




Open Simulation Framework

openSF

SYSTEM VALIDATION SPECIFICATION

Code : OPENSF-DMS-TEC-SVS-001
Issue : 3.4
Date : 04/04/2014

	Name	Function	Signature
Prepared by	Ricardo Moyano	V&V Manager	
Reviewed by	Rui Mestre	Project Engineer	
Approved by	Ricardo Moyano	Project Manager	
Signatures and approvals on original			

DEIMOS Space S.L.U.
 Ronda de Poniente, 19, Edificio Fiteni VI, 2-2ª
 28760 Tres Cantos (Madrid), SPAIN
 Tel.: +34 91 806 34 50 / Fax: +34 91 806 34 51
 E-mail: deimos@deimos-space.com

This page intentionally left blank

Document Information

Contract Data	
Contract Number:	22852/09/NL/FF
Contract Issuer:	ESA/ESTEC

Internal Distribution		
Name	Unit	Copies
Ricardo Moyano	DMS	1
Enrique del Pozo	DMS	1
Jose A. González	DMS	1
Rui Mestre	DME	1
Internal Confidentiality Level (DMS-COV-POL05)		
Unclassified <input type="checkbox"/>	Restricted <input checked="" type="checkbox"/>	Confidential <input type="checkbox"/>

External Distribution		
Name	Organisation	Copies
Raffaella Franco	ESTEC	1
Lavinia Fabrizi	ESTEC	1
Paolo Bensi	ESTEC	1

Archiving	
Word Processor:	MS Word 2000
File Name:	OPENSF-DMS-SVS-34.doc

Document Status Log

Issue	Change description	Date	Approved
1.0	First version of the document	20/11/2009	
1.1	Second version with updates discussed during AR1 coming from the RIDs and tests execution. Test for the web site have been included. Furthermore, tests have been changed to be in accordance to the test data sets provided by the delivery.	15/03/2010	
1.2	Section 3.5 has been updated reflecting the test directory structure used throughout the test procedures. Minor updates in the test procedures TC-0035 Tool assignment and TP-0085 Tool assignment have been added to the test case and procedure sections respectively.	20/04/2010	
1.3	New version including the updates for openSF version 2.0 <ul style="list-style-type: none"> ▪ Instrument entity removed ▪ Multi-repository capabilities ▪ <i>Parameter Editor</i> test cases ▪ New model languages support, F77, Matlab and IDL 	09/07/2010	
1.4	New version answering the comments generated by ESA on the PDR documentation package. Updated some test procedures for the openSF v2.0 acceptance	22/09/2010	
1.5	Parameter Editor test cases updated accorded to the software provided for acceptance release 2.	15/10/2010	

Issue	Change description	Date	Approved
1.6	<p>Test procedures updated with the RIDS raised during openSF version 2.0 acceptance.</p> <ul style="list-style-type: none"> <input type="checkbox"/> Added Traceability matrices from Test Procedures to Software Versions <input type="checkbox"/> Updated Reference to ECSS-40C <input type="checkbox"/> Typos corrected in the following test procedures TP-210, TP-220, TP-230, TP-240, TP-250, TP-260, TP-270 and TP-280 <input type="checkbox"/> Added section 3.4 giving an overview of the regression testing. <input type="checkbox"/> Added test case/procedure for regression testing process. <input type="checkbox"/> Traceability matrices updated with the new Regression Test Case <input type="checkbox"/> Removed step from test procedures where model redundancy is checked. <input type="checkbox"/> Added license requirements for IDL and Matlab in test procedures TP-220 and TP-240 	12/11/2010	
2.0	<p>New version including extended capabilities for openSF 2.2:</p> <ul style="list-style-type: none"> <input type="checkbox"/> Parameter Perturbation plug-in (from SEPSO) <input type="checkbox"/> Parameter Editor integration <input type="checkbox"/> Tool management extension <input type="checkbox"/> Check output generation <input type="checkbox"/> MATLAB errors inclusion <input type="checkbox"/> Import/Export capabilities <input type="checkbox"/> Extended log capabilities <input type="checkbox"/> Keyboard shortcuts <input type="checkbox"/> GUI Isolation 	03/02/2012	
2.1	<p>Minor corrections made to version 2.0 regarding the procedures to 1) perform perturbations and to 2) set a MATLAB session that controls the errors specific to v2.2 requirements.</p>	21/02/2012	
3.0	<p>New version including the tests for the added capabilities of openSF version 3.</p>	18/04/2013	
3.1	<p>New version answering the comments generated by ESA on the openSF V3 PDR documentation package.</p> <ul style="list-style-type: none"> <input type="checkbox"/> Added test case and test procedure for Migration to openSF v3 (TC-0530 and TP-0450) <input type="checkbox"/> Added performance test case and test procedure for openSF V3 (TC-0540 and TP-0460) <input type="checkbox"/> Updated the regression test 	31/05/2013	
3.2	<p>New version revisiting the tests for the added capabilities of openSF version 3.</p>	29/11/2013	

Issue	Change description	Date	Approved
3.3	<p>New version answering the comments generated by ESA during the openSF V3 AR meeting.</p> <p>Clarification of the following Test Procedures:</p> <ul style="list-style-type: none"> <input type="checkbox"/> TP-0330: Updated to align the naming with the HMI labels and naming; Removed step 15; <input type="checkbox"/> TP-0390: Updated step 1 to indicate how to change the number of execution threads; <input type="checkbox"/> TP-0400: updated step 2 to separate options available in the session definition view from the ones in a model contextual menu; <input type="checkbox"/> TP-0440: updated to clarify how to execute the test program exercising the Error primitives and to describe the outputs obtained in the execution; <p>Implementation of the following RIDs</p> <ul style="list-style-type: none"> <input type="checkbox"/> OSF-AR3-12: Added mpstat library as a prerequisite (section 3.5.3); <input type="checkbox"/> OSF-AR3-13: Updated TP-0150 to mention new behaviour of the functionality in openSF V3; <input type="checkbox"/> OSF-AR3-14: Added TP-0170 for being run as regression test (section 7.1); <input type="checkbox"/> OSF-AR3-15: Added TP-0270 for being run as regression test (section 7.1); <input type="checkbox"/> OSF-AR3-RF-04: Clarified that this document version covers the testing specification applicable from openSF v3 (section 1.2); added section 7.1 listing both the regression tests and the new capability tests to be applied for openSF v3. <p>Implementation of action ACT-AR3-04:</p> <ul style="list-style-type: none"> <input type="checkbox"/> Identified tests to be executed for openSF v3.1 (section 6.3 and 7.2). 	15/01/2014	
3.4	<p>New version including the system tests to address the integration of Python modules in openSF.</p>	04/04/2014	

Table of Contents

1. INTRODUCTION.....	14
1.1. PURPOSE.....	14
1.2. SCOPE.....	14
1.3. DOCUMENT STRUCTURE.....	14
1.4. ACRONYMS AND ABBREVIATIONS.....	15
1.5. DEFINITIONS.....	17
2. RELATED DOCUMENTS.....	18
2.1. APPLICABLE DOCUMENTS.....	18
2.2. REFERENCE DOCUMENTS.....	18
2.3. STANDARDS.....	18
3. TEST SPECIFICATION OVERVIEW	19
3.1. INTRODUCTION	19
3.2. TEST CASE SPECIFICATIONS.....	19
3.3. TEST PROCEDURE SPECIFICATIONS	19
3.4. REGRESSION TESTS.....	20
3.5. TEST ENVIRONMENT.....	20
3.5.1. <i>Hardware environment</i>	20
3.5.2. <i>Operating system</i>	21
3.5.3. <i>Framework pre-requisites</i>	21
3.5.4. <i>IDL and Matlab</i>	21
3.6. TEST ORGANISATION	22
3.7. TEST DATA	22
4. SYSTEM CASE SPECIFICATIONS.....	24
4.1. TEST CASE SPECIFICATION	24
4.1.1. <i>Test case TC-0010: Installation</i>	24
4.1.2. <i>Test case TC-0020: Configuration set-up</i>	25
4.1.3. <i>Test case TC-0030: Configuration edition</i>	26
4.1.4. <i>Test case TC-0035: Tool assignment</i>	27
4.1.5. <i>Test case TC-0040: I/O descriptors creation</i>	28
4.1.6. <i>Test case TC-0050: I/O descriptors deletion</i>	29
4.1.7. <i>Test case TC-0060: Model creation</i>	30
4.1.8. <i>Test case TC-0070: Incorrect model creation</i>	31
4.1.9. <i>Test case TC-0080: Model edition</i>	32
4.1.10. <i>Test case TC-0090: Model version creation</i>	33
4.1.11. <i>Test case TC-0100: Incorrect model version creation</i>	34
4.1.12. <i>Test case TC-0110: Model deletion</i>	35
4.1.13. <i>Test case TC-0120: Simulation creation</i>	36
4.1.14. <i>Test case TC-0130: Incorrect simulation creation</i>	37
4.1.15. <i>Test case TC-0140: Simulation edition</i>	38
4.1.16. <i>Test case TC-0150: Simulation deletion</i>	39
4.1.17. <i>Test case TC-0160: Session creation</i>	40

4.1.18. Test case TC-0170: Incorrect session creation.....	41
4.1.19. Test case TC-0180: Session edition.....	42
4.1.20. Test case TC-0190: Session deletion.....	43
4.1.21. Test case TC-0200: Session execution	44
4.1.22. Test case TC-0210: Failed session execution	45
4.1.23. Test case TC-0220: Session abortion.....	46
4.1.24. Test case TC-0230: Iterative session execution	47
4.1.25. Test case TC-0240: Session script execution	48
4.1.26. Test case TC-0250: Session execution with breakpoints.....	49
4.1.27. Test case TC-0260: Session results visualisation.....	50
4.1.28. Test case TC-0270: Session plots visualisation	51
4.1.29. Test case TC-0280: Multi-language model session execution	52
4.1.30. Test case TC-0290: Multi-platform session execution	53
4.1.31. Test case TC-0300: Web access for new users.....	54
4.1.32. Test case TC-0310: Web access for delivery download.....	55
4.1.33. Test case TC-0320: Web access for SPR management	56
4.1.34. Test case TC-0330: Multi-repository capabilities.....	57
4.1.35. Test case TC-0340: Multi-repository management.....	58
4.1.36. Test case TC-0350: Model languages support.....	59
4.1.37. Test case TC-0360: Parameter management system	60
4.1.38. Test case TC-0370: Rules definition	61
4.1.39. Test case TC-0380: Parameter validation	62
4.1.40. Test case TC-0390: Parameter editor integration	63
4.1.41. Test case TC-0400: Parameter perturbation	64
4.1.42. Test case TC-0410: MATLAB errors.....	65
4.1.43. Test case TC-0420: Shortcuts.....	66
4.1.44. Test case TC-0430: Backup/Restore capability	67
4.1.45. Test case TC-0440: Import/export capability	68
4.1.46. Test case TC-0450: GUI closure.....	69
4.1.47. Test case TC-0460: Parallelization.....	70
4.1.48. Test case TC-0470: Removal of logs from DB.....	71
4.1.49. Test case TC-0480: Flexible session management.....	72
4.1.50. Test case TC-0490: Model export/import	73
4.1.51. Test case TC-0500: Copy elements	74
4.1.52. Test case TC-0510: Element definition from XML.....	75
4.1.53. Test case TC-0520: Error generation capabilities.....	76
4.1.54. Test case TC-0530: Migration from openSF V2 to openSF V3.....	77
4.1.55. Test case TC-0540: Performance improvement of openSF V3	78
4.1.56. Test case TC-1000: Regression Test	79
4.2. INSPECTION CASE SPECIFICATION	81
4.2.1. Inspection case IC-0010: Framework inspections.....	81
4.2.2. Inspection case IC-0020: openSF code inspection	82
4.3. REVIEW CASE SPECIFICATION.....	83
4.3.1. Review case RC-0010: Development review.....	83
4.3.2. Review case RC-0020: Security review.....	84
4.3.3. Review case RC-0030: Documentation review	85
4.3.4. Review case RC-0040: Acceptance review	86
5. TEST PROCEDURES SPECIFICATION	87

5.1. TEST PROCEDURE TP-0010: INSTALLATION.....	87
5.1.1. Objective.....	87
5.1.2. Pre-conditions	87
5.1.3. Procedure Steps.....	87
5.2. TEST PROCEDURE TP-0020: CONFIGURATION SET-UP.....	88
5.2.1. Objective.....	88
5.2.2. Pre-conditions	88
5.2.3. Procedure Steps.....	88
5.3. TEST PROCEDURE TP-0030: I/O DESCRIPTOR MANAGEMENT	89
5.3.1. Objective.....	89
5.3.2. Pre-conditions	89
5.3.3. Procedure Steps.....	89
5.4. TEST PROCEDURE TP-0040: MODEL CREATION	90
5.4.1. Objective.....	90
5.4.2. Pre-conditions	90
5.4.3. Procedure Steps.....	90
5.5. TEST PROCEDURE TP-0050: MODEL EDITION AND DELETION	92
5.5.1. Objective.....	92
5.5.2. Pre-conditions	92
5.5.3. Procedure Steps.....	92
5.6. TEST PROCEDURE TP-0060: MODEL VERSION CREATION	93
5.6.1. Objective.....	93
5.6.2. Pre-conditions	93
5.6.3. Procedure Steps.....	93
5.7. TEST PROCEDURE TP-0070: SIMULATION CREATION	94
5.7.1. Objective.....	94
5.7.2. Pre-conditions	94
5.7.3. Procedure Steps.....	94
5.8. TEST PROCEDURE TP-0075: STAGE MANAGEMENT WITH MODELS AND SIMULATIONS.....	96
5.8.1. Objective.....	96
5.8.2. Pre-conditions	96
5.8.3. Procedure Steps.....	96
5.9. TEST PROCEDURE TP-0080: SIMULATION EDITION AND DELETION	97
5.9.1. Objective.....	97
5.9.2. Pre-conditions	97
5.9.3. Procedure Steps.....	97
5.10. TEST PROCEDURE TP-0085: TOOL ASSIGNMENT	98
5.10.1. Objective.....	98
5.10.2. Pre-conditions	98
5.10.3. Procedure Steps.....	98
5.11. TEST PROCEDURE TP-0090: SESSION CREATION	99
5.11.1. Objective.....	99
5.11.2. Pre-conditions	99
5.11.3. Procedure Steps.....	99
5.12. TEST PROCEDURE TP-0100: SESSION EDITION AND DELETION	101
5.12.1. Objective.....	101
5.12.2. Pre-conditions	101
5.12.3. Procedure Steps.....	101
5.13. TEST PROCEDURE TP-0110: SESSION EXECUTION.....	102

