

# SPS Software Test Cases

## PFS-SPS-LAMPRC1072-01 \*

Arnaud Le Fur, Hassan Siddiqui

October 23, 2019

## Contents

<b>1</b>	<b>Version and Changelog</b>	<b>6</b>
<b>2</b>	<b>Introduction</b>	<b>6</b>
<b>3</b>	<b>Test Case SPS-ALERT-TRIGGER-010: Alerts trigger mechanism</b>	<b>7</b>
3.1	Description . . . . .	7
3.2	Pass/Fail Condition(s) . . . . .	7
3.3	Hardware constraints . . . . .	7
3.4	Initial conditions . . . . .	7
3.5	Procedure . . . . .	7
3.6	Additional Notes . . . . .	7
<b>4</b>	<b>Test Case SPS-ALERT-INVALID-020: Alerts invalid values mechanism</b>	<b>8</b>
4.1	Description . . . . .	8
4.2	Pass/Fail Condition(s) . . . . .	8
4.3	Hardware constraints . . . . .	8
4.4	Initial conditions . . . . .	8
4.5	Procedure . . . . .	8
4.6	Additional Notes . . . . .	8
<b>5</b>	<b>Test Case SPS-ALERT-TIMEOUT-030: Alerts values out-of-date</b>	<b>9</b>
5.1	Description . . . . .	9
5.2	Pass/Fail Condition(s) . . . . .	9
5.3	Hardware constraints . . . . .	9
5.4	Initial conditions . . . . .	9
5.5	Procedure . . . . .	9
5.6	Additional Notes . . . . .	9

---

\*Previously document code assigned was PFS-SPS-PRU300005

<b>6</b>	<b>Test Case SPS-XCU-PCM-040: Cryostat power control module</b>	<b>10</b>
6.1	Description . . . . .	10
6.2	Pass/Fail Condition(s) . . . . .	10
6.3	Hardware constraints . . . . .	10
6.4	Initial conditions . . . . .	10
6.5	Procedure . . . . .	10
6.6	Additional Notes . . . . .	10
<b>7</b>	<b>Test Case SPS-XCU-GATEVALVE-050: Cryostat Gatevalve</b>	<b>11</b>
7.1	Description . . . . .	11
7.2	Pass/Fail Condition(s) . . . . .	11
7.3	Hardware constraints . . . . .	11
7.4	Initial conditions . . . . .	11
7.5	Procedure . . . . .	11
7.6	Additional Notes . . . . .	11
<b>8</b>	<b>Test Case SPS-XCU-TURBO-060: Cryostat Turbo Pump</b>	<b>12</b>
8.1	Description . . . . .	12
8.2	Pass/Fail Condition(s) . . . . .	12
8.3	Hardware constraints . . . . .	12
8.4	Initial conditions . . . . .	12
8.5	Procedure . . . . .	12
8.6	Additional Notes . . . . .	12
<b>9</b>	<b>Test Case SPS-XCU-IONPUMP-070: Cryostat Ion Pumps</b>	<b>13</b>
9.1	Description . . . . .	13
9.2	Pass/Fail Condition(s) . . . . .	13
9.3	Hardware constraints . . . . .	13
9.4	Initial conditions . . . . .	13
9.5	Procedure . . . . .	13
9.6	Additional Notes . . . . .	13
<b>10</b>	<b>Test Case SPS-XCU-GAUGE-080: Cryostat Gauge</b>	<b>14</b>
10.1	Description . . . . .	14
10.2	Pass/Fail Condition(s) . . . . .	14
10.3	Hardware constraints . . . . .	14
10.4	Initial conditions . . . . .	14
10.5	Procedure . . . . .	14
10.6	Additional Notes . . . . .	14
<b>11</b>	<b>Test Case SPS-XCU-COOLER-090: Cryostat Cooler</b>	<b>15</b>
11.1	Description . . . . .	15
11.2	Pass/Fail Condition(s) . . . . .	15
11.3	Hardware constraints . . . . .	15
11.4	Initial conditions . . . . .	15

