Validation Plan for CCS – IEGSE Testsequence Commanding Interface

	Name	Function	Date	Signature
Prepared by	E.Wiezorrek	EM	April 16, 2004	
Checked by				
Approved by				
Approved by				
Approved by				
Authorized by	O. H. Bauer	PM		

0 Preface

0.1 Distribution Record

Intern			Extern		
Department	Name	Qty	Company	Name	Qty
MPE	Bauer				
Electronic Archives at Leuven					

0.2 Document Change Record

Issue / Rev.	Date	Change Notice Number	Modified Pages or Paragraphs	Remarks / Nature of Change
Draft 1.0	04/16/04			Initial version for comments

0.3 Table of Contents

1 Introduction

1.1 Scope

The scope of this document is to describe an acceptance test which validates the IEGSE–CCS testsequence commanding interface.

The validation plan does not describe the installation and configuration of the necessary additional software like SCOS 2000 on both the CCS and IEGSE.

1.2 Concept of Operation

To allow project/industry to manage and schedule instrument tests during IST it is required to deliver predefined instrument test procedures. On the other hand the instrument teams are using HCSS to analyze the instrument tests. HCSS enforces a strong link between uplink (TC) and downlink (TM) and therefore puts some "markers" in the uplink to ensure this. So the instrument test procedure cannot be predefined in total. The variable parts of the instrument test procedure are

TC user identification (needed for HCSS TC history tracing),
TC parameters definition,
TC delta times between the TCs.

The IEGSE–CCS testsequence command interface shall be the means to solve this issue.

The instrument test procedure is executed on the CCS. After startup it will send a TC to the IEGSE telling it which test observation (which test) will be executed. The IEGSE will answer with a set of TM packets which transfer additional TC information back to CCS. The test procedures reads those TM packets and packs the additional TC information to the TCs actually send to the instrument.

1.3 Applicable Documents

1.4 Reference Documents

1.5 Acronyms

CCS	Central Checkout Systems
HCSS	Herschel Common Science System
HK	HouseKeeping
HPSDB	Herschel/Plank System DataBase
IEGSE	Instrument Electrical Ground Support Equipment
SCOS	Spacecraft Control & Operating Systems
TCL	Tool Command Language
TOPE	Test and Operations Procedure Environment

2 Test Setup

2.1 CCS

The CCS shall operate an installed and configured SCOS 2000 system including TOPE. Part of the

configuration is the integration of MIB files describing IEGSE TM and TC packets. These MIB files will be delivered well in advance to allow integration in the HPSDB and transfer to the CCS system.

Another part of the configuration will be the integration of TOPE test procedures used for this validation. Again there will be delivered in advance but a possibility to update them is expected.

No specific software must be installed on the CCS.

2.2 IEGSE

The IEGSE consists of a single PC running both the SCOS 2000/TOPE (V2.3e P5) system and HCSS (build TBD) although for this validation only the HCSS is necessary.

Before running the validation procedures the HCSS property file must be updated to reflect the network setup used for the validation.

As there will be no instrument available during the validation a specific TestClient application will be used on the IEGSE which accepts TCs and delivers TM HK packets.

2.3 Network

Both machines CSS and IEGSE must be configured for communication via the LAN.

3 Validation Procedures

3.1 Checking Network Connection

This procedure shall validate the LAN setup.

Step nr.	*Procedure*	*Expected Result*	*Check*
1	Using an CCS terminal window execute the “ping” command to check TCP/IP connection from CCS to IEGSE	Positive response	□
2	Using an IEGSE terminal window execute the “ping” command to check TCP/IP connection from IEGSE to CCS	Positive response	□

3.2 Establishing PIPE connection

This procedure shall validate the PIPE setup.

Step nr.	*Procedure*	*Expected Result*	*Check*
1	On CCS, start the PIPE server		□
2	On IEGSE, start the PIPE client		□

Step nr.	*Procedure*	*Expected Result*	*Check*
3	On CCS, check connection status	IEGSE status: connected	□
4	Wait at least 60 seconds, then check connection status again	IEGSE status: connected	□

3.3 Receiving IEGSE TM Packets

This procedure shall validate the IEGSE to CCS SCOS2000 connection.

