



**Herschel
PACS**

Validation Plan for IEGSE–
CCS Testsequence
command interface

Reference: PICC-ME-xx-yyy
Issue: Draft 2
Date: July 27, 2004
Page: 1 of 14

Validation Plan for CCS – IEGSE Testsequence Commanding Interface

	Name	Function	Date	Signature
Prepared by	E. Wieszorrek	EM	July 27, 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
Draft 2.0	07/27/04		Section 2	New description of the test setup
			Section 3.2	IEGSE is the PIPE server, CCS is the PIPE client add use of IEGSE PIPE indicators
			Section 3.3	add use of IEGSE PIPE indicators
			Section 3.4	specify a minimum of TC parameters as well.
			Section 3.6 ...	Section numbering corrected

0.3 Table of Contents

0	Preface.....	2
0.1	Distribution Record.....	2
0.2	Document Change Record.....	2
0.3	Table of Contents.....	3
1	Introduction.....	4
1.1	Scope.....	4
1.2	Concept of Operation.....	4
1.3	Applicable Documents.....	4
1.4	Reference Documents.....	4
1.5	Acronyms.....	4
2	Test Setup.....	5
2.1	System Overview.....	5
2.2	Description of Procedure Steps.....	6
3	Validation Procedures.....	6
3.1	Checking Network Connection.....	6
3.2	Establishing PIPE connection.....	7
3.3	Receiving IEGSE TM Packets.....	7
3.4	Executing Short Standard Test Procedure.....	8
3.5	Executing Standard Test Procedure (102 Telecommands).....	8
3.6	Executing Standard Test Procedure (202 Telecommands).....	9
3.7	Executing Standard Test Procedure (302 Telecommands).....	9
4	Test Procedure Example.....	11

	Herschel PACS	Validation Plan for IEGSE– CCS Testsequence command interface	Reference: PICC-ME-xx-yyy Issue: Draft 2 Date: July 27, 2004 Page: 4 of 14
---	--------------------------	--	---

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 for the instrument test procedure 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