5.13.1. Objective.....	102
5.13.2. Pre-conditions	102
5.13.3. Procedure Steps.....	102
5.14. TEST PROCEDURE TP-0120: FAILED SESSION EXECUTION	103
5.14.1. Objective.....	103
5.14.2. Pre-conditions	103
5.14.3. Procedure Steps.....	103
5.15. TEST PROCEDURE TP-0130: SESSION PLOTS VISUALISATION	104
5.15.1. Objective.....	104
5.15.2. Pre-conditions	104
5.15.3. Procedure Steps.....	104
5.16. TEST PROCEDURE TP-0140: ABORT SESSION EXECUTION	105
5.16.1. Objective.....	105
5.16.2. Pre-conditions	105
5.16.3. Procedure Steps.....	105
5.17. TEST PROCEDURE TP-0150: ITERATIVE SESSION EXECUTION	106
5.17.1. Objective.....	106
5.17.2. Pre-conditions	106
5.17.3. Procedure Steps.....	106
5.18. TEST PROCEDURE TP-0160: SESSION SCRIPT EXECUTION	108
5.18.1. Objective.....	108
5.18.2. Pre-conditions	108
5.18.3. Procedure Steps.....	108
5.19. TEST PROCEDURE TP-0170: SESSION EXECUTION WITH BREAKPOINTS	109
5.19.1. Objective.....	109
5.19.2. Pre-conditions	109
5.19.3. Procedure Steps.....	109
5.20. TEST PROCEDURE TP-0180: WEB ACCESS FOR NEW USERS	110
5.20.1. Objective.....	110
5.20.2. Pre-conditions	110
5.20.3. Procedure Steps.....	110
5.21. TEST PROCEDURE TP-0190: WEB ACCESS FOR DELIVERY DOWNLOAD	111
5.21.1. Objective.....	111
5.21.2. Pre-conditions	111
5.21.3. Procedure Steps.....	111
5.22. TEST PROCEDURE TP-0200: WEB ACCESS FOR SPR MANAGEMENT	112
5.22.1. Objective.....	112
5.22.2. Pre-conditions	112
5.22.3. Procedure Steps.....	112
5.23. TEST PROCEDURE TP-0210: IDL MODEL CREATION	113
5.23.1. Objective.....	113
5.23.2. Pre-conditions	113
5.23.3. Procedure Steps.....	113
5.24. TEST PROCEDURE TP-0220: SESSION EXECUTION WITH MODELS IN IDL.....	115
5.24.1. Objective.....	115
5.24.2. Pre-conditions	115
5.24.3. Procedure Steps.....	115
5.25. TEST PROCEDURE TP-0230: MATLAB MODEL CREATION	118
5.25.1. Objective.....	118

5.25.2. Pre-conditions	118
5.25.3. Procedure Steps.....	118
5.26. TEST PROCEDURE TP-0240: SESSION EXECUTION WITH MODELS IN MATLAB	120
5.26.1. Objective.....	120
5.26.2. Pre-conditions	120
5.26.3. Procedure Steps.....	120
5.27. TEST PROCEDURE TP-0250: REPOSITORY CREATION	123
5.27.1. Objective.....	123
5.27.2. Pre-conditions	123
5.27.3. Procedure Steps.....	123
5.28. TEST PROCEDURE TP-0260: REPOSITORY DELETION	124
5.28.1. Objective.....	124
5.28.2. Pre-conditions	124
5.28.3. Procedure Steps.....	124
5.29. TEST PROCEDURE TP-0270: REPOSITORY SWITCHING	125
5.29.1. Objective.....	125
5.29.2. Pre-conditions	125
5.29.3. Procedure Steps.....	125
5.30. TEST PROCEDURE TP-0280: PARAMETER EDITOR – PARAMETER CREATION.....	126
5.30.1. Objective.....	126
5.30.2. Pre-conditions	126
5.30.3. Procedure Steps.....	126
5.31. TEST PROCEDURE TP-0290: PARAMETER EDITOR – PARAMETER EDITION, DELETION.....	128
5.31.1. Objective.....	128
5.31.2. Pre-conditions	128
5.31.3. Procedure Steps.....	128
5.32. TEST PROCEDURE TP-0300: PARAMETER EDITOR – PARAMETER CONSISTENCY CHECKING	129
5.32.1. Objective.....	129
5.32.2. Pre-conditions	129
5.32.3. Procedure Steps.....	129
5.33. TEST PROCEDURE TP-0310: PARAMETER EDITOR – RULE CREATION.....	130
5.33.1. Objective.....	130
5.33.2. Pre-conditions	130
5.33.3. Procedure Steps.....	130
5.34. TEST PROCEDURE TP-0320: PARAMETER EDITOR – PARAMETER VALIDATION	131
5.34.1. Objective.....	131
5.34.2. Pre-conditions	131
5.34.3. Procedure Steps.....	131
5.35. TEST PROCEDURE TP-0330: PARAMETER PERTURBATION	132
5.35.1. Objective.....	132
5.35.2. Pre-conditions	132
5.35.3. Procedure Steps.....	132
5.36. TEST PROCEDURE TP-0340: MATLAB ERRORS	134
5.36.1. Objective.....	134
5.36.2. Pre-conditions	134
5.36.3. Procedure Steps.....	134
5.37. TEST PROCEDURE TP-0350: SHORTCUTS	135
5.37.1. Objective.....	135
5.37.2. Pre-conditions	135

5.37.3. Procedure Steps.....	135
5.38. TEST PROCEDURE TP-0360: BACKUP/RESTORE.....	136
5.38.1. Objective.....	136
5.38.2. Pre-conditions	136
5.38.3. Procedure Steps.....	136
5.39. TEST PROCEDURE TP-0370: IMPORT/EXPORT.....	137
5.39.1. Objective.....	137
5.39.2. Pre-conditions	137
5.39.3. Procedure Steps.....	137
5.40. TEST PROCEDURE TP-0380: GUI CLOSURE	138
5.40.1. Objective.....	138
5.40.2. Pre-conditions	138
5.40.3. Procedure Steps.....	138
5.41. TEST PROCEDURE TP-0390: PARALLELIZATION	139
5.41.1. Objective.....	139
5.41.2. Pre-conditions	139
5.41.3. Procedure Steps.....	139
5.42. TEST PROCEDURE TP-0400: FLEXIBLE SESSION MANAGEMENT.....	140
5.42.1. Objective.....	140
5.42.2. Pre-conditions	140
5.42.3. Procedure Steps.....	140
5.43. TEST PROCEDURE TP-0410: MODEL EXPORT/IMPORT	141
5.43.1. Objective.....	141
5.43.2. Pre-conditions	141
5.43.3. Procedure Steps.....	141
5.44. TEST PROCEDURE TP-0420: COPY ELEMENTS.....	142
5.44.1. Objective.....	142
5.44.2. Pre-conditions	142
5.44.3. Procedure Steps.....	142
5.45. TEST PROCEDURE TP-0430: ELEMENT DEFINITION FROM XML.....	143
5.45.1. Objective.....	143
5.45.2. Pre-conditions	143
5.45.3. Procedure Steps.....	143
5.46. TEST PROCEDURE TP-0440: ERROR GENERATION CAPABILITIES	144
5.46.1. Objective.....	144
5.46.2. Pre-conditions	144
5.46.3. Procedure Steps.....	144
5.47. TEST PROCEDURE TP-0450: MIGRATION PROCEDURE TO OPENSF V3.....	145
5.47.1. Objective.....	145
5.47.2. Pre-conditions	145
5.47.3. Procedure Steps.....	145
5.48. TEST PROCEDURE TP-0460: PERFORMANCE TEST – E2E SESSION EXECUTION.....	146
5.48.1. Objective.....	146
5.48.2. Pre-conditions	146
5.48.3. Procedure Steps.....	146
5.49. TEST PROCEDURE TP-0470: PYTHON MODEL CREATION	148
5.49.1. Objective.....	148
5.49.2. Pre-conditions	148
5.49.3. Procedure Steps.....	148

5.50. TEST PROCEDURE TP-0480: SESSION EXECUTION WITH MODELS IN PYTHON	150
5.50.1. Objective.....	150
5.50.2. Pre-conditions	150
5.50.3. Procedure Steps.....	150
5.51. TEST PROCEDURE TP-1000: REGRESSION TEST – E2E SESSION EXECUTION	152
5.51.1. Objective.....	152
5.51.2. Pre-conditions	152
5.51.3. Procedure Steps.....	152
6. TRACEABILITY MATRICES	154
6.1. TRACEABILITY BETWEEN VALIDATION CASES AND SYSTEM REQUIREMENTS.....	154
6.1.1. Traceability Matrix from Test Cases to SRD Requirements	154
6.1.2. Traceability Matrix from SRD Requirements to Test cases	160
6.2. TRACEABILITY BETWEEN TEST PROCEDURES AND TEST CASES	167
6.2.1. Traceability Matrix from Test Procedures to Test Cases.....	167
6.2.2. Traceability Matrix from Test Cases to Test Procedures.....	169
6.3. TRACEABILITY BETWEEN TEST PROCEDURES AND SOFTWARE VERSIONS	171
6.3.1. Traceability Matrix from Test Procedures to Software Versions.....	171
6.3.2. Traceability Matrix from Software Versions to Test Procedures.....	173
7. OPENSF TEST PLAN	175
7.1. OPENSF V3 TEST PLAN.....	175
7.2. OPENSF V3.1 TEST PLAN.....	176

List of Tables

Table 2-1: Applicable documents.....	18
Table 2-2: Reference documents	18
Table 2-3: Standards.....	18
Table 6-1: Traceability Matrix, Test Cases vs. SRD Requirements.....	154
Table 6-2: Traceability Matrix, SRD Requirements vs. Test Cases	160
Table 6-3: Traceability Matrix Test Procedures vs. Test Cases	167
Table 6-4: Traceability Matrix, Test Cases vs. Test Procedures	169
Table 6-5: Traceability Matrix, Test Procedures vs. Software Versions.....	171
Table 6-6: Traceability Matrix, Software Versions vs. Test Procedures.....	173
Table 7-1: Test Cases and Test Procedures for openSF V3	175
Table 7-2: Test Cases and Test Procedures for openSF V3.1	176

1. INTRODUCTION

1.1. Purpose

The present document has been produced by DEIMOS within the frame of the openSF project, and it describes the validation specification used for the system testing of the openSF software.

This document is a child document of the openSF Software Design, Development and Verification Plan (DDVP), and has been extracted from the document due to the size and the adequacy of having an independent document for sake of the validation of the system.

This document is intended to describe the items that shall be used to validate that openSF complies with the established system requirements, listed in [SRD], as one of the activities for the validation of the system.

1.2. Scope

This document covers the full openSF system requirements, regardless of the validation method that satisfies them (Test (T), Inspection (I), or Review of design (R)), although most of them shall be validated by Test. The testing specification is limited exclusively to the framework capabilities, although simple models shall be built and used for the simulation related tests.

The verification approach for each new version of openSF is based on (a) the execution of a set of test cases, specified to cover the functionalities introduced by the new openSF version, and (b) the execution of a regression test to confirm that the capabilities from previous openSF versions have not been affected. Refer to section 6.3.2 for further details on which test cases are specified for each openSF version.

This version of the SVS document covers the testing specification applicable starting from openSF V3. Please refer to section 7.2 for details on which test cases and test procedures are specified for openSF version 3.1.

1.3. Document structure

Information provided in this document includes:

- Introduction to the document itself (the present chapter).
- List of reference and applicable documents (section 2).
- The description of how the Test Cases and Procedures are arranged (section 3).
- Test case specifications, including for each test, the objective, the list of software requirements covered, the input and outputs and the pass-fail criteria (section 4).
- Test procedure specifications, including for each test, the objective, the pre-condition to be satisfied so that the test can successfully executed, and the list of steps for running the test (section 5).
- Traceability matrices between test definitions and software requirements and vice-versa respectively.

1.4. Acronyms and Abbreviations

The acronyms and abbreviations used in this document are the following ones:

Acronym	Description
AD	Architectural Design Applicable Document
ADD	Architectural Design Document
API	Application Programming Interface
AR	Acceptance Review
AT	Acceptance Test
CASE	Computer Aided Software Engineering
CFI	Customer Furnished Item
CM	Configuration Management Configuration Manager
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
DBMS	Database Management System
DDVP	Design, Development and Validation Plan
DMS	DEIMOS Space
ECP	Engineering Change Proposal
E-R	Entity Relationship
FAR	Factory Acceptance Review
FAT	Factory Acceptance Test
GUI	Graphical User Interface
HW	Hardware
I/F	Interface
I/O	Input/Output
ICD	Interface Control Document
IT	Integration Test
KOM	Kick-Off Meeting
MoM	Minutes of Meeting
MPI	Message Passing Interface
MPL	Message-Passing Language
MR	Management Review
NCR	Non-Conformance Report
OO	Object-Oriented
PA	Product Assurance
PAP	Product Assurance Plan

Acronym	Description
PDR	Preliminary Design Review
PM	Progress Meeting Project Manager
PMP	Project Management Plan
QA	Quality Assurance
QAP	Quality Assurance Plan
QR	Qualification Review
RD	Reference Document
RDBMS	Relational Data Base Management System
SCMP	Software Configuration Management Plan
SDP	Software Development Plan
SMR	Software Modification Report
SOW	Statement Of Work
SPR	Software Problem Report
SRD	System Requirements Document
SRN	Software Release Note
ST	System Test
STP	Software Test Plan
SUM	System User Manual
SVTR	System Validation Testing Report
SVVP	Software Verification & Validation Plan
SW	Software
TBC	To Be Confirmed
TBD	To Be Defined / Decided
UML	Unified Modelling Language
UT	Unit Test
V&V	Verification & Validation
WBS	Work Breakdown Structure
WPD	Work Package Description

1.5. Definitions

The definitions of the specific terms used in this document are the following ones:

- ❑ **Acceptance testing:** Formal testing conducted to determine whether or not a system satisfies the acceptance criteria, previously defined by the customer.
- ❑ **Anomaly:** Anything observed in the documentation or operation of software that deviates from expectations based on previously verified software products or reference documents.
- ❑ **Component Testing:** Testing conducted to verify the implementation of the design for one software element (e.g. unit, module) or a collection of software elements.
- ❑ **Integration Testing:** An orderly progression of testing in which software elements, hardware elements, or both are combined and tested until the entire system has been integrated.
- ❑ **System Testing:** The process of testing an integrated hardware and software system to verify that the system meets its software requirements.
- ❑ **Test Case:** Documentation specifying inputs, predicted results, and a set of execution conditions for a test item.
- ❑ **Test Plan:** Documentation specifying the scope, approach, resources, and schedule of intended testing activities.
- ❑ **Test Procedure:** Documentation specifying a sequence of actions for the execution of a test.
- ❑ **Validation:** The process of evaluating software at the end of the software development process to ensure compliance with software requirements.
- ❑ **Verification:** The process of determining whether or not the products of a given phase of the software development life cycle fulfil the requirements established during the previous phase.