Step nr.	*Procedure*	*Expected Result*	*Check*
1	On IEGSE, start the Router process		□
2	On CCS, start a SCOS 2000 TM desktop application and select the IEGSE alphanumeric display		□
3	On IEGSE, start the CcsProcedureHandler (A number of IEGSE TM packets will be generated and transmitted during startup)		□
4	On CCS, verify update of alphanumeric display	Updated display	□
5	On IEGSE, start the TestClient application. It will be used for the following validation procedures.		□

Note: Don't stop both the Router, the CcsProcedureHandler and the TestClient application on IEGSE as they are needed for the following procedures.

3.4 Executing Short Standard Test Procedure

This procedure already validates the complete interface. The instrument test procedure contains only a few TC to the instrument so the whole procedure will finish in a short time scale.

Note: On IEGSE, both the Router, the CcsProcedureHandler and the TestClient shall still be running. They are started as part of the validation procedure 3.3 “Receiving IEGSE TM packets”

Step nr.	*Procedure*	*Expected Result*	*Check*
1	On IEGSE, verify the Router application is still running		□
2	On IEGSE, verify the CcsProcedureHandler application is still running		□

Step nr.	*Procedure*	*Expected Result*	*Check*
3	On CCS, using TOPE start the TCL procedure "ICVshortStandard.tcl"		□
4	On IEGSE, check output of the CcsProcedureHandler console window	Verify the reception of the TC and the transmission of the TM packets	□
5	On IEGSE, check output of the TestClient console window.	Verify the reception of the "instrument" commands.	□
6	On CCS, verify the TOPE TCL procedure ends with success	The TCL procedure ends with a non error message	□

3.6 Executing Standard Test Procedure (102 Telecommands)

This procedure validates the complete interface. The instrument test procedure contains 102 instrument TCs.

Note: On IEGSE, both the Router, the CcsProcedureHandler and the TestClient shall still be running. They are started as part of the validation procedure 3.3 "Receiving IEGSE TM packets"

Step nr.	*Procedure*	*Expected Result*	*Check*
1	On IEGSE, verify the Router application is still running		□
2	On IEGSE, verify the CcsProcedureHandler application is still running		□
3	On CCS, using TOPE start the TCL procedure "ICVStandard-102.tcl"		□
4	On IEGSE, check output of the CcsProcedureHandler console window	Verify the reception of the TC and the transmission of the TM packets	□
5	On IEGSE, check output of the TestClient console window.	Verify the reception of the "instrument" commands.	□
6	On CCS, verify the TOPE TCL procedure ends with success	The TCL procedure ends with a non error message	□

3.8 Executing Standard Test Procedure (202 Telecommands)

This procedure validates the complete interface. The instrument test procedure contains 202 instrument TCs to increase the load on the system.

Note: On IEGSE, both the Router, the CcsProcedureHandler and the TestClient shall still be running. They are started as part of the validation procedure 3.3 "Receiving IEGSE TM packets"

Step nr.	*Procedure*	*Expected Result*	*Check*
1	On IEGSE, verify the Router application is still running		□
2	On IEGSE, verify the CcsProcedureHandler application is still running		□
3	On CCS, using TOPE start the TCL procedure "ICVStandard-202.tcl"		□
4	On IEGSE, check output of the CcsProcedureHandler console window	Verify the reception of the TC and the transmission of the TM packets	□
5	On IEGSE, check output of the TestClient console window.	Verify the reception of the "instrument" commands.	□
6	On CCS, verify the TOPE TCL procedure ends with success	The TCL procedure ends with a non error message	□

3.10 Executing Standard Test Procedure (302 Telecommands)

This procedure validates the complete interface. The instrument test procedure contains 202 instrument TCs to increase the load on the system.