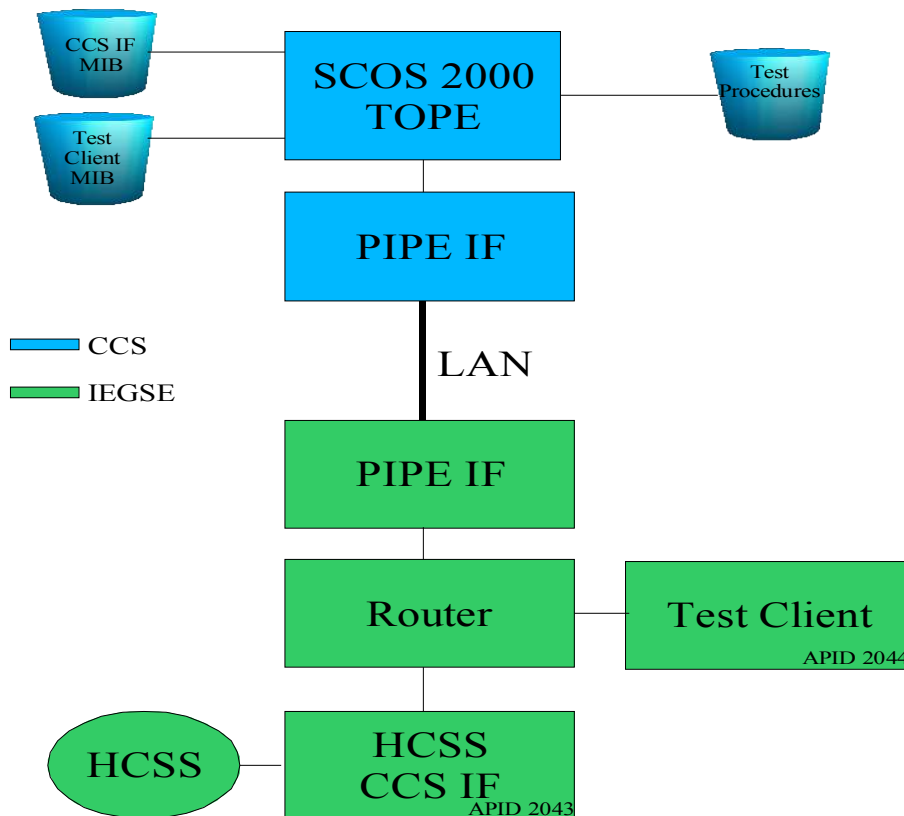
1.5 Acronyms

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 System Overview



2.1.1 CCS

The CCS shall operate an installed and configured SCOS 2000 system including TOPE. Part of the configuration is the integration of two set of MIB files, one describing TM and TC packets to support the CCS testsequence interface, the other to support a test client used only for this validation. The 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.

The CCS must be configured in a way that TC packets with APID values of 2043 (PACS IEGSE) and 2044 (SPIRE IEGSE) are routed to the IEGSE using the PIPE interface via the IS LAN not using the CDMU DFE.

No specific software must be installed on the CCS.



2.1.2 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 which accepts TCs will be used on the IEGSE. This TestClient listens to a specific APID allocated for the IEGSE.

2.1.3 Network

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

2.2 Description of Procedure Steps

A typical validation procedure starts with the execution of a TOPE test procedure running at the CCS. The test procedure will start with the request for a TC (describing the observation mode to be executed with its parameters) to be sent to the IEGSE CCS interface. Once the IEGSE received this TC (via CCS-SCOS 2000, PIPE interface) it will be routed to the (IEGSGE) HCSS CCS IF task.

This application will schedule the observation within HCSS. The resulting telecommand parameters to be used for the observation will be packed in TM packets which are send back to the CCS. The CCS-SCOS 2000 system will receive the TM packets and pass them to the CCS TOPE environment. Via TOPE the original test procedure can read the telecommand parameters and construct the final TC sequence.

Note: The TM packets send from the IEGSE to the CCS are not sent periodically but on request from the CCS (triggered by the test procedure execution).


In the final environment this TC sequence will be sent to the HERSCHEL instruments. As they are not available for this validation the TC sequence will send to the IEGSE (selected by APID), again via the CCS-SCOS2000/TOPE, PIPE interface). This time the TCs are received by the TestClient application which simply will display the TCs as they arrive.

3 Validation Procedures

3.1 Checking Network Connection

This procedure shall validate the LAN setup.

<i>Step nr.</i>	<i>Procedure</i>	<i>Expected Result</i>	<i>Check</i>
1	Using an CCS terminal window execute the “ping” command to check TCP/IP connection from CCS to IEGSE	Positive response	<input type="checkbox"/>

	Herschel PACS	Validation Plan for IEGSE– CCS Testsequence command interface	Reference: PICC-ME-xx-yyy Issue: Draft 2 Date: July 27, 2004 Page: 7 of 14
---	--------------------------	--	---

<i>Step nr.</i>	<i>Procedure</i>	<i>Expected Result</i>	<i>Check</i>
2	Using an IEGSE terminal window execute the “ping” command to check TCP/IP connection from IEGSE to CCS	Positive response	<input type="checkbox"/>

3.2 Establishing PIPE connection

This procedure shall validate the PIPE setup.

<i>Step nr.</i>	<i>Procedure</i>	<i>Expected Result</i>	<i>Check</i>
1	On IEGSE, start the PIPE server		<input type="checkbox"/>
2	On CCS, start the PIPE client		<input type="checkbox"/>
3	On IEGSE, check connection status	PIPE-GW: CCS pane shall turn green	<input type="checkbox"/>
4	On CCS, check connection status	IEGSE status: connected	<input type="checkbox"/>
5	Wait at least 60 seconds, then check connection status again	IEGSE status: connected	<input type="checkbox"/>

3.3 Receiving IEGSE TM Packets

This procedure shall validate the IEGSE to CCS SCOS2000 connection.