Additional definitions more specific to openSF can be found in [SRD].

2. RELATED DOCUMENTS

This section details the list of applicable and reference documents used for the generation of this document, as well as the standards that have been applied. Notice that the latest issue and dates of the documents can be found on the openSF wiki page (opensf.deimos-space.com).

2.1. Applicable Documents

The following table specifies the applicable documents that shall be complied with during project development.

Table 2-1: Applicable documents

Reference	Code	Title
[SRD]	OPENSF-DMS-SRD-001	openSF System Requirements Document
[ICD]	OPENSF-DMS-ICD-001	openSF Interface Control Document
[SUM]	OPENSF-DMS-SUM-001	openSF Software User Manual
[ADD]	OPENSF-DMS-ADD-001	openSF Architecture Design Document
[CR1]	EOP-SFP/2012-12-1686/PB/ag	Change Request for the openSF V3 activities description

2.2. Reference Documents

The following table specifies the reference documents that shall be taken into account during project development.

Table 2-2: Reference documents

Reference	Code	Title
[DDVP]	OPENSF-DMS-DDVP-001	openSF Design, Development and Validation Plan

2.3. Standards

The following table specifies the standards that shall be complied with during project development.

Table 2-3: Standards

Reference	Code	Title
[Q80B]	ECSS-Q-80B	ECSS Space Product Assurance – Software Product Assurance
[E40C]	ECSS-E-40C	Software development Standard

3. TEST SPECIFICATION OVERVIEW

3.1. Introduction

Tests are designed to verify features or sets of features of the system. One feature may correspond to one only requirement or to a group of several requirements.

The features that are to be tested correspond to the system requirements that are to be verified by test, according to the validation method of the requirements as identified in [SRD].

For sake of an effective validation of openSF, DEIMOS has taken the approach to specify as many test cases as different features or functionalities of the simulator. On the other hand, test procedures shall be specified to define logical use cases of the simulator covering as many test cases as possible. At the end the major goal is to ensure that, with the execution of the test procedures, all system requirements validated by Test satisfied.

3.2. Test Case Specifications

Test cases must specify the details of the test approach for the set of the system requirements to be validated. Test cases shall be defined containing the following information:

- ❑ **Test case identifier:** Test cases shall be identified according the following convention:

TC-*nnnn*: <test case name>

where *nnnn* is a sequence number using four digits (e.g. 0010), and <test case name> is the name associated to the test case. The test case name will be unique for each test case, and will be made up of a few descriptive words.

- ❑ **Purpose:** Objectives of the test case in terms of features that are to be tested.
- ❑ **System requirement coverage:** List of software requirements validated by the test case.
- ❑ **Resources and tools.** Needed resources and tools needed by the use cases. For example, they may be scripts or utilities to verify the test results.
- ❑ **Inputs:** Input data needed for the test case. Specifies the data sets and the values of the data sets representing the inputs to the test case.
- ❑ **Outputs:** Expected output data produced as a result of the test case execution.
- ❑ **Pass/Fail criteria:** Describes the criteria to be used to determine whether the test passes or fails, and consequently determines if the involved requirements are fulfilled or not.

3.3. Test Procedure Specifications

Test procedures describe the sequence of actions the operator or test engineer should handle in order to properly execute the test.

A test procedure is constituted by:

- ❑ **Test procedure identifier:** Test procedures shall be identified according the following convention:

TP-*nnnn*: <test procedure name>

where *nnnn* is a sequence number using four digits (e.g. 0010), and <test procedure name> is the name associated to the test case. The test procedure name will be unique, and will be made up of a few descriptive words.

- ❑ **Objective:** Objectives of the test procedure, generally in terms of the test cases covered by the procedure. As mentioned earlier, a test procedure is planned to may cover one or more test cases, depending on the design of the procedure.
- ❑ **Pre-conditions:** List of conditions that must be met to ensure the correct behaviour of the test procedure.
- ❑ **Procedure steps:** Identify all steps of the procedure and the corresponding expected results. Those steps and activities that are repeatedly performed for many tests can be placed in a separate test procedure which is invoked by all affected procedures.

3.4. Regression Tests

Regression testing is the process of testing changes to computer programs to make sure that the older programming still works with the new changes, and is aimed to uncover software errors after an update due to new implementations or SPR solutions.

Regression testing identifies when code modifications cause previously-working functionality to regress, or fail, ultimately allowing you to catch regression errors as soon as they are introduced. It is a common mistake testing a piece of code only once, and then assumes it continues to work unless they intentionally modify it. However, even routine and minor code changes can have unexpected side effects that might break previously-verified functionality.

In the openSF case to perform the validation of all software functionalities requires running all test procedures. This is not practical in terms of hours as running all test procedures can take long time, and those shall be re-run every new release. For speeding up the validation process a test case/procedure has been defined which basically tests core functionalities of the framework (running an E2E simulation scenario).

3.5. Test Environment

This section identifies the elements that make up the environment needed to run the system tests defined in this document.

This environment is composed of the elements necessary to execute the tests:

1. The execution environment, which comprises the elements necessary to run system tests, i.e. hardware, operating system, software tools, etc.
2. The validation methods, which provide the means to validate the results of the tests.
3. The test data.

3.5.1. Hardware environment

Hardware must at least fulfil the following requirements:

- ❑ 32-bit 1GHz processor

- 1 GB of RAM memory installed
- 30MB of free space to install.

3.5.2. Operating system

Operating systems supported by openSF are three.

- Linux. Kernel version 2.6.
- Windows. Windows XP 32 bits.
- MacOS X 10.6 Snow Leopard.

For acceptance, it is recommended to use the Linux environment. In case the software version under testing is 3.0, only Linux OS is supported.

3.5.3. Framework pre-requisites

Pre-requisite	Purpose	Licensing	Distribution site
Sun Java(TM) 2 Runtime Environment, Standard Edition 1.6 or superior	OpenSF runs within this execution environment.	GNU GPL / Java Community Process	http://www.java.com/en/download/
MySQL client and server 5.6 or superior	OpenSF stores information in this relational database	GPL or Proprietary License	http://dev.mysql.com/downloads/mysql/5.6.html
MySQL connector/J 5.1.24	OpenSF uses this library to connect to the MySQL database server.	GPL or Proprietary License	http://dev.mysql.com/get/Downloads/Connector-J/mysql-connector-java-5.1.24.tar.gz/from/pick
mpstat	OpenSF uses this library to assess the cpu core usage statistics.	GNU GPL	http://linuxcommand.org/man_pages/mpstat1.html

After installing the MySQL server, the user name and password must be set to root.

Note that after the installation of the Sun Java 2 RE and the MySQL client and server, the PATH system variable must contain the folder location of their main executables.

Additionally the MySQL connector library is distributed within openSF package; users shall check the presence of it in the OPENSF_HOME folder.

3.5.4. IDL and Matlab

In order to run the tests corresponding to IDL™ and Matlab™ models the following pre-requisites are needed:

- Matlab execution tests:** A full-licensed Matlab™ R2008b or superior must be installed and accessible through the system path. No extra toolboxes required.

- ❑ **IDL execution tests:** the software IDL™ 6.2 or superior must be installed and accessible through the system path. Regarding the license needed, it is recommended a full-development one but is also possible to run corresponding tests with Runtime and Virtual-Machine licenses.

For a more detailed explanation about IDL™ and Matlab™ installation and license issues please check the correspondent section in [SUM]. See also [ADD] for details about IDL and Matlab interaction mechanism.

3.6. Test Organisation

Test procedures shall be organised under a dedicated directory named `test`. An environment variable named `TEST_HOME` shall point to this directory. Under this directory, data are arranged in a set of sub-directories:

- **bin/** including the set of files that will represent the binary or executable files of the models created for the tests and are used for the simulation examples.
- **data/**, used to store the files related to the models. It is further split in two directories:
 - **conf/** locating the samples of the configuration files for the models created for the tests
 - **input/** containing the input files for the models created for the tests
- **src/** where the source files for the models are kept.

The figure below summarises the directory structure applicable for the openSF test development.

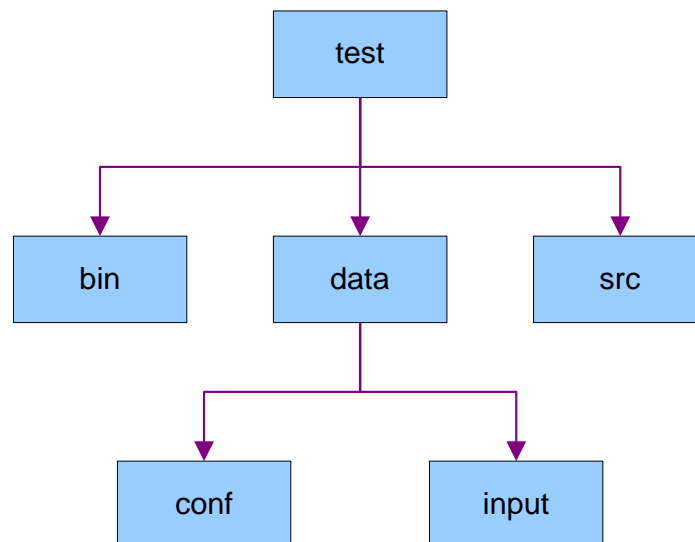


Figure 3-1: Test directory structure

3.7. Test Data

When installing the openSF tool there are two possibilities given to the user in what respects to the data sets to be loaded:

1. **Empty data set.** No data shall be present in the system, meaning that users need to define the stages and IO descriptors, and from there, create the models, simulations and sessions.

2. **Validation data set.** This data set represents the status of the data repository upon a successful execution of all the tests provided in this document. The validation data set is useful for a quicker familiarization of new users of the elements used by openSF.

Consequently, for the validation campaign laid out in this document, the installation of the openSF with an empty data set has been used.

4. SYSTEM CASE SPECIFICATIONS

This section comprises the case specifications to fulfil all openSF system requirements. Cases are divided following to the different validation methods used in openSF to validate requirements. The validation methods applied in openSF are:

- **Test (T)** – Execution of the element under certain conditions to check the outputs corresponding to particular inputs. Test Case Specifications are specified according to the description given in section 3.
- **Inspection (I)** – Exhaustive evaluation of the code by manual reading;
- **Review of design (R)** – Review of project documentation.

4.1. Test Case Specification

4.1.1. Test case TC-0010: Installation

4.1.1.1. Purpose

This test case checks the automatism of the installation procedure for openSF.

4.1.1.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0050 SR-INS-0010

4.1.1.3. Resources and Tools

- Installation script that automatically installs the tool when executed.

4.1.1.4. Input Specification

N/A

4.1.1.5. Output Specification

- openSF is installed and working.

4.1.1.6. Pass/Fail Criteria

[PFC-1] The installation procedure is highly automated via executable files/scripts.

[PFC-2] openSF is installed according to the installation procedure steps explained in the Software User Manual, [SUM].

[PFC-3] When launched, openSF includes a splash screen.

[PFC-4] openSF provides a framework including functions to perform model management, simulation management, session management, and session result visualisation.

4.1.2. Test case TC-0020: Configuration set-up

4.1.2.1. Purpose

This test case checks the capability of openSF to set-up the configuration of the tool in terms of the different stages that participate in simulations.

4.1.2.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

		SR-FUN-0340	SR-FUN-0350
SR-FUN-0360	SR-FUN-0380	SR-OPE-0010	

4.1.2.3. Resources and Tools

N/A

4.1.2.4. Input Specification

The input for the test case is constituted by the data fields that define the stages of the processing chains:

- For each stage the following information is needed
 - Stage identifier
 - Description
 - Stage position in the processing chain

4.1.2.5. Output Specification

- OpenSF configuration set-up

4.1.2.6. Pass/Fail Criteria

[PFC-1] A function to add stages is available from the openSF HMI.

[PFC-2] After the successful configuration set-up, stages appear on the corresponding lists that openSF presents to users upon selection.

4.1.3. Test case TC-0030: Configuration edition

4.1.3.1. Purpose

This test case checks the capability of openSF to edit the configuration of the tool in terms of changing the stages lists applicable to simulations/sessions, and deleting stages from an existing configuration.

4.1.3.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

	SR-FUN-0390	SR-FUN-0400	SR-FUN-0410
SR-FUN-0420	SR-OPE-0010		

4.1.3.3. Resources and Tools

N/A

4.1.3.4. Input Specification

NA

4.1.3.5. Output Specification

OpenSF configuration set-up

4.1.3.6. Pass/Fail Criteria

[PFC-1] A function to modify stages is available from the openSF HMI.

[PFC-2] A function to delete stages is available from the openSF HMI.

[PFC-3] After the successful configuration set-up, stages appear on the corresponding lists that openSF presents to users upon selection.

[PFC-4] Affected items including stages are removed from the data repository.

4.1.4. Test case TC-0035: Tool assignment

4.1.4.1. Purpose

This test case checks the capability to assign external tools to specific file types used by openSF.

4.1.4.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0030 SR-FUN-1240

4.1.4.3. Resources and Tools

N/A

4.1.4.4. Input Specification

N/A

4.1.4.5. Output Specification

An external tool is attached to one or more specific file types.

4.1.4.6. Pass/Fail Criteria

[PFC-1] The system provides a function to attach third party viewers to input/output files so that they are run during the session's execution.

[PFC-2] After the successful addition of an external tool, it appears on the corresponding tool list that openSF presents to users.

4.1.5. Test case TC-0040: I/O descriptors creation

4.1.5.1. Purpose

This test case checks the capability of openSF to create new input and output descriptors whose contents are fully compliant to the [ICD].

4.1.5.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0140 SR-FUN-0150 SR-FUN-0180 SR-OPE-0010

4.1.5.3. Resources and Tools

N/A

4.1.5.4. Input Specification

The input for the test case is constituted by the data fields that define a new IO descriptor, this is:

- Tag (identifier) that uniquely identifies the IO descriptor within the system
- General description of the IO descriptor
- Number of files
- For each file:
 - Name of the file as read/written by the model
 - Description

4.1.5.5. Output Specification

- Input and output descriptor, available from the openSF data repository.

4.1.5.6. Pass/Fail Criteria

[PFC-1] A function to add new IO descriptors into the system is available from the openSF HMI.

[PFC-2] The IO descriptors are correctly created and stored in the openSF data repository with the expected data (i.e. values from the data fields listed in the Input Specification).