11.5	Procedure . . . . .	15
11.6	Additional Notes . . . . .	15
<b>12</b>	<b>Test Case SPS-XCU-TEMPS-100: Cryostat Temps</b>	<b>16</b>
12.1	Description . . . . .	16
12.2	Pass/Fail Condition(s) . . . . .	16
12.3	Hardware constraints . . . . .	16
12.4	Initial conditions . . . . .	16
12.5	Procedure . . . . .	16
12.6	Additional Notes . . . . .	16
<b>13</b>	<b>Test Case SPS-XCU-HEATERS-110: Cryostat Heaters</b>	<b>17</b>
13.1	Description . . . . .	17
13.2	Pass/Fail Condition(s) . . . . .	17
13.3	Hardware constraints . . . . .	17
13.4	Initial conditions . . . . .	17
13.5	Procedure . . . . .	17
13.6	Additional Notes . . . . .	17
<b>14</b>	<b>Test Case SPS-LOG-120: Process logging</b>	<b>18</b>
14.1	Description . . . . .	18
14.2	Pass/Fail Condition(s) . . . . .	18
14.3	Hardware constraints . . . . .	18
14.4	Initial conditions . . . . .	18
14.5	Procedure . . . . .	18
14.6	Additional Notes . . . . .	18
<b>15</b>	<b>Test Case SPS-IO-130: Test File I/O</b>	<b>19</b>
15.1	Description . . . . .	19
15.2	Pass/Fail Condition(s) . . . . .	19
15.3	Hardware constraints . . . . .	19
15.4	Initial conditions . . . . .	19
15.5	Procedure . . . . .	19
15.6	Additional Notes . . . . .	19
<b>16</b>	<b>Test Case SPS-BIAS-140: Detector Bias</b>	<b>20</b>
16.1	Description . . . . .	20
16.2	Pass/Fail Condition(s) . . . . .	20
16.3	Hardware constraints . . . . .	20
16.4	Initial conditions . . . . .	20
16.5	Procedure . . . . .	20
16.6	Additional Notes . . . . .	20

<b>17 Test Case SPS-DARK-150: Detector Dark</b>	<b>21</b>
17.1 Description . . . . .	21
17.2 Pass/Fail Condition(s) . . . . .	21
17.3 Hardware constraints . . . . .	21
17.4 Initial conditions . . . . .	21
17.5 Procedure . . . . .	21
17.6 Additional Notes . . . . .	21
<b>18 Test Case SPS-SHUTTER-160: Shutter control</b>	<b>22</b>
18.1 Description . . . . .	22
18.2 Pass/Fail Condition(s) . . . . .	22
18.3 Hardware constraints . . . . .	22
18.4 Initial conditions . . . . .	22
18.5 Procedure . . . . .	22
18.6 Additional Notes . . . . .	22
<b>19 Test Case SPS-SLIT-170: Slit movement</b>	<b>23</b>
19.1 Description . . . . .	23
19.2 Pass/Fail Condition(s) . . . . .	23
19.3 Hardware constraints . . . . .	23
19.4 Initial conditions . . . . .	23
19.5 Procedure . . . . .	23
19.6 Additional Notes . . . . .	23
<b>20 Test Case SPS-REXM-180: Red exchange mechanism movement</b>	<b>24</b>
20.1 Description . . . . .	24
20.2 Pass/Fail Condition(s) . . . . .	24
20.3 Hardware constraints . . . . .	24
20.4 Initial conditions . . . . .	24
20.5 Procedure . . . . .	24
20.6 Additional Notes . . . . .	24
<b>21 Test Case SPS-BIA-190: Check Back Illumination Assembly Photoresistance</b>	<b>25</b>
21.1 Description . . . . .	25
21.2 Pass/Fail Condition(s) . . . . .	25
21.3 Hardware constraints . . . . .	25
21.4 Initial conditions . . . . .	25
21.5 Procedure . . . . .	25
21.6 Additional Notes . . . . .	25
<b>22 Test Case SPS-TEMPS-200: Check SPS Temperature</b>	<b>26</b>
22.1 Description . . . . .	26
22.2 Pass/Fail Condition(s) . . . . .	26
22.3 Hardware constraints . . . . .	26

