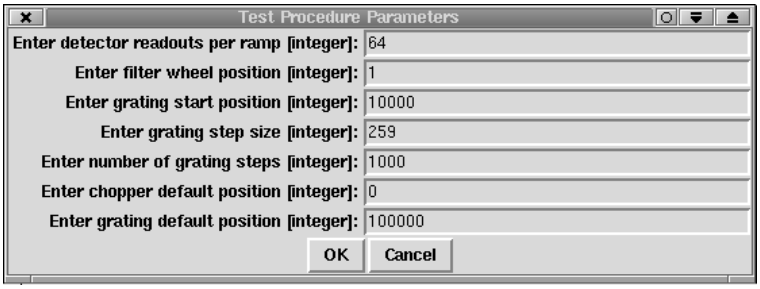

Illustration 3: Example for a Test Procedure Parameter dialog

Pressing the "OK" button will start the test procedure with these parameters (once they are valid), pressing the "Cancel" button will cancel the test procedure execution.

When there is no connection to the TestControl server of the HCSS a dialog window pops up to confirm that the test procedure shall be executed without HCSS logging. The actual reason for the connection failure can be different than the example in Illustration 4.

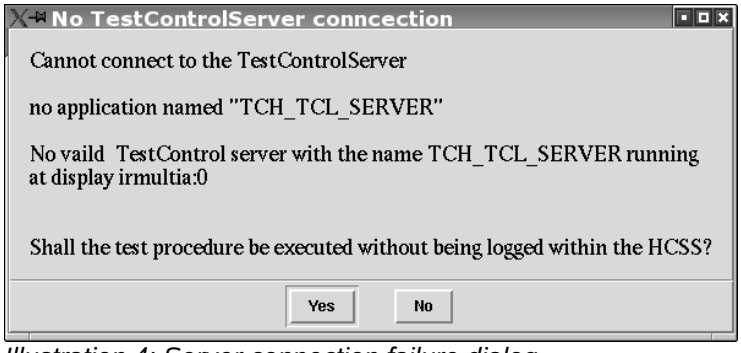

Illustration 4: Server connection failure dialog

Press the "Yes" button to start the test procedure execution in that case. Please note that any test procedure statements which requires interaction with the HCSS (like requesting the telecommands for an observation mode) will fail.

Once the test procedure execution is started the "Run" button changes to a "Stop/Abort" button. Pressing it suspends the execution of the test procedure and pops up a dialog where the operator may choose to either continue the test procedure or to abort it.

2 If the environment variable TOPE_DIR is not defined the local test procedure files are located in the directory "../TC", relative to the TOPE installation directory.

5.2 Editing a Test Procedure

To edit a test procedure select one in one of the list boxes and then press the "Edit" button. An editor window will pop up with the selected file displayed. The editor is defined with the environment variable EXIF_TOPE_EDITOR which is defined by TOPE to point to a TCL editor.

5.3 Importing a Local Test Procedure

Press the "Import Local" button. A file selection dialog will pop up for the selection of the local file to be imported. Press the "Open" button of the dialog once the proper file is selected. The file will be copied to the local test procedure registry of TOPE.

5.4 Importing an HCSS Test Procedure

Press the "Import HCSS" button. A file selection dialog will pop up for the selection of the local file to be imported. Press the "Open" button of the dialog once the proper file is selected. The file will be copied to the HCSS registry for test procedures.

5.5 Exporting a Test Procedure

To export a test procedure select one in one of the list boxes and then press the "Export" button. A file selection dialog will pop to specify the local file name for the exported test procedure.

5.6 Control of Logging

All log messages are written to the lower panel of the TOPE command window. If a test procedure is executed while TestControl is connected to the TestControlServer of the HCSS all log messages are also copied to the HCSS where they are stored within the test execution context.

In addition the log messages can be written to a log file. Press the “OpenLog” button of the TOPE command window to select the log file name and to open it. After that the button is renamed to “CloseLog” and can be used to close the log file. The “TraceLog” button allows an online view of the current log file.

Once the “Include async log” check button is pressed TOPE (and TestControl) will log all telecommand verification messages reported from SCOS 2000.

5.7 General Pre-Test and Post-Test Procedure

There is a TCL procedure which is executed before every single test procedure which allows to do some settings for every test. It is located in the TOPE installation directory[3] and it is named “GeneralScriptHeader.tcl”. It is already running in the current TestExecution context (see page 9 below) so all *putlog* messages will be recorded in HCSS as well. Any error in the pre-test script will abort the test.

Similar a TCL procedure exists which is executed after every test. It can be used for example to clean up. It is located in the same directory and it is named “GeneralScriptTrailer.tcl”. In this script *putlog* statements have to used carefully because the TestExecution context might be already closed by an explicit

3 In the standard SCOS 2000 V2.3e release this directory is “~/tcl/TOPE”.

statement in the current test procedure. In that case the post-test TCL procedure will be aborted at this *putlog* statement.

6 TestControl --- HCSS Interaction

6.1 TestExecution Objectst

Whenever a test procedure (both stored as local file or stored within the HCSS) is executed a corresponding TestExecution object is created and stored within the HCSS. Each TestExecution object holds following information:

- start time of test execution
- end time of test execution
- the test procedure itself
- instrument model under test
- a list of executed observations within the test execution
- a test log (containing both log message from the user/operator and automatic log messages generated by TOPE/TestControl)
- the final exit message (as string)
- the final exit status (as integer)

All these data can later be retrieved using IA/QLA or other applications.