[PFC-3] After the successful creation of the IO descriptors, they appear on the IO descriptor list that openSF presents to users upon selection.

4.1.6. Test case TC-0050: I/O descriptors deletion

This test case checks the capability of openSF to delete an existing I/O descriptor from the data repository.

4.1.6.1. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0160 SR-FUN-0170 SR-SEC-0010 SR-OPE-0010

4.1.6.2. Resources and Tools

N/A

4.1.6.3. Input Specification

N/A

4.1.6.4. Output Specification

No information of the deleted I/O descriptor is available from the openSF data repository.

4.1.6.5. Pass/Fail Criteria

[PFC-1] A function to delete I/O descriptors is available from the openSF HMI.

[PFC-2] The I/O descriptor to delete is selected from the list of available descriptors in openSF.

[PFC-3] Upon selection of the I/O descriptor to delete, the system presents a dialogue box requesting the user to confirm the operation.

[PFC-4] The I/O descriptor is correctly deleted and no trace is kept in openSF data repository.

4.1.7. Test case TC-0060: Model creation

4.1.7.1. Purpose

This test case checks the capability of openSF to create a new model, meaning that its input and output interfaces are fully compliant to the [ICD].

4.1.7.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0010	SR-FUN-0060	SR-FUN-0070	SR-FUN-0080
SR-FUN-0090	SR-FUN-0130	SR-FUN-0140	SR-FUN-0190
SR-INT-0020	SR-OPE-0010		

4.1.7.3. Resources and Tools

N/A

4.1.7.4. Input Specification

The input for the test case is constituted by the data fields that define a new model, this is:

- Model identifier
- Model version
- Stage
- Input and Output descriptors
- Description
- Author

- Source code
- Binary code (compiled source)
- XML file schema (.xsd), including configuration parameters that defines the model's behaviour
- Default XML configuration file (.xml), which format is according to the xsd file defined above.

4.1.7.5. Output Specification

- Model, available from openSF data repository.
- Model XML file schema and default configuration XML file located at \$OPENSF_HOME/models/<model_name>/conf directory
- Model source code and binary code located at \$OPENSF_HOME/models/<model_name>/src and \$OPENSF_HOME/models/<model_name>/bin directories respectively

4.1.7.6. Pass/Fail Criteria

[PFC-1] A function to add new models into the system is available from the openSF HMI.

[PFC-2] The model appears in the model list at the data repository frame on the left side of the openSF HMI.

[PFC-3] The model is correctly created and stored in the openSF data repository with the expected data (i.e. values of the data fields listed in the Input Specification).

[PFC-4] Model's configuration is located at \$OPENSF_HOME/models/<model_name>/conf directory

[PFC-5] Model's source and binary code are located at \$OPENSF_HOME/models/<model_name>/src and \$OPENSF_HOME/models/<model_name>/bin directories respectively.

4.1.8. Test case TC-0070: Incorrect model creation

4.1.8.1. Purpose

This test case checks the capability of openSF to fail in the creation of the model when incorrect conditions are met:

- The model does not have a unique identifier, or
- Mandatory data for the model creation have not been defined.

4.1.8.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0200 SR-FUN-0210 SR-OPE-0010

4.1.8.3. Resources and Tools

N/A

4.1.8.4. Input Specification

The input for the test case is constituted by the data fields that provoke a failure in the model creation, this is:

- Non unique Model identifier in the data repository
- Non-existent XML configuration file or binary code for the model

4.1.8.5. Output Specification

- Error message indicating the reason for the model creation failure.

4.1.8.6. Pass/Fail Criteria

[PFC-1] When one or more mandatory fields are left blank, openSF reports an error indicating the violated condition. (SR-FUN-0510)

[PFC-2] Upon a duplicated model identifier, openSF reports an error indicating such circumstance.

4.1.9. Test case TC-0080: Model edition

4.1.9.1. Purpose

This test case checks the capability of openSF to edit the information associated to an existing model.

4.1.9.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0220	SR-FUN-0230	SR-FUN-0240	SR-FUN-0250
SR-SEC-0010	SR-INT-0020	SR-OPE-0010	

4.1.9.3. Resources and Tools

N/A

4.1.9.4. Input Specification

The input for the test case is constituted by a subset of the following items, this is:

- Description
- Author

4.1.9.5. Output Specification

- Model, available from openSF data repository

4.1.9.6. Pass/Fail Criteria

[PFC-1] A function to edit models is available from the openSF HMI.

[PFC-2] The model to edit is selected from the list of available model in openSF.

[PFC-3] The information the system presents to edit a model are the description and author fields.

[PFC-4] Upon selecting the “Edit” button, the system presents a dialogue box requesting the user to confirm the operation.

[PFC-5] The model is correctly modified and stored in the openSF data repository.

4.1.10. Test case TC-0090: Model version creation

4.1.10.1. Purpose

This test case checks the capability of openSF to create a new version of an existing model.

4.1.10.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0260 SR-FUN-0270 SR-FUN-0280 SR-FUN-0290
SR-OPE-0010

4.1.10.3. Resources and Tools

N/A

4.1.10.4. Input Specification

The input for the test case is constituted by a subset of the following items, this is:

- Model version
- Author
- Description
- Source code
- Binary code (compiled source)

4.1.10.5. Output Specification

- New model version, available from the openSF data repository
- Model's version source code and binary code located at \$OPENSF_HOME/models/<model_name>/src and \$OPENSF_HOME/models/<model_name>/bin directories respectively

4.1.10.6. Pass/Fail Criteria

- [PFC-1]** A function to add new model versions into the system is available from the openSF HMI.
- [PFC-2]** The model from which a new version is to be created appears in the model list in the openSF HMI.
- [PFC-3]** The information the system presents to create a new model version is limited to: the model version, author, description, source code, and binary code.
- [PFC-4]** The model version is correctly modified and stored in the openSF data repository.
- [PFC-5]** Model's source and binary code are located at \$OPENSF_HOME/models/<model_name>/src and \$OPENSF_HOME/models/<model_name>/bin directories respectively.

4.1.11. Test case TC-0100: Incorrect model version creation

4.1.11.1. Purpose

This test case checks the capability of openSF to fail in the creation of the model version when incorrect conditions are met:

- The model identifier/version couple is not unique in the system

4.1.11.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0300 SR-OPE-0010

4.1.11.3. Resources and Tools

N/A

4.1.11.4. Input Specification

The input for the test case is constituted by a subset of the following items, this is:

- Model identifier
- Model version

for a model and software version that already exists in the system.

4.1.11.5. Output Specification

- Error message indicating the reason for the model version creation failure.

4.1.11.6. Pass/Fail Criteria

[PFC-1] Upon a duplicated model/version couple, openSF reports an error indicating such circumstance.

4.1.12. Test case TC-0110: Model deletion

4.1.12.1. Purpose

This test case checks the capability of openSF to delete an existing model from the data repository.

4.1.12.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0310 SR-FUN-0320 SR-FUN-0330 SR-SEC-0010
SR-OPE-0010

4.1.12.3. Resources and Tools

N/A

4.1.12.4. Input Specification

N/A

4.1.12.5. Output Specification

- No information of the deleted model is available from the openSF data repository.
- Model directory (\$OPENSF_HOME/models/<model_name>) no longer exists.

4.1.12.6. Pass/Fail Criteria

[PFC-1] A function to delete models is available from the openSF HMI.

[PFC-2] The model to delete is selected from the list of available models in openSF.

[PFC-3] Upon selection of the model to delete, the system presents a dialogue box requesting the user to confirm the operation.

[PFC-4] The model is correctly deleted and no trace is kept in openSF data repository or in the file system, i.e. the \$OPENSF_HOME/models/<model_name> does not exist.

4.1.13. Test case TC-0120: Simulation creation

4.1.13.1. Purpose

This test case checks the capability of openSF to create a simulation.

4.1.13.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0430	SR-FUN-0440	SR-FUN-0450	SR-FUN-0460
SR-FUN-0470	SR-FUN-0480	SR-FUN-0490	SR-INT-0010
SR-OPE-0010			

4.1.13.3. Resources and Tools

N/A

4.1.13.4. Input Specification

The input for the test case is constituted by the data fields that define a new simulation, this is:

- Simulation identifier
- Description
- Author
- Origin and target stages
- List of models participating in the simulation

4.1.13.5. Output Specification

- Simulation, available from openSF data repository.

4.1.13.6. Pass/Fail Criteria

[PFC-1] A function to add new simulations into the system is available from the openSF HMI.

[PFC-2] The input data for the simulation creation is according to the Input Specification.

[PFC-3] The simulation appears in the simulation list at the data repository frame on the left side of the openSF HMI.

[PFC-4] The simulation is correctly created and stored in the openSF data repository with the expected data (i.e. values of the data fields listed in the Input Specification).

4.1.14. Test case TC-0130: Incorrect simulation creation

4.1.14.1. Purpose

This test case checks the capability of openSF to fail in the creation of the simulation when incorrect conditions are met:

- The simulation does not have a unique identifier, or
- Mandatory data for the simulation creation have not been defined.

4.1.14.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0440 SR-FUN-0500 SR-FUN-0510 SR-OPE-0010

4.1.14.3. Resources and Tools

N/A

4.1.14.4. Input Specification

The input for the test case is constituted by the different conditions that can cause an unsuccessful simulation creation:

- Non unique Simulation identifier in the data repository
- Inconsistent origin and target stages, i.e. the order of the target stage is lower than the origin's one.
- Incomplete model specification

4.1.14.5. Output Specification

- Error message indicating the reason for the simulation creation failure.

4.1.14.6. Pass/Fail Criteria

[PFC-1] A function to add new simulations into the system is available from the openSF HMI.

[PFC-2] Upon a duplicated simulation identifier, openSF reports an error indicating such circumstance.

[PFC-3] Upon an inconsistent origin and target stage definition, openSF reports an error indicating the violated condition.

[PFC-4] The HMI only allows define the model sequence following the stage definition, configured for the openSF instance, starting from the origin state and ending at the target stage.

[PFC-5] Upon an inconsistent origin and target stage definition, openSF reports an error indicating the violated condition.

[PFC-6] openSF only permits to select one model per stage, defined for the simulation.

4.1.15. Test case TC-0140: Simulation edition

4.1.15.1. Purpose

This test case checks the capability of openSF to edit a simulation already available from the data repository.

4.1.15.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0520	SR-FUN-0530	SR-FUN-0540	SR-FUN-0550
SR-SEC-0010	SR-INT-0010	SR-OPE-0010	

4.1.15.3. Resources and Tools

N/A

4.1.15.4. Input Specification

The input for the test case is constituted by the data fields that modify an existing simulation, this is:

- Description
- Author

4.1.15.5. Output Specification

- Simulation, available from openSF data repository, with the new information.

4.1.15.6. Pass/Fail Criteria

[PFC-1] A function to edit simulations is available from the openSF HMI.

[PFC-2] The information the system presents to edit a simulation is composed by the description and author fields.

[PFC-3] Upon selecting the “Edit” button, the system presents a dialogue box requesting the user to confirm the operation

[PFC-4] The simulation is correctly modified and stored in the openSF data repository.

4.1.16. Test case TC-0150: Simulation deletion

4.1.16.1. Purpose

This test case checks the capability of openSF to delete an existing simulation from the openSF data repository.

4.1.16.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0560 SR-FUN-0570 SR-FUN-0580 SR-SEC-0010
SR-OPE-0010

4.1.16.3. Resources and Tools

N/A

4.1.16.4. Input Specification

Identifier of the simulation to delete.

4.1.16.5. Output Specification

No information of the deleted simulation is available from openSF data repository.

4.1.16.6. Pass/Fail Criteria

[PFC-1] A function to delete simulations is available from the openSF HMI.

[PFC-2] The simulation to delete is selected from the list of available simulations in openSF.

[PFC-3] Upon selection of the simulation to delete, the system presents a dialogue box requesting the user to confirm the operation.

[PFC-4] The simulation is correctly deleted and no trace is kept in the openSF data repository.

4.1.17. Test case TC-0160: Session creation

4.1.17.1. Purpose

This test case checks the capability of openSF to create a session.

4.1.17.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

	SR-FUN-0590	SR-FUN-0600	SR-FUN-0610
SR-FUN-0620	SR-FUN-0630	SR-FUN-0640	SR-FUN-0760
SR-FUN-0780	SR-FUN-0790	SR-FUN-0800	SR-FUN-0810
SR-OPE-0010			

4.1.17.3. Resources and Tools

N/A

4.1.17.4. Input Specification

The input for the test case is constituted by the following data fields:

- Session identifier
- Description
- Author
- List of simulations comprising the session
- List of model configuration files for all models of the session (if different from the default)
- List of input files (if different from the default)
- Optionally the list of tools assigned to the files involved in the session (input, output, configuration).

4.1.17.5. Output Specification

- Session, available from the openSF data repository.

4.1.17.6. Pass/Fail Criteria

- [PFC-1] A function to add new sessions into the system is available from the openSF HMI.
- [PFC-2] Consistency checks are performed when setting the parameters of the models participating in the session.
- [PFC-3] The user is able to add simulations to the session by managing the simulation list that the system presents.
- [PFC-4] The session appears in the simulation list at the data repository frame on the left side of the openSF HMI.
- [PFC-5] The session is correctly created and stored in the openSF data repository with the expected data (i.e. values of the data fields listed in the Input Specification). This includes the shell script associated to the batch execution.

4.1.18. Test case TC-0170: Incorrect session creation

4.1.18.1. Purpose

This test case checks the capability of openSF to fail in the creation of the session when incorrect conditions are met:

- The session does not have a unique identifier, or
- The simulation list contains no simulation.

4.1.18.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0650 SR-FUN-0660 SR-OPE-0010

4.1.18.3. Resources and Tools

N/A

4.1.18.4. Input Specification

The input for the test case is constituted by the different conditions that can cause an unsuccessful simulation creation:

- Non unique simulation identifier in the data repository
- Simulation list with no simulation selected

4.1.18.5. Output Specification

- Error message indicating the reason for the simulation creation failure.

4.1.18.6. Pass/Fail Criteria

[PFC-1] A function to add new sessions into the system is available from the openSF HMI.

[PFC-2] Upon a duplicated session identifier, openSF reports an error indicating such circumstance.

[PFC-3] Upon an empty simulation list, openSF reports an error indicating the violated condition.

4.1.19. Test case TC-0180: Session edition

4.1.19.1. Purpose

This test case checks the capability of openSF to edit a session already available from the data repository.