22.4	Initial conditions . . . . .	26
22.5	Procedure . . . . .	26
<b>23</b>	<b>Test Case SPS-IIS-210: Check Internal Illumination Sources</b>	<b>27</b>
23.1	Description . . . . .	27
23.2	Pass/Fail Condition(s) . . . . .	27
23.3	Hardware constraints . . . . .	27
23.4	Initial conditions . . . . .	27
23.5	Procedure . . . . .	27
23.6	Additional Notes . . . . .	27

# 1 Version and Changelog

The version of this document is **0.3** .

Version	Date	Author	Description
0.4	2019-10-23	A Le Fur	INSTRM-799 : add interlock test
0.3	2019-10-21	H Siddiqui	Assigned new doc code PFS-SPS-LAMPRC1072
0.2	2019-10-16	A Le Fur	Updated test cases
0.1	2019-07-10	H Siddiqui	First version

## 2 Introduction

This document describes the test cases used for demonstrating the validity of the **ICS** software used for SPS commanding, prior to delivery of the SM1 module to Subaru at the end of 2019.

The intention is for these test cases to be run at during the summer of 2019. A report of the outcome would then be provided to the Project Office, who would then decide whether the software is of an adequate level for use at Subaru.

## 3 Test Case SPS-ALERT-TRIGGER-010: Alerts trigger mechanism

### 3.1 Description

Checks an expected alarm is raised using different alert mechanisms :

1. LIMITS ALERT : value within bounds.
2. REGEXP ALERT : regular expression.
3. Callback : dedicated python function.

Dedicated testsActor keywords are tracked by alertsActor. A command is sent to testsActor to generate out of range values and therefore trigger the alerts.

### 3.2 Pass/Fail Condition(s)

**Pass** Expected alerts seen on STS server side and written to log when the alerts are triggered.

1. an expected alert is raised when all keytest1 values are not within 160-166
2. an expected alert is raised when keytest2 value is above 1.0e-6
3. an expected alert is raised when keytest3[1] is not "OK"

**Fail** no alerts are raised .

### 3.3 Hardware constraints

None.

### 3.4 Initial conditions

alertsActor and testsActor started

### 3.5 Procedure

1. AlertsActor : connect tests controller: `alerts connect controller=tests`
2. wait 30 seconds, keywords are generated within ranges, no alerts are raised yet
3. TestsActor : trigger the alerts: `tests alerts trigger`
4. AlertsActor : disconnect tests controller: `alerts disconnect controller=tests`

### 3.6 Additional Notes

None.

## 4 Test Case SPS-ALERT-INVALID-020: Alerts invalid values mechanism

### 4.1 Description

Checks that an expected alarm is raised when an invalid value is generated. Since dedicated testsActor keywords are tracked by alertsActor, a command is sent to testsActor to generate invalid values and therefore trigger the alerts.

### 4.2 Pass/Fail Condition(s)

**Pass** Expected alerts seen on STS server side and written to log when the alerts are triggered.

1. an expected alert is raised when at least one keytest1 field value is invalid
2. an expected alert is raised when keytest2 value is invalid

**Fail** no alerts are raised

### 4.3 Hardware constraints

None.

### 4.4 Initial conditions

alertsActor and testsActor started

### 4.5 Procedure

1. AlertsActor : connect tests controller: `alerts connect controller=tests`
2. wait 30 seconds, keywords are generated within ranges, no alerts are raised yet
3. TestsActor : trigger the alerts: `tests alerts invalid`
4. AlertsActor : disconnect tests controller: `alerts disconnect controller=tests`

### 4.6 Additional Notes

None.