Note: On IEGSE, both the Router, the CcsProcedureHandler and the TestClient shall still be running. They are started as part of the validation procedure 3.3 "Receiving IEGSE TM packets"

Step nr.	*Procedure*	*Expected Result*	*Check*
1	On IEGSE, verify the Router application is still running		□
2	On IEGSE, verify the CcsProcedureHandler application is still running		□
3	On CCS, using TOPE start the TCL procedure "ICVStandard-302.tcl"		□
4	On IEGSE, check output of the CcsProcedureHandler console window	Verify the reception of the TC and the transmission of the TM packets	□
5	On IEGSE, check output of the TestClient console window.	Verify the reception of the "instrument" commands.	□
6	On CCS, verify the TOPE TCL procedure ends with success	The TCL procedure ends with a non error message	□

4 Test Procedure Exampe

Here is the “ICVshortStandard.tcl” file as an example for the test procedures which are used for this validation.

```
#
# This script is generated by HCSS software. There are no
# user maintainable parts in it.
# Breaking the seals stops warranty.
#
proc putLog {msg} {
    putlog "** $msg"
}

proc putError {msg} {
    putlog "**! $msg"
    uplevel return
}

proc setTimeMarker {} {
    return [clock clicks -milliseconds]
}

proc waitSinceMarker {timeMarker milliSecondsToWait} {
    set currentTime [clock clicks -milliseconds]
    set timeToWait [expr $timeMarker + $milliSecondsToWait - $currentTime]
    putLog "Time to wait for TC since start of sequence: $milliSecondsToWait
milliseconds, actual time to wait: $timeToWait milliseconds"
    update
    if {$timeToWait > 0} {
        waittime [expr $timeToWait / 1000.]
    }
}
proc hex2string {hexString} {
        set numberOfChars [string length $hexString]
        for {set ix 0} {$ix < $numberOfChars} {incr ix 2} {
                set subStr [string range $hexString $ix [expr $ix + 1]]
                scan $subStr %2x intVal
                append retVal [format %c $intVal]
        }
        return $retVal
}

proc fetchParameter {name CCS_IEGSE_timeStamp} {
    set var [fetch $name]
    set timeStamp [gettimestamp $var]
    while {$timeStamp < $CCS_IEGSE_timeStamp} {
      waittime 5.0
      putlog "oops, new fetch necessary"
      set var [fetch $name]
      set timeStamp [gettimestamp $var]
    }
    return $var
}
```

```
proc getTcName {cnt CCS_IEGSE_timeStamp} {
    set YMname [format "YM%3.3X964" [expr 0x000 | $cnt]]
    return [string trimright [hex2string [getrawvalue [fetchParameter $YMname
$CCS_IEGSE_timeStamp]]]]
}

proc getWaitTime {cnt CCS_IEGSE_timeStamp} {
    set YMname [format "YM%3.3X964" [expr 0x800 | $cnt]]
    return [getrawvalue [fetchParameter $YMname $CCS_IEGSE_timeStamp]]
}

proc getTcID {cnt CCS_IEGSE_timeStamp} {
    set YMname [format "YM%3.3X964" [expr 0x400 | $cnt]]
    return [getrawvalue [fetchParameter $YMname $CCS_IEGSE_timeStamp]]
}

proc getTcParams {cnt CCS_IEGSE_timeStamp} {
    set YMname [format "YM%3.3X964" [expr 0xC00 | $cnt]]
    return [string trimright [hex2string [getrawvalue [fetchParameter $YMname
$CCS_IEGSE_timeStamp]]]]
}

#
# Number of instrument TCs in this script
#
set numberOfTcs 3;