4.1.19.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0670	SR-FUN-0680	SR-FUN-0690	SR-FUN-0700
SR-FUN-0710	SR-FUN-0720	SR-FUN-0780	SR-FUN-0790
SR-FUN-0800	SR-FUN-0810	SR-SEC-0010	SR-OPE-0010

4.1.19.3. Resources and Tools

N/A

4.1.19.4. Input Specification

The input for the test case is constituted by the data fields that modify an existing session, this is:

- Description
- Author
- List of simulations
- List of corresponding model configuration files used by each model
- Global parameter file, common to all models in a simulation chain.
- List of input files
- List of output files
- New session identifier, only in case the Simulation list is changed.

4.1.19.5. Output Specification

- Simulation, available from openSF data repository, with the new information.

4.1.19.6. Pass/Fail Criteria

[PFC-1] A function to edit simulations is available from the openSF HMI.

[PFC-2] The information of the session is presented to the user with the editable fields corresponding to the Input Specification list.

[PFC-3] Consistency checks are performed when setting the parameters of the models participating in the session

[PFC-4] Upon selecting the “Edit” button, the system presents a dialogue box requesting the user to confirm the operation

[PFC-5] In case the simulation list has changed, the system requests a new identifier for the session (“Save as” capability).

[PFC-6] The session is correctly modified and stored in the openSF data repository with the expected data (i.e. values from the data fields listed in the Input Specification).

4.1.20. Test case TC-0190: Session deletion

4.1.20.1. Purpose

This test case checks the capability of openSF to delete an existing session from the data repository.

4.1.20.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0730 SR-FUN-0740 SR-FUN-0750 SR-SEC-0100
SR-OPE-0010

4.1.20.3. Resources and Tools

N/A

4.1.20.4. Input Specification

Identifier of the session to delete.

4.1.20.5. Output Specification

No information of the deleted simulation is available from openSF data repository.

4.1.20.6. Pass/Fail Criteria

[PFC-1] A function to delete sessions is available from the openSF HMI.

[PFC-2] The session to delete is selected from the list of available sessions in openSF.

[PFC-3] Upon selection of the session to delete, the system presents a dialogue box requesting the user to confirm the operation.

[PFC-4] The session is correctly deleted and no trace is kept in the openSF data repository.

4.1.21. Test case TC-0200: Session execution

4.1.21.1. Purpose

This test case checks the capability of openSF to run a session available from the data repository.

4.1.21.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0010	SR-FUN-0840	SR-FUN-0850	SR-FUN-0860
SR-FUN-0870	SR-FUN-0880	SR-FUN-0900	SR-INT-0030
SR-OPE-0010	SR-OPE-0020	SR-FUN-1250	SR-FUN-1260
SR-FUN-1290	SR-FUN-1300	SR-FUN-1330	

4.1.21.3. Resources and Tools

N/A

4.1.21.4. Input Specification

The input for the test case is constituted by the identifier of the session to run.

- Identifier of the session to run
- Model's input and configuration files, placed at the expected locations

4.1.21.5. Output Specification

- Simulation results, available from openSF:
 - General data of the executed simulation: duration, status = SUCCESSFUL
 - Log session
 - Output file list, each file located at its specified location.
- Model's output file and intermediate configuration files (if applicable) exist in \$OPENSF/sessions/<session-name>.

4.1.21.6. Pass/Fail Criteria

[PFC-1] A function to run sessions is available from the openSF HMI.

[PFC-2] During the session run, log messages are displayed on the HMI. The date and time format of each messages is YYYY-MM-DD HH:mm:ss.

[PFC-3] Upon completion, the session appears in the list of executed sessions identified by the session identifier plus the data and time it was executed.

[PFC-4] The general output parameter of the simulation indicating the status is equal to SUCCESSFUL.

[PFC-5] Outputs are generated at the specified locations. If not, openSF warns about this circumstance.

[PFC-6] The log window can be maximised.

4.1.22. Test case TC-0210: Failed session execution

4.1.22.1. Purpose

This test case checks the capability of openSF to run a session available from the data repository.

4.1.22.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0840	SR-FUN-0850	SR-FUN-0860	SR-FUN-0910
SR-INT-0030	SR-OPE-0010	SR-OPE-0020	

4.1.22.3. Resources and Tools

N/A

4.1.22.4. Input Specification

The input for the test case is constituted by the identifier of the session to run.

- Identifier of the session to run
- Model's input and configuration files are missing

4.1.22.5. Output Specification

- Simulation results, available from openSF:
 - General data of the executed simulation: duration, status = FAILED
 - Log session
- Model's output file and intermediate configuration files (if applicable) are not generated.

4.1.22.6. Pass/Fail Criteria

[PFC-1] A function to run sessions is available from the openSF HMI.

[PFC-2] openSF reports an error indicating that input files are missing.

[PFC-3] The general output parameter of the simulation indicating the status is equal to FAILED.

4.1.23. Test case TC-0220: Session abortion

4.1.23.1. Purpose

This test case checks the capability of openSF to abort a running session.

4.1.23.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0890 SR-INT-0030 SR-OPE-0010 SR-OPE-0020

4.1.23.3. Resources and Tools

N/A

4.1.23.4. Input Specification

The input for the test case is constituted by the identifier of the session to run.

- Identifier of the session to run
- Model's input and configuration files, placed at the expected locations

4.1.23.5. Output Specification

- Simulation results, available from openSF:
 - General data of the executed simulation: duration, status = FAILED
 - Log session
- Model's output file and intermediate configuration files (if applicable) are not generated.

4.1.23.6. Pass/Fail Criteria

[PFC-1] During the simulation run, log messages are displayed on the HMI. (SR-FUN-0900)

[PFC-2] A function to abort a simulation is available from the openSF HMI when the session is running.

[PFC-3] Upon the session abort, the session identifier and date and time corresponding to the session execution do not appear in the session list.

4.1.24. Test case TC-0230: Iterative session execution

4.1.24.1. Purpose

This test case checks the capability of openSF to execute iteratively an existing session. An iterative session is understood when some models are executed with some configuration parameters using more than one value, i.e. a discrete group of values or a data range, with top/bottom limits and an incremental step.

4.1.24.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0820	SR-FUN-0830	SR-FUN-0840	SR-FUN-0850
SR-FUN-0860	SR-FUN-0870	SR-FUN-0880	SR-FUN-0900
SR-INT-0030	SR-OPE-0010	SR-OPE-0020	SR-OPE-0050

4.1.24.3. Resources and Tools

N/A

4.1.24.4. Input Specification

- Identifier of the session to be iterated. This session must have at least two models with at least one parameter in each configuration file.

4.1.24.5. 4.1.30.5. Output Specification

- Session execution results, available from the data repository:
 - General data of the executed simulation: duration, status = SUCCESSFUL
 - Log session
 - Output file list
- Model's output file and intermediate configuration files (if applicable) exist in \$OPENSF_HOME/sessions/<session-name>. Different output files are generated for every model execution with different configuration.

4.1.24.6. Pass/Fail Criteria

[PFC-1] A function to run sessions is available from the openSF HMI.

[PFC-2] During the session run, log messages are displayed on the HMI.

[PFC-3] Every model instance execution (with different configuration) generates a different output file.

[PFC-4] The session appears in the list of executed sessions identified by the session identifier plus the data and time it was executed.

[PFC-5] The general output parameter of the session indicating the status is equal to SUCCESSFUL.

4.1.25. Test case TC-0240: Session script execution

4.1.25.1. Purpose

This test case checks the capability of openSF to execute a batch script associated to an existing session.

4.1.25.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-OPE-0020 SR-OPE-0030 SR-OPE-0040

4.1.25.3. Resources and Tools

N/A

4.1.25.4. Input Specification

- Session batch script, available at \$OPENSF_HOME/sessions/<session-name>.

4.1.25.5. Output Specification

- No trace of the execution is kept in the openSF database
- Log session is displayed on the screen.
- Model's output file and intermediate configuration files (if applicable) exist in \$OPENSF_HOME/sessions/<session-name>.

4.1.25.6. Pass/Fail Criteria

[PFC-1] Upon the script execution, the outputs obtained are identical from running the session in interactive mode.

4.1.26. Test case TC-0250: Session execution with breakpoints

4.1.26.1. Purpose

This test case checks the capability of openSF to schedule breakpoints in the execution of a session.

4.1.26.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0010 SR-FUN-0020 SR-FUN-0770 SR-OPE-0010

4.1.26.3. Resources and Tools

N/A

4.1.26.4. Input Specification

The input for the test case is constituted by the identifier of the session to run.

- Identifier of the session to run
- Model's input and configuration files, placed at the expected locations
- Breakpoints set for the session

4.1.26.5. Output Specification

- Simulation results, available from openSF:
 - General data of the executed simulation: duration, status = SUCCESSFUL
 - Log session
 - Output file list
- Model's output file and intermediate configuration files (if applicable) exist in \$OPENSF/sessions/<session-name>.

4.1.26.6. Pass/Fail Criteria

[PFC-1] A function to set breakpoints before and in the middle of a session execution is available from the openSF HMI.

[PFC-2] The session execution stops at the defined breakpoints.

4.1.27. Test case TC-0260: Session results visualisation

4.1.27.1. Purpose

This test case checks the capability of openSF to visualize the results associated to an existing session.

4.1.27.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0010	SR-FUN-0940	SR-FUN-0950	SR-FUN-0960
SR-FUN-0970	SR-FUN-0980	SR-FUN-1010	SR-FUN-1020
SR-OPE-0010	SR-FUN-1280	SR-FUN-1300	SR-FUN-1310
SR-FUN-1320	SR-FUN-1330		

4.1.27.3. Resources and Tools

N/A

4.1.27.4. Input Specification

- Execution's identifier including the session name and the execution's date and time.

4.1.27.5. Output Specification

N/A

4.1.27.6. Pass/Fail Criteria

- [PFC-1] A function to visualise the results of the executed sessions is available from the openSF HMI. (SR-FUN-0920, SR-FUN-1540)
- [PFC-2] Upon the selection of the option to visualise the results of a session, the HMI presents the status of the session (SUCCESSFUL, FAILED, ...), the models' results and the log session.
- [PFC-3] The messages belonging to the log session include
- Date and time of the log message generation
 - Severity level of the message (I=Info, W=Warning, E=Error)
 - Source: model and function within the model that triggers the exception
 - Message text
- [PFC-4] Log messages corresponding to models execution are stored per model.
- [PFC-5] The date and time format of log session name is YYYY-MM-DD HH:mm:SS:sss.
- [PFC-6] The date and time format of log messages is YYYY-MM-DD HH:mm:SS:sss.
- [PFC-7] The log window can be maximised.

4.1.28. Test case TC-0270: Session plots visualisation

4.1.28.1. Purpose

This test case checks the capability of openSF to plot the results associated to the models' output of an existing session.

4.1.28.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0010 SR-FUN-0990 SR-FUN-1000 SR-OPE-0010

4.1.28.3. Resources and Tools

Every product tool listed in the session definition must be present and available to execute.

4.1.28.4. Input Specification

Session output files.

4.1.28.5. Output Specification

N/A

4.1.28.6. Pass/Fail Criteria

[PFC-1] A function to visualise the results of the executed sessions is available from the openSF HMI.

[PFC-2] The HMI is able to invoke the corresponding third party tools to plot output files or compare two files.

4.1.29. Test case TC-0280: Multi-language model session execution

4.1.29.1. Purpose

This test case checks the capability of openSF to run a session containing simulations using models written in different programming languages.

4.1.29.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-RES-0040

4.1.29.3. Resources and Tools

N/A

4.1.29.4. Input Specification

The input for the test case is constituted by the identifier of the session containing models written in C, C++ and F90.

4.1.29.5. Output Specification

Simulation results, available from openSF:

- General data of the executed simulation: duration, status = SUCCESSFUL
- Log session
- Output file list

Model's output file and intermediate configuration files (if applicable) exist in \$OPENSF/sessions/<session-name>.

i.e. same outputs as "Session execution" test case.

4.1.29.6. Pass/Fail Criteria

[PFC-1] A function to run sessions is available from the openSF HMI.

[PFC-2] During the session run, log messages are displayed on the HMI.

[PFC-3] Upon completion, the session appears in the list of executed sessions identified by the session identifier plus the data and time it was executed.

[PFC-4] The general output parameter of the simulation indicating the status is equal to SUCCESSFUL.

i.e. same PFC as "Session execution" test case.

4.1.30. Test case TC-0290: Multi-platform session execution

4.1.30.1. Purpose

This test case checks the capability of openSF to run on three different software platforms: Windows XP, Mac OS and Linux.

4.1.30.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-IMP-0040

4.1.30.3. Resources and Tools

N/A

4.1.30.4. Input Specification

Any existing session, whose models have been previously compiled with success on each platform.

4.1.30.5. Output Specification

For each execution:

- Simulation results, available from openSF:
 - General data of the executed simulation: duration, status = SUCCESSFUL
 - Log session
 - Output file list
- Model's output file and intermediate configuration files (if applicable) exist in \$OPENSF/sessions/<session-name>.

4.1.30.6. Pass/Fail Criteria

[PFC-1] openSF runs on the three platforms and is able to launch the execution of the session.

[PFC-2] During the session run, the models execute successfully and log messages are displayed on the HMI.

[PFC-3] For each platform, the general output parameter of the session indicating the status is equal to SUCCESSFUL.

4.1.31. Test case TC-0300: Web access for new users

4.1.31.1. Purpose

This test case checks the registration of new users for the openSF web site access.

4.1.31.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1030 SR-FUN-1070 SR-FUN-1080 SR-FUN-1090

4.1.31.3. Resources and Tools

N/A

4.1.31.4. Input Specification

N/A

4.1.31.5. Output Specification

- User correctly registered in the openSF web site.

4.1.31.6. Pass/Fail Criteria

[PFC-1] An operation to register new users is available from the openSF web site.

[PFC-2] ESA and DEIMOS openSF team receive an e-mail with the request from a non-registered user to access the web site.

[PFC-3] User is successfully registered into the web site, and therefore can access the web site.

4.1.32. Test case TC-0310: Web access for delivery download

4.1.32.1. Purpose

This test case checks the web site access for the openSF download capability (document or openSF release).

4.1.32.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1030 SR-FUN-1060

4.1.32.3. Resources and Tools

N/A

4.1.32.4. Input Specification

N/A

4.1.32.5. Output Specification

OpenSF delivery downloaded at the users platform.

4.1.32.6. Pass/Fail Criteria

[PFC-1] The registered user logs into the openSF web site with success.

[PFC-2] An operation to download the latest openSF is available from the openSF web site.

[PFC-3] The openSF deliverable is successfully downloaded.

4.1.33. Test case TC-0320: Web access for SPR management

4.1.33.1. Purpose

This test case checks the web site access for the openSF SPR management.