## 5 Test Case SPS-ALERT-TIMEOUT-030: Alerts values out-of-date

### 5.1 Description

Checks that an expected alarm is raised when keyword values are no longer generated with `TIMELIM=180 secs`. Since dedicated testsActor keywords are tracked by alertsActor, a command is sent to testsActor to stop to keyword generation and therefore trigger the alerts.

### 5.2 Pass/Fail Condition(s)

**Pass** Expected alerts seen on STS server side and written to log when the alerts are triggered.

1. an expected TIMEOUT alert is raised after 180 seconds

**Fail** no alerts are raised

### 5.3 Hardware constraints

None.

### 5.4 Initial conditions

alertsActor and testsActor started

### 5.5 Procedure

1. AlertsActor : connect tests controller: `alerts connect controller=tests`
2. wait 30 seconds, keywords are generated within ranges, no alerts are raised yet
3. TestsActor : trigger the alerts: `tests alerts timeout`
4. wait 180 seconds, keywords are no longer generated, alerts should be raised
5. AlertsActor : disconnect tests controller: `alerts disconnect controller=tests`

### 5.6 Additional Notes

None.

## 6 Test Case SPS-XCU-PCM-040: Cryostat power control module

### 6.1 Description

Check the communication with the PCM board of a given camera, check that the inputPower voltages and status are correct. Retrieve and generate volt, current, power for each PCM channel.

### 6.2 Pass/Fail Condition(s)

**Pass** PCM dataset will be generated and command should finish with ‘test=power-cam,OK’

**Fail** If an exception is raised ‘test=power-cam,FAILED’ will be generated with its full trace.

### 6.3 Hardware constraints

1. SPS RACK powered up
2. BEE powered up.

### 6.4 Initial conditions

xcuActor and testsActor started

### 6.5 Procedure

1. testsActor : send power test command: `tests power cam=r1` .
2. wait for the command to end.

### 6.6 Additional Notes

None.

## 7 Test Case SPS-XCU-GATEVALVE-050: Cryostat Gatevalve

### 7.1 Description

Check the communication with the gatevalve of a given camera, check that the statuses are correct. Retrieve and generate gatevalve and interlock status.

### 7.2 Pass/Fail Condition(s)

**Pass** Gatevalve dataset will be generated and command should finish with ‘`test=gatevalve-cam,OK`’

**Fail** If an exception is raised ‘`test=gatevalve-cam,FAILED`’ will be generated with its full trace.

### 7.3 Hardware constraints

1. SPS RACK powered up
2. BEE powered up.

### 7.4 Initial conditions

xcuActor and testsActor started

### 7.5 Procedure

1. testsActor : send gatevalve test command: `tests gatevalve cam=r1` .
2. wait for the command to end.

### 7.6 Additional Notes

None.

## 8 Test Case SPS-XCU-TURBO-060: Cryostat Turbo Pump

### 8.1 Description

Check the communication with the turbo of a given camera, check that the statuses are correct. Retrieve and generate turbo telemetry.

### 8.2 Pass/Fail Condition(s)

**Pass** Turbo dataset will be generated and command should finish with ‘`test=turbo-cam,OK`’

**Fail** If an exception is raised ‘`test=turbo-cam,FAILED`’ will be generated with its full trace.

### 8.3 Hardware constraints

1. SPS RACK powered up
2. BEE powered up.

### 8.4 Initial conditions

xcuActor and testsActor started

### 8.5 Procedure

1. testsActor : send turbo test command: `tests turbo cam=r1` .
2. wait for the command to end.

### 8.6 Additional Notes

None.

## 9 Test Case SPS-XCU-IONPUMP-070: Cryostat Ion Pumps

### 9.1 Description

Check the communication with the ionpumps of a given camera, check that the statuses are correct. Retrieve and generate ionpump state and telemetry.