#
# touch all required TM packets
#   (seems necessary for initial request for EXIF_TM{1|2|3}
#
#putLog "Touch all necessary TM packets"
#for {set ix 0} {$ix < $numberOfTcs} {incr ix} {
#    set YMname [format "YM%3.3X964" [expr 0x400 | $ix]]
#    fetch $YMname
#    update
#}
#fetch YM3FF964
#fetch YM7FF964
#fetch YMBFF964

set CCS_IEGSE_timeStamp [clock format [clock seconds] -format "%Y.%j.%H.%M.%S"
-gmt 1 ]
putLog "Get time Stamp: $CCS_IEGSE_timeStamp"
#
# trigger observation generation by sending a telecommand packet to EGSE
#
tcsend YC000964 nowait {YP000964 "IntegrationTest;"} \
    {YP001964 2} \
    {YP002964 "numberOfBb=0;"} \
    {YP002964 "dummy=null;"} \
```

```
#
# wait for TM parameter YM3FF964 which is in the last TM packet
#
putLog "Waiting for IEGSE TM packets"

waittime 5.0
set ObsName [fetch YM3FF964]
set timeStamp [gettimestamp $ObsName]
while {$timeStamp < $CCS_IEGSE_timeStamp} {
    putlog "$timeStamp > $CCS_IEGSE_timeStamp"
    waittime 5.0
    set ObsName [fetch YM3FF964]
    set timeStamp [gettimestamp $ObsName]
}
putLog "Got all TM packets for ObservationMode [string trimright [hex2string
[getrawvalue $ObsName]]]"
update

#
# check number of received TCinfo packets
#
putLog "Check number of returned TC infos"
set varYMBFF964 [fetch YMBFF964]
set timeStamp [gettimestamp $varYMBFF964]
while {$timeStamp < $CCS_IEGSE_timeStamp} {
    waittime 5.0
    set varYMBFF964 [fetch YMBFF964]
    set timeStamp [gettimestamp $varYMBFF964]
}

set rcvNoTcs [getrawvalue $varYMBFF964]
if {$rcvNoTcs != $numberOfTcs} {
    putError "Number of received TCinfos ($rcvNoTcs) does not match expected
count ($numberOfTcs)"
}
update

#
# get all TC data
#
putLog "Retrieve TCinfo data"
for {set ix 0} {$ix < $numberOfTcs} {incr ix} {
    set tcName_$ix [getTcName $ix $CCS_IEGSE_timeStamp]
    set waitTime_$ix [getWaitTime $ix $CCS_IEGSE_timeStamp]
    set userId_$ix [getTcID $ix $CCS_IEGSE_timeStamp]
    set tcParams_$ix [getTcParams $ix $CCS_IEGSE_timeStamp]
    update
}
putLog "Got all TC info data"
#
# check TC names in TCinfo packets
#
 if { \
  $tcName_0 == "ZCOBSID" && \
```

```
  $tcName_1 == "ZCOBSID" && \
  $tcName_2 == "ZCBBID" && \
1} {} else {putError "improper Telecommand name sequence"}

#
# reset timeMarker
#
set timeMarker [setTimeMarker]

putLog "Start of TC sequence"

#
# Telecommand sequence to be executed
#
waitSinceMarker $timeMarker $waitTime_0
eval tcsend ZCOBSID nowait userrequestid $userId_0 $tcParams_0
waitSinceMarker $timeMarker $waitTime_1
eval tcsend ZCOBSID nowait userrequestid $userId_1 $tcParams_1
waitSinceMarker $timeMarker $waitTime_2
eval tcsend ZCBBID nowait userrequestid $userId_2 $tcParams_2
#
# a typical example of the telecommand sequence
#
set dummy {
waitsincemarker 0
tcsend ZCOBSID nowait userrequestid 95631 {ZPFNCTID "OBS ID" ENG} {ZPOBSID
"536872066" RAW}
waitsincemarker 1000
tcsend ZCOBSID nowait userrequestid 95632 {ZPFNCTID "OBS ID" ENG} {ZPOBSID
"536870912" RAW}
waitsincemarker 1000
tcsend ZCBBID nowait userrequestid 95633 {ZPFNCTID "BB ID" ENG} {ZPBBID
"1073741824" RAW}
}
set dummy {}
```