4.1.33.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1030 SR-FUN-1040 SR-FUN-1050

4.1.33.3. Resources and Tools

N/A

4.1.33.4. Input Specification

N/A

4.1.33.5. Output Specification

SPR correctly registered in the openSF web site.

4.1.33.6. Pass/Fail Criteria

[PFC-1] The registered user logs into the openSF web site with success.

[PFC-2] An access to Mantis is available from the openSF web site.

[PFC-3] The SPR is correctly added in Mantis.

4.1.34. Test case TC-0330: Multi-repository capabilities

4.1.34.1. Purpose

This test case checks the capability of openSF to open different repositories within the same application instance.

4.1.34.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1100 SR-FUN-1110

4.1.34.3. Resources and Tools

Java database connector for MySQL (JDBC) included within openSF distribution.

4.1.34.4. Input Specification

A set of repositories has been previously defined within the system.

4.1.34.5. Output Specification

- New repository correctly loaded into openSF.
- This repository shall contain different simulation entities (stages, models, descriptors ,etc...).

4.1.34.6. Pass/Fail Criteria

[PFC-1] openSF is able to successfully open a pre-defined repository. This implies loading all the stored simulation entities, stages, descriptors, models etc...

[PFC-2] openSF completely closes the current repository removing all the simulation entities.

[PFC-3] openSF shows all the repositories present on the system.

4.1.35. Test case TC-0340: Multi-repository management

4.1.35.1. Purpose

This test case checks the capability of openSF to manage the different simulation repositories within the system. This includes the creation of a new empty repository or deletion of a previously defined one. This test case also covers the creation of a repository backup dumping the database info to a SQL file.

4.1.35.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1100 SR-FUN-1110

4.1.35.3. Resources and Tools

Java database connector for MySQL (JDBC) included within openSF distribution.

4.1.35.4. Input Specification

The input for this test case depends on the capability to be tested:

- Creation

N/A

- Deletion and backup

At least one repository must exist within the system.

4.1.35.5. Output Specification

Depending on the capability being tested:

- Creation

One new repository with a unique identifier. It shall be accessible from the openSF repository management interface.

- Deletion

The deleted repository is no longer present in the system and the related database has been successfully removed.

- Backup

A SQL script file containing the database structure and data shall be present within the filesystem. The file name contains the repository identifier and the backup creation date.

4.1.35.6. Pass/Fail Criteria

[PFC-1] openSF creates a new repository and it is available from the openSF repository management interface.

[PFC-2] openSF successfully builds a SQL dump file containing the database structure and data tables.

[PFC-3] openSF completely removes a repository from the system and the database. It shall be no longer available from the MMI.

4.1.36. Test case TC-0350: Model languages support

4.1.36.1. Purpose

This test case checks the capability of openSF to run a session containing simulations using models written in different programming languages. This test extends the available programming languages, IDL, Matlab, Fortran 77, and Python.

4.1.36.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-RES-0050

4.1.36.3. Resources and Tools

The models involved in this test case must make use of the openSF Integration Libraries (OSFI). These libraries are distributed within openSF package

4.1.36.4. Input Specification

- The input for the test case is constituted by the identifier of the session containing models written in Matlab, IDL and Fortran 77.

4.1.36.5. Output Specification

- Simulation results, available from openSF:
 - General data of the executed simulation: duration, status = SUCCESSFUL
 - Log session
 - Output file list
- Model's output file and intermediate configuration files (if applicable) exist in \$OPENSF/sessions/<session-name>.

i.e. same outputs as "Session execution" test case.

4.1.36.6. Pass/Fail Criteria

[PFC-1] A function to run sessions is available from the openSF HMI.

[PFC-2] During the session run, log messages are displayed on the HMI.

[PFC-3] Upon completion, the session appears in the list of executed sessions identified by the session identifier plus the data and time it was executed.

[PFC-4] The general output parameter of the simulation indicating the status is equal to SUCCESSFUL.

4.1.37. Test case TC-0360: Parameter management system

4.1.37.1. Purpose

This test case checks the availability of *Parameter Editor* tool within openSF and the ability to open configuration files.

4.1.37.2. Software Requirements Verified

The following requirements from [SRD] are covered or partially covered by this test case:

SR-FUN-1120	SR-FUN-1130	SR-FUN-1140	SR-FUN-1150
SR-FUN-1160	SR-FUN-1170	SR-FUN-0790	

4.1.37.3. Resources and Tools

Parameter Editor is in the openSF package and can be found in the framework home folder.

4.1.37.4. Input Specification

A set of configuration files from an openSF pre-defined session.

4.1.37.5. Output Specification

The *Parameter Editor* appears on screen showing all the session related configuration files.

4.1.37.6. Pass/Fail Criteria

[PFC-1] The *Parameter Editor* can be launched from the command line and It appears on screen when the execution script is launched.

[PFC-2] The *Parameter Editor* opens all the configuration files from the openSF session (global and local configuration files).

4.1.38. Test case TC-0370: Rules definition

4.1.38.1. Purpose

This test case covers the capability of creating rules that imply constraints to parameter values.

4.1.38.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1120 SR-FUN-1140 SR-FUN-1160

4.1.38.3. Resources and Tools

Parameter Editor is in the openSF package and can be found in the framework home folder.

4.1.38.4. Input Specification

Set of configuration parameters.

4.1.38.5. Output Specification

Rule Creation

New rule defined, identified by a unique code.

4.1.38.6. Pass/Fail Criteria

[PFC-1] The *Parameter Editor* successfully creates a new rule and it is stored in the correspondent file.

4.1.39. Test case TC-0380: Parameter validation

4.1.39.1. Purpose

This test case checks the system capability to validate a set of parameters against a set of rules.

4.1.39.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1120 SR-FUN-1140 SR-FUN-1160 SR-FUN-0790

4.1.39.3. Resources and Tools

Parameter Editor is in the openSF package and can be found in the framework home folder.

4.1.39.4. Input Specification

- Set of configuration files with simulation parameters.
- Rules file linking some of the parameters defined within the session configuration files.

4.1.39.5. Output Specification

Log messages showing the result of the validation process.

4.1.39.6. Pass/Fail Criteria

[PFC-1] The *Parameter Editor* provides a function to validate a set of parameters.

[PFC-2] The *Parameter Editor* performs correctly the parameter validation. The validation process is divided in two steps:

- a. Parameter consistency checking, where parameter type, dimensions, and values range are checked.
- b. Rules application, where the relations and constraints between parameters are checked.

[PFC-3] The HMI shows the log messages from the validation process.

4.1.40. Test case TC-0390: Parameter editor integration

4.1.40.1. Purpose

This test case checks that the Parameter Editor and Parameter Perturbation capabilities are integrated in openSF.

4.1.40.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1180

SR-FUN-1190

SR-FUN-1200

4.1.40.3. Resources and Tools

N/A

4.1.40.4. Input Specification

N/A

4.1.40.5. Output Specification

N/A

4.1.40.6. Pass/Fail Criteria

[PFC-1] The openSF GUI contains a menu option/button that displays the Parameter editor tool.

[PFC-2] In the session creation/edition window an option/button exists to perform the perturbation of models' configuration parameters.

4.1.41. Test case TC-0400: Parameter perturbation

4.1.41.1. Purpose

This test case checks the system capability to apply mathematical perturbation functions to configuration parameters of models so that multiple runs of these take place as part of the simulation.

4.1.41.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1200 SR-FUN-1210 SR-FUN-1220 SR-FUN-1230

4.1.41.3. Resources and Tools

N/A

4.1.41.4. Input Specification

The input for the test case is constituted by the identifier of the session to run.

- Identifier of the session to run
- Model's input and configuration files, placed at the expected locations. At least there is one model that has configuration parameters susceptible to be perturbed.
- In the window where the perturbation is created, apart from the perturbation type and associated parameters setting, the number of shots must be set (to N) so that the number of runs of the models can be checked.

4.1.41.5. Output Specification

- Model's output file and intermediate configuration files (if applicable) exist in \$OPENSF/sessions/<session-name>.
- For each model implied in the parameter perturbation, two XML files are generated with the mean and standard deviation values.

4.1.41.6. Pass/Fail Criteria

[PFC-1] The openSF GUI contains a menu option/button that displays the Parameter Perturbation window.

[PFC-2] The session has been executed without errors.

[PFC-3] For the models involved in the multiple runs:

- a. There are as many intermediate products as number of shots (i.e. N).
- b. As output, two XML files are generated corresponding to the mean and standard deviation of the all values from the intermediate products, enabling therefore the execution of next model in the simulation.

4.1.42. Test case TC-0410: MATLAB errors

4.1.42.1. Purpose

This test case checks the capability of openSF to capture messages generated by MATLAB.

4.1.42.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1270 SR-FUN-1280 SR-FUN-1340

4.1.42.3. Resources and Tools

The models involved in this test case must make use of the openSF Integration Libraries (OSFI). These libraries are distributed within openSF package.

4.1.42.4. Input Specification

- Session containing at least one model written in Matlab. This model produces an error generated by MATLAB itself.

4.1.42.5. Output Specification

- MATLAB error reported in the log.

4.1.42.6. Pass/Fail Criteria

[PFC-1] During the execution of the model containing the MATLAB error (out of memory or similar), openSF reports the error message in the model's log,

4.1.43. Test case TC-0420: Shortcuts

4.1.43.1. Purpose

This test case checks that the shortcuts defined for specific openSF activities are working correctly.

4.1.43.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1350

4.1.43.3. Resources and Tools

N/A

4.1.43.4. Input Specification

N/A

4.1.43.5. Output Specification

N/A

4.1.43.6. Pass/Fail Criteria

[PFC-1] The following operations are activated with the CTRL-letter keys.

- Ctrl+T: Tools
- Ctrl+O: Configuration
- Ctrl+P: Parameter Editor
- Ctrl+H: Help
- Ctrl+A: About openSF
- Ctrl+X: Exit openSF
- Ctrl+D: Descriptors
- Ctrl+G: Stages
- Ctrl+M: Models
- Ctrl+I: Simulations
- Ctrl+S: Sessions
- Ctrl+R: Executions Results
- Ctrl+L: Logs

4.1.44. Test case TC-0430: Backup/Restore capability

4.1.44.1. Purpose

This test case checks that openSF can perform a backup/restore of the repository from a specific option in the GUI.

4.1.44.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1360 SR-FUN-1370

4.1.44.3. Resources and Tools

N/A

4.1.44.4. Input Specification

N/A

4.1.44.5. Output Specification

SQL script including the contents of the openSF data repository.

4.1.44.6. Pass/Fail Criteria

[PFC-1] The openSF GUI contains a menu option/button to perform a backup of the data repository.

[PFC-2] The outcome of the backup operation is an SQL script containing the data repository at the moment the backup operation was invoked.

[PFC-3] The restore operation fills in the openSF data repository by loading the SQL script obtained from the backup.

4.1.45. Test case TC-0440: Import/export capability

4.1.45.1. Purpose

This test case checks the import/export capability from the openSF GUI.

4.1.45.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1380	SR-FUN-1390	SR-FUN-1400	SR-FUN-1410
SR-FUN-1420	SR-FUN-1430	SR-FUN-1440	

4.1.45.3. Resources and Tools

N/A

4.1.45.4. Input Specification

N/A

4.1.45.5. Output Specification

- SQL script including the contents of the exported session (output from the export function).
- Data repository with the contents of an exported session (output from the import function).

4.1.45.6. Pass/Fail Criteria

- [PFC-1]** The openSF GUI contains a menu option/button to perform import and export of sessions from the data repository.
- [PFC-2]** The outcome of the export operation is an SQL script including the contents of the session selected to be exported.
- [PFC-3]** The import of a previously exported SQL file adds the session it contains into the current data repository.

4.1.46. Test case TC-0450: GUI closure

4.1.46.1. Purpose

This test case checks that openSF warns the operator when closing the GUI with a running session.

4.1.46.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-OPE-0060 SR-OPE-0070

4.1.46.3. Resources and Tools

N/A

4.1.46.4. Input Specification

N/A

4.1.46.5. Output Specification

N/A

4.1.46.6. Pass/Fail Criteria

[PFC-1] When quitting openSF when a session is running, a dialogue box is presented to the user informing of such circumstance.

[PFC-2] In case the operator decides to quit, the session running is killed.

4.1.47. Test case TC-0460: Parallelization

4.1.47.1. Purpose

This test case checks the parallelisation capability of openSF.

4.1.47.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1450 SR-FUN-1460 SR-FUN-1470

4.1.47.3. Resources and Tools

N/A

4.1.47.4. Input Specification

A session including models that include perturbation functions.

4.1.47.5. Output Specification

Two outputs corresponding to the executed sessions, one with the parallelisation capability disabled and the other one with it enabled.

4.1.47.6. Pass/Fail Criteria

[PFC-1] openSF presents option to parallelise the execution of sessions

[PFC-2] The execution of a parallelised session takes shorter than the sequential one.

[PFC-3] Outputs of both runs are identical.

4.1.48. Test case TC-0470: Removal of logs from DB

4.1.48.1. Purpose

This test case checks that log messages part of log sessions are no longer stored in the database but in a dedicated file.

4.1.48.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1345

4.1.48.3. Resources and Tools

N/A

4.1.48.4. Input Specification

N/A.

4.1.48.5. Output Specification

Log file corresponding to the session execution.

4.1.48.6. Pass/Fail Criteria

[PFC-1] Log messages are stored in a file in the session's output directory.

4.1.49. Test case TC-0480: Flexible session management

4.1.49.1. Purpose

This test case checks the flexibility that openSF V3 provides in terms of:

- Selection of a sub-chain of a simulation for the session execution
- Selection of a specific version of a model participating in the simulation to be run
- By-pass the execution of a model
- Possibility to delete the intermediate data generated from the session execution

4.1.49.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1480 SR-FUN-1490 SR-FUN-1500 SR-FUN-1510
SR-FUN-1520 SR-FUN-1530

4.1.49.3. Resources and Tools

N/A

4.1.49.4. Input Specification

Session with one simulation including models with several versions.

4.1.49.5. Output Specification

Output data of the session including only the simulations final model's output.

4.1.49.6. Pass/Fail Criteria

[PFC-1] openSF allows the possibility to run a simulation from a certain point.

[PFC-2] openSF allows the possibility to select model versions for each model participating in a simulation.

[PFC-3] openSF allows the possibility to remove intermediate data during the session execution

[PFC-4] openSF allows the possibility to by-pass a model execution when defining the session to run.

4.1.50. Test case TC-0490: Model export/import

4.1.50.1. Purpose

This test case checks the capability to export/import the information relative to the participation of a model in a specific session run.