### 9.2 Pass/Fail Condition(s)

**Pass** Ionpump dataset will be generated and command should finish with ‘test=ionpump-cam,OK’

**Fail** If an exception is raised ‘test=ionpump-cam,FAILED’ will be generated with its full trace.

### 9.3 Hardware constraints

1. SPS RACK powered up
2. BEE powered up.
3. Ionpump controllers powered up.

### 9.4 Initial conditions

xcuActor and testsActor started

### 9.5 Procedure

1. testsActor : send ionpump test command: `tests ionpump cam=r1` .
2. wait for the command to end.

### 9.6 Additional Notes

None.

## 10 Test Case SPS-XCU-GAUGE-080: Cryostat Gauge

### 10.1 Description

Check the communication with the gauge of a given camera, check that the value is valid

### 10.2 Pass/Fail Condition(s)

**Pass** Gauge dataset will be generated and command should finish with `'test=gauge-cam,OK'`

**Fail** If the sensor reads NaN or zero or if an exception is raised `'test=gauge-cam,FAILED'` will be generated

### 10.3 Hardware constraints

1. SPS RACK powered up
2. BEE powered up.
3. Gauge powered up.

### 10.4 Initial conditions

xcuActor and testsActor started

### 10.5 Procedure

1. testsActor : send gauge test command: `tests gauge cam=r1` .
2. wait for the command to end.

### 10.6 Additional Notes

None.

## 11 Test Case SPS-XCU-COOLER-090: Cryostat Cooler

### 11.1 Description

Check the communication with the cooler of a given camera, check that the statuses are correct. Retrieves and generates Cryocooler telemetry.

### 11.2 Pass/Fail Condition(s)

**Pass** Cooler dataset will be generated and command should finish with ‘test=cooler-cam,OK’

**Fail** If an exception is raised ‘test=cooler-cam,FAILED’ will be generated with its full trace.

### 11.3 Hardware constraints

1. SPS RACK powered up
2. BEE powered up.
3. Cooler controller powered up.

### 11.4 Initial conditions

xcuActor and testsActor started

### 11.5 Procedure

1. testsActor : send cooler test command: `tests cooler cam=r1`.
2. wait for the command to end.

### 11.6 Additional Notes

None.

## 12 Test Case SPS-XCU-TEMPS-100: Cryostat Temps

### 12.1 Description

Check the communication with the temperature board of a given camera, check that the values are valids

### 12.2 Pass/Fail Condition(s)

**Pass** Temps dataset will be generated and command should finish with ‘test=temps-cam,OK’

**Fail** If a sensor reads NaN or zero or if an exception is raised ‘test=temps-cam,FAILED’ will be generated

### 12.3 Hardware constraints

1. SPS RACK powered up
2. BEE powered up.
3. Temperature board powered up.

### 12.4 Initial conditions

xcuActor and testsActor started

### 12.5 Procedure

1. testsActor : send temps test command: `tests temps cam=r1` .
2. wait for the command to end.

### 12.6 Additional Notes

None.



## 13 Test Case SPS-XCU-HEATERS-110: Cryostat Heaters

### 13.1 Description

Check the communication with the heaters of a given camera, check that the statuses are correct.

### 13.2 Pass/Fail Condition(s)

**Pass** Heaters dataset will be generated and command should finish with `'test=heaters-cam,OK'`

**Fail** If an exception is raised `'test=heaters-cam,FAILED'` will be generated with its full trace.

### 13.3 Hardware constraints

1. SPS RACK powered up
2. BEE powered up.
3. Temperature board powered up.

### 13.4 Initial conditions

xcuActor and testsActor started

### 13.5 Procedure

1. testsActor : send heaters test command: `tests heaters cam=r1 .`
2. wait for the command to end.

### 13.6 Additional Notes

None.

## 14 Test Case SPS-LOG-120: Process logging

### 14.1 Description

Checks that logging is active for a typical process.

### 14.2 Pass/Fail Condition(s)

**Pass** Logging of INFO, FAIL and WARN messages are correctly written to the expected log file

**Fail** Any result otherwise.

### 14.3 Hardware constraints

None.