<i>Step nr.</i>	<i>Procedure</i>	<i>Expected Result</i>	<i>Check</i>
1	On IEGSE, start the Router process	PIPE-GW: Router pane shall turn green	<input type="checkbox"/>
2	On CCS, start a SCOS 2000 TM desktop application and select the IEGSE alphanumeric display		<input type="checkbox"/>
3	On IEGSE, start the CcsProcedureHandler (A number of IEGSE TM packets will be generated and transmitted during startup)		<input type="checkbox"/>
4	On CCS, verify update of alphanumeric display	Updated display	<input type="checkbox"/>
5	On IEGSE, start the TestClient application. It will be used for the following validation procedures.		<input type="checkbox"/>

Note: Don't stop 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.

The TC used in the test procedure shall contain up to 50 parameters.

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”

<i>Step nr.</i>	<i>Procedure</i>	<i>Expected Result</i>	<i>Check</i>
1	On IEGSE, verify the Router application is still running		<input type="checkbox"/>
2	On IEGSE, verify the CcsProcedureHandler application is still running		<input type="checkbox"/>
3	On CCS, using TOPE start the TCL procedure “ICVshortStandard.tcl”		<input type="checkbox"/>
4	On IEGSE, check output of the CcsProcedureHandler console window	Verify the reception of the TC and the transmission of the TM packets	<input type="checkbox"/>
5	On IEGSE, check output of the TestClient console window.	Verify the reception of the “instrument” commands.	<input type="checkbox"/>
6	On CCS, verify the TOPE TCL procedure ends with success	The TCL procedure ends with a non error message	<input type="checkbox"/>

3.5 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”

<i>Step nr.</i>	<i>Procedure</i>	<i>Expected Result</i>	<i>Check</i>
1	On IEGSE, verify the Router application is still running		<input type="checkbox"/>
2	On IEGSE, verify the CcsProcedureHandler application is still running		<input type="checkbox"/>
3	On CCS, using TOPE start the TCL procedure “ICVStandard-102.tcl”		<input type="checkbox"/>
4	On IEGSE, check output of the CcsProcedureHandler console window	Verify the reception of the TC and the transmission of the TM packets	<input type="checkbox"/>
5	On IEGSE, check output of the TestClient console window.	Verify the reception of the “instrument” commands.	<input type="checkbox"/>

<i>Step nr.</i>	<i>Procedure</i>	<i>Expected Result</i>	<i>Check</i>
6	On CCS, verify the TOPE TCL procedure ends with success	The TCL procedure ends with a non error message	<input type="checkbox"/>

3.6 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”

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

3.7 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”

<i>Step nr.</i>	<i>Procedure</i>	<i>Expected Result</i>	<i>Check</i>
1	On IEGSE, verify the Router application is still running		<input type="checkbox"/>
2	On IEGSE, verify the CcsProcedureHandler application is still running		<input type="checkbox"/>
3	On CCS, using TOPE start the TCL procedure “ICVStandard-302.tcl”		<input type="checkbox"/>



Herschel PACS

Validation Plan for IEGSE–
CCS Testsequence
command interface

Reference: PICC-ME-xx-yyy
Issue: Draft 2
Date: July 27, 2004
Page: 10 of 14

<i>Step nr.</i>	<i>Procedure</i>	<i>Expected Result</i>	<i>Check</i>
4	On IEGSE, check output of the CcsProcedureHandler console window	Verify the reception of the TC and the transmission of the TM packets	<input type="checkbox"/>
5	On IEGSE, check output of the TestClient console window.	Verify the reception of the “instrument” commands.	<input type="checkbox"/>
6	On CCS, verify the TOPE TCL procedure ends with success	The TCL procedure ends with a non error message	<input type="checkbox"/>



4 Test Procedure Example

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} {ZPBID  
"1073741824" RAW}  
}  
set dummy {}
```