4.1.50.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1540	SR-FUN-1550	SR-FUN-1560	SR-FUN-1570
SR-FUN-1580	SR-FUN-1590	SR-FUN-1600	SR-FUN-1610

4.1.50.3. Resources and Tools

N/A

4.1.50.4. Input Specification

In case of a model export, the input data is the information of the model that has taken part in a particular session run (configuration and input data files), all available from openSF repository.

In case of a model import, the input is constituted by the compressed file obtained by the export capability.

4.1.50.5. Output Specification

In case of the model export, a compressed file with the models data files needed for its execution.

In case of the model import, the contents of the compressed file are part of openSF repository.

4.1.50.6. Pass/Fail Criteria

[PFC-1] openSF presents options to perform a model export/import when displaying a specific session.

[PFC-2] When executing a model export, openSF generates a compressed file including the model's configuration and input data files.

[PFC-3] When executing a model import, the configuration and input data files (from the compressed file) are part of the session where the import was run.

4.1.51. Test case TC-0500: Copy elements

4.1.51.1. Purpose

This test case checks the capability to copy elements through the openSF HMI.

4.1.51.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1620 SR-FUN-1630

4.1.51.3. Resources and Tools

N/A

4.1.51.4. Input Specification

Elements existing in the openSF repository.

4.1.51.5. Output Specification

Copied elements.

4.1.51.6. Pass/Fail Criteria

[PFC-1] openSF presents options to copy IO descriptors, tools, models, simulations and sessions.

[PFC-2] Elements are successfully copied in the openSF repository.

4.1.52. Test case TC-0510: Element definition from XML

4.1.52.1. Purpose

This test case checks the capability to import the definition of openSF elements from XML files.

4.1.52.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1640 SR-FUN-1650 SR-FUN-1660 SR-FUN-1670

4.1.52.3. Resources and Tools

N/A

4.1.52.4. Input Specification

Two files:

- A non-valid XML file
- XML file with openSF element definitions. Some of them include duplicated names with respect to the openSF repository they are going to be imported.

4.1.52.5. Output Specification

In the case of importing a non-valid XML file there shall be no change in the openSF repository.

In case of the importing a valid XML file the imported elements are part of openSF repository.

4.1.52.6. Pass/Fail Criteria

[PFC-1] openSF presents an option to import element definitions.

[PFC-2] When importing a non-valid XML file an acknowledgeable dialog message shall be shown to the operator stating the first detected issue in the file

[PFC-3] When importing a valid XML file elements are successfully created in the openSF repository (duplicated definitions with respect to the openSF repository shall be imported replacing the existing ones).

4.1.53. Test case TC-0520: Error generation capabilities

4.1.53.1. Purpose

This test case checks the new error generation functions susceptible to be used by openSF models.

4.1.53.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1680 SR-FUN-1690 SR-FUN-1700 SR-FUN-1710

SR-FUN-1720 SR-FUN-1730

4.1.53.3. Resources and Tools

N/A

4.1.53.4. Input Specification

Reference file (generated in Octave) with the output of the same function calls that used in the executable using the C++ OSFEG library.

4.1.53.5. Output Specification

Output report from the executable using C++ OSFEG library functions

4.1.53.6. Pass/Fail Criteria

[PFC-1] Outputs obtained from the executable fully match with the reference file, meaning that the error functions are working correctly.

4.1.54. Test case TC-0530: Migration from openSF V2 to openSF V3

4.1.54.1. Purpose

This test case checks the migration procedure from openSF V2.X to openSF V3.

4.1.54.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1345 SR-FUN-1640 SR-FUN-1650 SR-FUN-1660 SR-FUN-1670

4.1.54.3. Resources and Tools

The migration script provided in the openSF installation package shall be used.

4.1.54.4. Input Specification

An openSF V2.X instance installation containing multiple definitions of IO descriptors, tools, stages, models, simulations, sessions and parameter perturbations as well as session executions with the corresponding log messages.

An openSF V3 instance installed with empty data set.

4.1.54.5. Output Specification

Two files:

- An XML file with openSF element definitions;
- A file containing log messages.

The openSF V3 instance installation fully populated.

4.1.54.6. Pass/Fail Criteria

[PFC-1] Elements from the input openSF V2.x instance are successfully created in the openSF V3 repository.

4.1.55. Test case TC-0540: Performance improvement of openSF V3

4.1.55.1. Purpose

This test case checks the performance improvement of openSF V3 with respect to openSF V2.X (i.e. when selecting the parallelisation option and with the removal of the logs from the database). It consists of running a pre-defined E2E simulation which is part of the validation data set provided in the installation of openSF.

4.1.55.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-1345 SR-FUN-1450 SR-FUN-1460 SR-FUN-1470

4.1.55.3. Resources and Tools

N/A.

4.1.55.4. Input Specification

An openSF V2.X instance installed with the validation data set.

An openSF V3 instance installed with the validation data set.

4.1.55.5. Output Specification

- Simulation results, available from openSF V2.X, highlighting openSF overhead: (a) general data of the executed simulation: duration, status = SUCCESSFUL; (b) log session; (c) output file list (model's output file and intermediate configuration files exist in \$OPENSF/sessions/<session-name>);
- Simulation results, available from openSF V3 (with parallelisation option disabled), highlighting openSF overhead: (a) general data of the executed simulation: duration, status = SUCCESSFUL; (b) log session; (c) output file list (model's output file and intermediate configuration files exist in \$OPENSF/sessions/<session-name>);
- Simulation results, available from openSF V3 (with parallelisation option enabled), highlighting openSF overhead: (a) general data of the executed simulation: duration, status = SUCCESSFUL; (b) log session; (c) output file list (model's output file and intermediate configuration files exist in \$OPENSF/sessions/<session-name>).

4.1.55.6. Pass/Fail Criteria

[PFC-1] The overhead of executing a session in openSF V3 is smaller than with openSF V2.X.

[PFC-2] The duration of session execution in openSF V3 is shorter than in openSF V2.X.

[PFC-3] The duration of session execution in openSF V3 with parallelisation option enabled is shorter than with parallelisation option disabled.

4.1.56. Test case TC-1000: Regression Test

4.1.56.1. Purpose

This test case checks core functionalities of the framework. It consists of an installation of openSF with the validation data set and the running of a pre-defined E2E simulation.

4.1.56.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-FUN-0010	SR-FUN-0840	SR-FUN-0850	SR-FUN-0860
SR-FUN-0870	SR-FUN-0880	SR-FUN-0900	SR-INT-0030
SR-OPE-0010	SR-OPE-0020	SR-FUN-0050	SR-INS-0010

4.1.56.3. Resources and Tools

- Installer binary that automatically installs the application.

4.1.56.4. Input Specification

The input for the test case is constituted by the identifier of the session to run.

- Identifier of the session to run
- Model's input and configuration files, placed at the expected locations

4.1.56.5. Output Specification

- openSF is installed and working
- Simulation results, available from openSF:
 - General data of the executed simulation: duration, status = SUCCESSFUL
 - Log session
 - Output file list

Model's output file and intermediate configuration files (if applicable) exist in \$OPENSF/sessions/<session-name>.

4.1.56.6. Pass/Fail Criteria

- [PFC-1] openSF is installed according to the installation procedure steps explained in the Software User Manual, [SUM].
- [PFC-2] openSF contains all functionalities listed in the [SUM].
- [PFC-3] openSF provides a framework including functions to perform model management, simulation management, session management, and session result visualisation.
- [PFC-4] A function to run sessions is available from the openSF HMI.
- [PFC-5] During the session run, log messages are displayed on the HMI.

- [PFC-6] Upon completion, the session appears in the list of executed sessions identified by the session identifier plus the data and time it was executed.
- [PFC-7] The general output parameter of the simulation indicating the status is equal to SUCCESSFUL.
- [PFC-8] Launch other functions of openSF (Parameter editor, import/export capability, etc.) by executing the corresponding TP and check that the results are not affected by the latest software changes.

4.2. Inspection Case Specification

4.2.1. Inspection case IC-0010: Framework inspections

4.2.1.1. Purpose

This case performs inspections made on the openSF framework, in particular in what refers to the usage of the Java Swing technology for the HMI, and JDBC for the database connectivity.

4.2.1.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-RES-0020 SR-RES-0030 SR-IMP-0050 SR-IMP-0060

4.2.1.3. Tools

None

4.2.1.4. Input Specification

The following inputs are required to carry out the inspection:

openSF framework software source code

4.2.1.5. Output Specification

Verification report with the evidence of the inspection.

4.2.1.6. Verification Criteria

The inspection case shall be considered successful when it is guaranteed that:

- The usage of Java Swing technology has been verified for the HMI development.
- openSF data repository is based on MySQL.
- The usage of the standard (JDBC) Java programming language has been verified for the interaction with RDBMS.

4.2.2. Inspection case IC-0020: openSF code inspection

4.2.2.1. Purpose

This inspection case aims at verifying that:

- Each source code module of the openSF infrastructure source has a common header template describing terms of use, license and versioning
- The source code does not have any hard-coded parameter.

4.2.2.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-IMP-0090 SR-IMP-0100

4.2.2.3. Tools

- None

4.2.2.4. Input Specification

The following inputs are required to carry out the inspection:

- openSF framework software source code

4.2.2.5. Output Specification

- Verification report with the evidence of the inspection.

4.2.2.6. Verification Criteria

The inspection case shall be considered successful when it is guaranteed that:

- All source code modules have a common header template describing terms of use, license and versioning.
- There are no hard-coded parameter values.

4.3. Review Case Specification

4.3.1. Review case RC-0010: Development review

4.3.1.1. Purpose

This case aims at reviewing that openSF has followed the ESA Software standards and has been specified and designed using UML. Additionally, the case checks that openSF libraries can be compiled with the latest version of gcc.

4.3.1.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-IMP-0010 SR-IMP-0020 SR-IMP-0030 SR-IMP-0040

4.3.1.3. Tools

- None

4.3.1.4. Input Specification

The following inputs are required to carry out the review:

- openSF software and documentation

4.3.1.5. Output Specification

- Verification report with the review procedure and the evidence of compliance.

4.3.1.6. Verification Criteria

The review case shall be considered successful when it is guaranteed that:

- The openSF software development has followed the specific processes defined in ECSS.
- The openSF requirement analysis and design have been performed using UML by means of a dedicated CASE tool.
- openSF libraries can be compiled with the latest version of gcc 4.3.4.

4.3.2. Review case RC-0020: Security review

4.3.2.1. Purpose

- This case aims at reviewing the design of openSF in terms of security.

4.3.2.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-IMP-0070 SR-IMP-0080 SR-SEC-0020 SR-SAF-0010
SR-SAF-0080

4.3.2.3. Tools

- None

4.3.2.4. Input Specification

The following inputs are required to carry out the review:

- openSF design documentation

4.3.2.5. Output Specification

- Verification report with the review procedure and the evidence of compliance.

4.3.2.6. Verification Criteria

The case is considered successful when the review of the design guarantees that:

- Error management guarantees that error reported by a model does not prevent simulations from stopping in a controlled manner
- openSF design prevents that any application crash will not cause to have corrupted data in the database.
- openSF design permits the tool to run on Intel hardware architectures under both single and multiprocessor machines
- openSF design ensures that the tool is not constrained to any kind of user.

4.3.3. Review case RC-0030: Documentation review

4.3.3.1. Purpose

This case aims at reviewing the documentation of openSF.

4.3.3.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-MAI-0010 SR-CON-0010

4.3.3.3. Tools

None

4.3.3.4. Input Specification

The following inputs are required to carry out the review:

openSF design documentation

4.3.3.5. Output Specification

Verification report with the review procedure and the evidence of compliance.

4.3.3.6. Verification Criteria

The case is considered successful when the review of the documentation guarantees that:

- The description of the procedures for performing the backup/restore, data management and software maintenance are documented in the [AD SUM].
- The configuration aspects applicable for the project are defined in the Software Configuration Management Plan
- There is a forward and backward traceability between the user and software requirements

4.3.4. Review case RC-0040: Acceptance review

4.3.4.1. Purpose

This case reviews the acceptance of openSF on the target platform at ESTEC premises.

4.3.4.2. Software Requirements Verified

The following requirements from [SRD] are covered by this test case:

SR-VVA-0080

4.3.4.3. Tools

None

4.3.4.4. Input Specification

The following inputs are required to carry out the review:

openSF software and documentation

4.3.4.5. Output Specification

Verification report with the review procedure and the evidence of compliance.

4.3.4.6. Verification Criteria

The case is considered successful when the review of the documentation guarantees that:

openSF can be formally accepted on the target platform at ESTEC

5. TEST PROCEDURES SPECIFICATION

5.1. Test Procedure TP-0010: Installation

5.1.1. Objective

This procedure covers the execution of the following test cases:

- ❑ Test case TC-0010: Installation

5.1.2. Pre-conditions

- ❑ The Operational platform (operational machine plus the target operating system) is available.
- ❑ The Software User Manual [SUM] contains the procedure to install the software.

5.1.3. Procedure Steps

1. Install the openSF software in accordance with the installation procedures identified in the [SUM].
2. Perform a visual inspection of the openSF target directory, and check that the openSF directory structure has been correctly installed at the expected directory.
3. Perform a visual inspection of openSF COTS target directory and check that the COTS have been installed at the expected directory.
4. Run the openSF tool by launching a command shell window and typing in the following command at the \$OPENSF_HOME¹:

```
./openSF
```

5. Check that the openSF tool includes the following functionalities at the top menu and at the left frame of the tool:
 - System
 - Repository (management of descriptors, stages, models, simulations and sessions)
 - Executions

¹ For Windows, the way to open the tool is by clicking on the openSF icon placed on the desktop.

5.2. Test Procedure TP-0020: Configuration set-up

5.2.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0020: Configuration set-up
- Test case TC-0030: Configuration edition

5.2.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- There are no stages created, this is, the data repository is empty.

5.2.3. Procedure Steps

1. Create the stages of the processing chains. For this, access the “Repository” menu and select the “New stage” option from the menu bar placed on top of the HMI.
2. Enter the following information in the window that appears:

Identifier : GEOMETRY Description : Geometry stage.
--

3. Click on the Accept button.
4. Access the “Repository” menu and select the “Stages ...” option from the menu bar placed on top of the HMI. Check that the “GEOMETRY” appears on the list that is displayed and the position is 0.
5. Click on the “New Stage” button and enter the information for a new stage:

Identifier : Stage2 Description : Second stage.
--

6. Click on the Accept button.
7. Access the “Repository” menu and select the “Stages...” option from the menu bar placed on top of the HMI. Check that “Stage2” has been added to the list with the position set to 1.
8. Repeat steps 5 and 6 but for the third stage (“OSS”/“OSS stage”). When accessing the stage list, check that “OSS” has been added to the list with the position set to 2.
9. Select “Stage2” and delete it by pressing the “Delete” button. Check that the new position for “OSS” is 1.