### 14.4 Initial conditions

/data and /software volumes are created.

### 14.5 Procedure

None.

Step	Description	Pass/FAIL	Comment
1	TBW		

### 14.6 Additional Notes

A test script installed in the /software folder would address this test effectively.

## 15 Test Case SPS-IO-130: Test File I/O

### 15.1 Description

Checks that data can be read and written from the input and output directories.

### 15.2 Pass/Fail Condition(s)

**Pass** Data can be written to and read from the `/data` volume, and software can be written to and read from the `/software` volume.

**Fail** Any result otherwise.

### 15.3 Hardware constraints

None.

### 15.4 Initial conditions

`/data` and `/software` volumes are created.

### 15.5 Procedure

For each volume:

Step	Description	Pass/FAIL	Comment
1	Create small files		
2	Read those files		
3	Copy those files to new locations within same volume and check that they are readable.		

### 15.6 Additional Notes

A test script installed in the `/software` folder would address this test effectively.

## 16 Test Case SPS-BIAS-140: Detector Bias

### 16.1 Description

Take a bias on a given detector, check that the file is created correctly and fits header critical keys are set. Check that `IMAGETYP==BIAS` and `exptime==0`

### 16.2 Pass/Fail Condition(s)

**Pass** filepath and fits header keys be generated and command should finish with `'test=bias-cam,OK'`

**Fail** If an exception is raised `'test=bias-cam,FAILED'` will be generated with its full trace.

### 16.3 Hardware constraints

1. SPS RACK powered up
2. **BEE** powered up.
3. Fee powered up

### 16.4 Initial conditions

ccdActor and testsActor started

### 16.5 Procedure

1. testsActor : send bias test command: `tests bias cam=r1 .`
2. wait for the command to end.

### 16.6 Additional Notes

None.

## 17 Test Case SPS-DARK-150: Detector Dark

### 17.1 Description

Take a 10 seconds dark on a given detector, check that the file is created correctly and FITS header-critical keys are set. Check that `IMAGETYP==DARK` and `exptime==10`

### 17.2 Pass/Fail Condition(s)

**Pass** filepath and fits header keys be generated and command should finish with `'test=dark-cam,OK'`

**Fail** If an exception is raised `'test=dark-cam,FAILED'` will be generated with its full trace.

### 17.3 Hardware constraints

1. SPS RACK powered up
2. BEE powered up.
3. Fee powered up

### 17.4 Initial conditions

ccdActor and testsActor started

### 17.5 Procedure

1. testsActor : send dark test command: `tests dark cam=r1` .
2. wait for the command to end.

### 17.6 Additional Notes

None.

## 18 Test Case SPS-SHUTTER-160: Shutter control

### 18.1 Description

Check the communication with the shutters, check that the statuses are correct.  
Check movement for each shutter :

Step	Description
1	Ask for a 5 seconds exposure
2	Check the exptime error is below 0.1 sec
3	Check that the transient time is below 1 sec

Open each shutter and check BIA interlock.

### 18.2 Pass/Fail Condition(s)

**Pass** Shutters open/close as expected with the correct speed, turning BIA on is forbidden  
`'test=shutters-smId,OK'`

**Fail** If an exception is raised `'test=shutters-smId,FAILED'` will be generated with its full trace.

### 18.3 Hardware constraints

Enu rack powered up and connected to the PFS network

### 18.4 Initial conditions

enuActor and testsActor started

### 18.5 Procedure

1. testsActor : send shutters test command: `tests shutters sm1` .
2. wait for the command to end.

### 18.6 Additional Notes

None.