6.2 TestProcedure Objects

The test procedure (as TCL script) can be stored either as local file or within the HCSS. Local test procedure files are automatically transferred to the HCSS to be stored before the test executions starts. In that way with every execution of a local test procedure file the TestExecution object has a proper reference to the test procedure being used. TestControl does not check whether a local test procedure file already exists in the HCSS. For every run a new copy will be transferred to the HCSS.

Test procedures which are fetched from the HCSS are simply referred by the TestExecution object.

7 Test Procedure Syntax

7.1 Test Procedure Layout

Test procedures are written in the scripting language TCL (version 8.3). In addition the TK extension of TCL is available.

Test procedures are executed within the TOPE/TestControl environment using the TCL statement "source". Therefore test procedure parameters must be specified using special comment lines within the test procedure (see next section). TestControl will then inquire the operator about the test procedure parameters before executing the test procedures. The parameters are available as global variables within the test procedure.

7.2 Test Procedure special comment lines

TestControl will recognize a few special comment lines by special format as follows:

```
# @tagname taginfo
```

Every TCL comment line must start with the "#" character in its first column. The TestControl tags always start with the "@" character which must be separated from the "#" character by some white space. The tagname follows immediate the "@" character. Again separated by white space the additional tag information follows.

TestControl recognizes following tagnames:

tag name	*tag info*	*used already*
author	author's name	no
date	last release date of test procedure as assigned by the editor	no
version	test procedure version as assigned by the editor	no
purpose	single line comment to summary use of test procedure	no
comment	additional comments (this tag may be repeated)	no
param	information about test procedure parameters (this tag must be repeated for each single parameter) The tag info consist of four fields separated by white space: name type default prompt **name**: The name of the global variable to be used within the test procedure to reference this parameter **type**: One of four possible types: boolean integer float string **default**: the default value for the parameter It must be enclosed in quotation marks if it contains white space characters. When there is no default value for this parameter type two consecutive quotation marks. **prompt**: The prompt string used for the query. The prompt can contain white space characters. An example: # @param checking boolean yes Include error checking?	yes

Comment lines with tag names starting with "Tch" must not be modified.

7.3 Test Procedure TCL statements provided by TOPE

TestControl supports all TCL statements provided by TOPE. Please refer to document [R1] for more information about this commands.

Some of the TCL statements supplied by TOPE.

TCL statement	*comment*
tcsend	To uplink a telecommand
fetch	To fetch the value of a TM parameter
subscribe	Subscribe to a TM parameter
waitfor	To wait for a condition of a subscribed TM parameter
waittime	To wait for a time interval
putlog	To log a message

7.4 Starting and closing test procedures

There is no special TCL statement to start a test procedure. All necessary actions are performed by TestControl before the test procedure is actually executed. See section 7.2 on how to provide test procedure parameters.

A test procedure is also closed automatically after the last statement is executed or the TCL statement "return" is encountered. In that case the exit status in the HCSS database will be recorded as "0", the exit message will be "Test procedure finished".

To enter another exit status for the test procedure the “closeTest” command shall be used before the “return” statement. The command “closeTest” takes two arguments, the exit status and an exit message. The exit status shall be a 0 to indicate success or a positive integer number to indicate an error condition detected by the test procedure script. The “return” statement shall return the string “TestControl client: test execution already closed” to avoid the standard exit status and exit message.

The TCL statement "exit" must not be used to terminate a test procedure. It will kill your complete TOPE session.

Any error detected by TCL will also terminate the test procedure. In that case the exit status will be a negative number.

7.5 Logging to the HCSS

There is no special TCL statement for HCSS logging. All log records produced by the TOPE "putlog" statement will be copied to the active TestExecution log.

Committing every single log message to the HCSS database can cause serious delays in the execution of the test. The HCSS TestControl server allows to commit log messages in blocks, see also the HCSS TestControl server user guide. The HCSS log message commits to the database can be controlled in two ways from the EGSE TestControl client.

7.5.1 commitPendingStuff TCL statement

Using the *commitPendingStuff* statement will cause all pending log messages to be committed to the HCSS database. The current drawback is log messages from test executions running in parallel will also be committed.

7.5.2 TCH_COMMIT_IDLE_TIMER Environment Variable

If the time to wait for a TC to be released (specified by the observation) is greater than *TCH_COMMIT_IDLE_TIMER* seconds an implicit call of *commitPendingStuff* will happen causing all pending log messages to be committed (see above).

7.6 Accessing and executing HCSS observations

There two TCL statements to access and execute HCSS observations:

- getObservationCommands
  Returns the telecommand list which will execute the observation
- sendObservationCommands
  Uplinks the telecommand list returned by getObservationCommands

getObservationCommands <observationName> <observationParameters>

Returns the telecommand list from the observation named <observationName> with the actual <observationParameters>. The returned telecommand list must be executed with the sendObservationCommands TCL statement.

<observationName>: The CUS name of the observation mode.

<observationParameters>: A TCL array containing the observation parameter values. The element names of the array elements are the observation parameter names. The <observationParameters> array can be missing if there are no observation parameters.

sendObservationCommands <telecommandlist>

This TCL statement send the telecommands returned by getObservationCommands to there SCOS 2000 uplink system.

<telecommandlist>: The telecommandlist returned by getObservationCommands.

An example:

```
#
# make sure there is nothing left from a previous test execution
# variables are not automatically erased
catch {unset obsParams}
#
# set all observation parameters
#
set obsParams(noScans) 3
set obsParams(stepsProScan) 40
set obsParams(stepWidth) 3
#
# connect to HCSS and get observation telecommands
#
set cmdList [getObservationCommands IntegrationTest obsParams]
#
# and uplink the telecommands
#
```

```
sendObservcationCommands $cmdList
```