## 19 Test Case SPS-SLIT-170: Slit movement

### 19.1 Description

Check the communication with the slit hexapod, check that the statuses are correct. Check slit movement such that the expected positions are reached:

1. for each position A, B, C, D, E, F:
  - (a) command slit to be moved to that position
  - (b) check whether the expected position is reached with total absolute error below 50 microns

Step	Description
1	Position A (1,0,0,0,0,0)
2	Position B (0,1,0,0,0,0)
3	Position C (0,0,1,0,0,0)
4	Position D (0,0,0,1,0,0)
5	Position E (0,0,0,0,1,0)
6	Position F (0,0,0,0,0,1)

### 19.2 Pass/Fail Condition(s)

**Pass** Slit movement as expected ‘test=slit-smId,OK’

**Fail** If an exception is raised ‘test=slit-smId,FAILED’ will be generated with its full trace.

### 19.3 Hardware constraints

Slit hexapod controller powered up and connected to the PFS network

### 19.4 Initial conditions

enuActor and testsActor started

### 19.5 Procedure

1. testsActor : send slit test command: `tests slit sm1` .
2. wait for the command to end.

### 19.6 Additional Notes

None.

## 20 Test Case SPS-REXM-180: Red exchange mechanism movement

### 20.1 Description

Check the communication with rexm motor controller, check that the statuses are correct.  
Checks that the red exchange mechanism moves as expected:

1. for each position : low resolution, medium resolution
2. command rexm to be moved to that position
3. check whether the expected position is reached with a correct timing

Step	Description
1	Initialization, low position
2	Go to med position
3	Go back to low position

### 20.2 Pass/Fail Condition(s)

**Pass** Rexam movement as expected ‘`test=rexm-smId,OK`’

**Fail** If an exception is raised ‘`test=rexm-smId,FAILED`’ will be generated with its full trace.

### 20.3 Hardware constraints

Enu rack powered up and connected to the PFS network

### 20.4 Initial conditions

enuActor and testsActor started

### 20.5 Procedure

1. testsActor : send rexm test command: `tests rexm sm1 .`
2. wait for the command to end.

### 20.6 Additional Notes

None.



## 21 Test Case SPS-BIA-190: Check Back Illumination Assembly Photoresistance

### 21.1 Description

Check the communication with biasha board, check that the statuses are correct.  
Turn on the BIA, check photoresistances values and check shutters interlock.

### 21.2 Pass/Fail Condition(s)

**Pass** The photoresistances are measured as expected. opening any shutter is forbidden  
`'test=bia-smId,OK'`

**Fail** If an exception is raised `'test=bia-smId,FAILED'` will be generated with its full trace.

### 21.3 Hardware constraints

Enu rack powered up and connected to the PFS network

### 21.4 Initial conditions

enuActor and testsActor started

### 21.5 Procedure

1. testsActor : send bia test command: `tests bia sm1 .`
2. wait for the command to end.

### 21.6 Additional Notes

None.

## 22 Test Case SPS-TEMPS-200: Check SPS Temperature

### 22.1 Description

Check the communication with the temperature controller, check that the values are valids

### 22.2 Pass/Fail Condition(s)

**Pass** Temps dataset will be generated and command should finish with ‘test=temps-smId,OK’

**Fail** If a sensor reads NaN or zero or if an exception is raised ‘test=temps-smId,FAILED’ will be generated

### 22.3 Hardware constraints

Temperature controller powered up and connected to the PFS network

### 22.4 Initial conditions

enuActor and testsActor started

### 22.5 Procedure

1. testsActor : send temps test command: `tests temps sm1` .
2. wait for the command to end.

## **23 Test Case SPS-IIS-210: Check Internal Illumination Sources**

### **23.1 Description**

Turn on each lamp of Internal Illumination Sources and check its power consumption.

### **23.2 Pass/Fail Condition(s)**

**Pass** The power consumption are measured as expected. `'test=iis-smId,OK'`

**Fail** If an exception is raised `'test=iis-smId,FAILED'` will be generated with its full trace.

### **23.3 Hardware constraints**

Enu rack powered up and connected to the PFS network

### **23.4 Initial conditions**

enuActor and testsActor started

### **23.5 Procedure**

1. testsActor : send iis test command: `tests iis sm1 .`
2. wait for the command to end.

### **23.6 Additional Notes**

None.