5.3. Test Procedure TP-0030: I/O descriptor management

5.3.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0040: I/O descriptors creation
- Test case TC-0050: I/O descriptors deletion


5.3.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- The descriptors identified as “GEO-I” and “IOD2” do not exist in the system.

5.3.3. Procedure Steps

1. Create a new I/O descriptor. For this, access the “Repository” menu and select the “New descriptor” option from the menu bar placed on top of the HMI.
2. Enter the following information in the window that appears:

Identifier : GEO-I
Description : Input descriptor for the Geometry model

3. For the files, click on the  icon to specify the first file type of the descriptor. Check that the files list presents two fields to describe each file: “Default file” and “Description”.
4. Enter the following information, clicking on each field and pressing <Enter> once the information has been introduced:

Default file : InputGeo.xml
Description : Input file for the Geometry model

5. Click on the Accept button.
6. Access the “Repository” menu and select the “Descriptors ...” option from the menu bar placed on top of the HMI. Check that “GEO-I” appears on the list that is displayed.
7. Repeat the process (steps 1 to 5) to create a second IO descriptor named “IOD2” with only one file (iod2.txt).
8. Access the “Repository” menu and select the “Descriptors ...” option from the menu bar. Check that “GEO-I” and “IOD2” appear on the list that is displayed.
9. Select “IOD2” from the list and click on the “Delete” button.
10. Check that a message is displayed requesting to confirm the operation. Click on the “Yes, delete” option.
11. Check that “IOD2” descriptor does not appear in the list.

5.4. Test Procedure TP-0040: Model creation

5.4.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0060: Model creation
- Test case TC-0070: Incorrect model creation

5.4.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- The model identified as “Geometry” does not exist in the system (openSF file system and database).²
- Stages allocated to the model exist in the system, by executing TP-0020 for them.
- IO descriptors needed for the model exist in the system: “GEO-I” and “GEO-O” by executing TP-0030 for each of them. For GEO-O, the information needed is
 - Identifier : GEO-O
 - Description : Output descriptor for the Geometry model
 - Files: One file named Geometry.xml
- The environment variable \$TEST_HOME is correctly set and available.
- The data files for the model to create should exist at \$TEST_HOME/data/ directory.

5.4.3. Procedure Steps

1. Create the model. For this, access the “Repository” menu and select the “New model” option from the menu bar placed on top of the HMI.
2. Check that a multi-tabbed window appears containing the information needed to create a new model.
3. Enter the following information in the “General” tab:

Model Id	:	
Model version:	:	1.0
Description	:	Geometry model
Author	:	Tester
Type	:	Select the GEOMETRY stage from the pulldown menu presented for the field.
Source file	:	Select “\$TEST_HOME\data\src\Geometry.cpp” from the file dialogue presented for the field.
Binary file	:	Select “\$TEST_HOME\data\bin\GeometryModule ³ ” from the file dialogue presented for the field.

² The identifier of the model to create shall be referred throughout the procedure as Geometry although obviously it is applicable to any model.

4. Fill in the Default file filed for the XML Configuration file by selecting “\$TEST_HOME\data\conf\geoConf.xml”.
5. Click on the “Input/Output” tab and from the “Input Descriptor” pulldown menu and select the “GEO-I” item.
6. From the “Output descriptor” pulldown menu select the “GEO-O” item.
7. Create the model by clicking on the “Accept” button.
8. Check that an error is reported indicating that no model identifier is specified.
9. Enter the identifier of the model in the “General” tab:

Model Id : Geometry

10. Create the model by clicking on the “Accept” button.
11. Check that the model identifier “Geometry (1.0)” (model identifier plus the version in parenthesis) appears under the “Models” folder at the left frame of the HMI.

³ This name is valid for the Linux and Mac platforms. The name for Windows is GeometryModule.exe.

5.5. Test Procedure TP-0050: Model edition and deletion

5.5.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0080: Model edition
- Test case TC-0110: Model deletion

5.5.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- Test TP-0040 has been successfully executed. Therefore, “Geometry” exists in the system (database and file system).

5.5.3. Procedure Steps

1. Access the “Repository” tab from the left frame of the window and click on “Models” option.
2. Check that “Geometry” appears on the list that is displayed. Select it and click on the “Edit” button.
3. Verify that in the “General” tab the model identifier and version are displayed in read-only mode. For the rest of fields enter the following information:

Description : New Geometry description
Author : New tester

4. Modify the model by clicking on the “Accept” button.
5. Repeat step 1 and verify that the “General” tab contains the information entered in step 3.
6. Access the “Repository” tab from the left frame of the window and click on “Models” option. Delete model “Geometry” by selecting it from the list and clicking on the “Delete” button. Confirm the operation when requested by the HMI.
7. Check that the model “Geometry” is no longer displayed under the models folder from the left frame.⁴

⁴ If the model is already part of simulations and sessions, the deletion of the model provokes also the automatic deletion of all these items.

5.6. Test Procedure TP-0060: Model version creation

5.6.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0090: Model version creation
- Test case TC-0100: Incorrect model version creation

5.6.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- Test TP-0040 has been successfully executed. Therefore, “Geometry (1.0)” exists in the openSF database and file system.
- The environment variable \$TEST_HOME must be correctly set and available.
- The data files for the model version to create should exist at \$TEST_HOME/data/ directory.

5.6.3. Procedure Steps

1. Access the “Repository” tab from the left frame of the window and click on the “Models” option.
2. Select “GeometryModule (1.0)” from the list and select the “New model version” option when pressing the mouse right button..
3. Check that a window appears containing the information of the model created in TP-0040 but with the model version automatically increased.
4. Change back the model version to 1.0 and click on the “Accept” button.
5. Check that an error is reported by the system indicating that the version already exists for the model.
6. Enter the following information in the “General” tab:

Model version : 2.0
Author : Tester
Source file : Select “\$TEST_HOME\data\Geometry_20.cpp” (or the precise location where the source code file is) from the file dialogue presented for the field.
Binary file : Select “\$TEST_HOME\data\Geometry_20.bin” (or the precise location where the binary file is) from the file dialogue presented for the field.

7. Create the model by clicking on the “Accept” button.
8. Check that the new version of the model appears under the model family (or stage) identified as “GeometryModule (2.0)” at the left frame of the HMI.

5.7. Test Procedure TP-0070: Simulation creation

5.7.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0120: Simulation creation
- Test case TC-0130: Incorrect simulation creation

5.7.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- The simulation identified as “Sim01” does not exist in the system (openSF file system and database).⁵
- 2 models already exist in the data repository with the following information:
 - Geometry:
 - Stage – GEOMETRY
 - Input descriptor: GEO_I
 - Output descriptor: GEO_O
 - ObservingSystem:
 - Stage – OSS.
 - Input descriptor: GEO_O (same as Geometry’s output descriptor)
 - Output descriptor: OSS_O, including one XML file named Instrument.xml.

Thus, the models are logically connected by sharing a common descriptor.

5.7.3. Procedure Steps

1. Create the simulation. For this, access the “Repository” menu and select the “New simulation” option from the menu bar placed on top of the HMI.
2. Check that a window appears containing the information needed to create a new simulation.
3. Enter the following information in the “General properties” area:

Identifier	:	
Author	:	Tester
Description	:	Simulation with Geometry and OSS models
Start Stage	:	Select “GEOMETRY” from the pulldown menu that is presented for the field.
Stop Stage	:	Select “OSS” from the pulldown menu that is presented for the field.

4. From the “Simulation stages” area check that:

⁵ The identifier of the simulation to create shall be referred throughout the procedure as sim01 although it can be applicable to any simulation.

- The simulation stage selected is “GEOMETRY” (i.e. the same as the Start Stage defined in step 3)
 - The “Models available” table presents the list of models associated to “Geometry”. Check that “Geometry (1.0)” is part of the list.
 - The “Current Simulation Definition” list presents no models.
5. Select “Geometry (1.0)” from the “Models available” table and click on the “Next stage” button.
 6. Check that:
 - The simulation stage selected is “OSS”
 - The “Models available” table presents the list of models associated to “OSS”. Check that “ObservingSystem (1.0)” is part of the list.
 - The “Current Simulation Definition” list presents “Geometry” appearing under the “Geometry” node.
 7. Select “ObservingSystem (1.0)” from the “Models available” table and click on the “Next stage” button.
 8. Check that the models selected on the previous stages are collected in the final list presented in the “Current simulation definition” area. Click on the “Accept” button to create the simulation.
 9. Check that an error is reported indicating that no simulation identifier is specified.
 10. Enter the identifier of the simulation in the “General properties” area:

Identifier : Sim01
 11. Create the simulation by clicking on the “Accept” button.
 12. Check that the simulation identifier “sim01” appears under the “Simulations” folder at the left frame of the HMI. Double-click on it so that the window containing the information of the simulation is displayed.
 13. Check that the simulation contains all the information set in previous steps.

5.8. Test Procedure TP-0075: Stage management with models and simulations

5.8.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0020: Configuration set-up
- Test case TC-0030: Configuration edition

5.8.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- Models and simulations associated to the existing stages exist in the openSF data repository.

5.8.3. Procedure Steps

1. Access the list of stages by clicking the “Repository” menu on top of the HMI and selecting the “Stages” option.
2. Select one of the existing stages and press the “Delete” button. Check that an error is displayed informing that the stage cannot be removed.
3. Create a new stage filling in the following information in the window that appears upon the selection of the “New stage” button:

Identifier : StageX Description : Last stage.
--

4. Click on the Accept button.
5. Check that “StageX” appears at the end of the list.
6. Select “StageX” from the list and press the “Up” and “Down” buttons. Check that an error is displayed informing that the stage cannot be moved.
7. Click on the “Edit” button.
8. Check that a new window appears with the information of the selected stage. Enter the following information in the window that appears:

Description : New description for stage X.
--

9. Click on the Accept button.
10. Check that the new description appears on the stages list.
11. Select “StageX” from the list and press the “Delete” button.
12. Check that a dialog window is presented to the user requesting a confirmation of the stage deletion. Press the “Yes” button to confirm the deletion.
13. Check that the stage is successfully deleted.

5.9. Test Procedure TP-0080: Simulation edition and deletion

5.9.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0140: Simulation edition
- Test case TC-0150: Simulation deletion

5.9.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- Test TP-0070 has been successfully executed. Therefore, “Sim01” exists in the system (openSF file system and database).

5.9.3. Procedure Steps

1. Access the “Repository” tab from the left frame of the window and click on “Simulations” option.
2. Check that “Sim01” appears on the list that is displayed. Select it and click on the “Edit” button.
3. Check that a window appears containing the information of the simulation created in TP-0070.
4. Verify that in the “General properties” the simulation identifier is displayed in read-only mode. For the rest of fields enter the following information:

Description	: New description for the simulation with Geometry and OSS models
Author	: New Tester

5. Modify the simulation by clicking on the “Accept” button.
6. Repeat steps 1 to 3 and verify that the “General properties” area contains the information entered in step 4.
7. Execute TP-0070 to create a new simulation with exactly the same information as “Sim01” but with a new identifier called “Sim02”.
8. Delete the simulation “Sim02” by selecting the simulation from the left frame and selecting the “Delete” option when clicking on the mouse right button. Confirm the operation when requested by the system.
9. Check that the simulation “Sim02” is no longer displayed under the simulations folder from the left frame.⁶

⁶ If the simulation is already part of sessions, the deletion of the simulation provokes also the automatic deletion of all these items.

5.10. Test Procedure TP-0085: Tool assignment

5.10.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0035: Tool assignment

5.10.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.

5.10.3. Procedure Steps

1. Associate an external tool to the XML files used in openSF. For that, click on the System->Tools->NewTools option.
2. Enter the following information in the window that is displayed

Identifier	: gedit ⁷
Description	: XML file editor
Action	: Edit XML files
Extension	: xml, hdf
Executable	: /usr/bin/gedit

3. Click on the Accept button and check that the tool has been inserted in the tool list by accessing the System->Tools option.
4. Access an XML file from the File system menu on the top left frame of the GUI. Keep the mouse pointer on top of the file name.
5. Click on the right button of the mouse and check that the tool created in step 2 appears in the context menu.

⁷ Gedit is the default text editor for the Linux platform with gnome desktop. For Windows, Notepad can be used, and for Mac the TextEdit tool.

5.11. Test Procedure TP-0090: Session creation

5.11.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0160: Session creation
- Test case TC-0170: Incorrect session creation

5.11.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- The session identified as “Session01” does not exist in the system (openSF file system and database).⁸
- The simulations to be part of the session should exist in the system. In this case there is only one simulation so test TP-0070 has been successfully executed for the simulation that composes the session.
- The environment variable \$TEST_HOME must be correctly set and available.
- Input data and configuration files for the session exist in \$TEST_HOME/data/input. These data is constituted by the input data files needed for the simulation to be run plus the configuration files: one for the global configuration and one for each model.

5.11.3. Procedure Steps

1. Create the session. For this, access the “Repository” menu and select the “New session” option from the menu bar placed on top of the HMI.
2. Check that the system presents a window asking the user to create a new session or to import the settings from an existing folder. Click on the “Create new” button.
3. Check that a window appears containing the information needed to create a new session.
4. Enter the following information in the “General properties” area:

Identifier :
Description : Session for testing purposes.
Author : Tester

5. Click on the “Accept” button to create the session.
6. Check that an error is reported indicating that no session identifier is specified.
7. Enter the identifier of the session in the “General properties” area:

Identifier : Session01

8. Click on the “Accept” button to create the session.

⁸ The identifier of the session to create shall be referred throughout the procedure as session01 although it can be applicable to any session.

9. Check that a dialog is shown indicating that no simulations have been defined as part of the session. Press “No”.
10. From the “Simulations set” area add as the simulations to the session by using the “Add” button and selecting the corresponding simulation (“sim01”)
11. Load the input data files for each simulation by clicking on the “Input” tab from the “Session setup” area.
12. For each input file with “Missing” status, double click on the “File instance”. Check that a file dialogue is displayed requesting the input file. Select the corresponding file from \$TEST_HOME/data/input directory.
13. Access the “Configuration” tab from the “Session setup” area. For each configuration file existing in \$TEST_HOME/data/input, double click on the “File instance” of the corresponding model. Check that a file dialogue is displayed requesting the input file. Select the corresponding file from \$TEST_HOME/data/conf directory:

Global/Model	Identifier
Global	GlobalConfiguration.xml
Geometry	geoConf.xml
ObservingSystem	ossConf.xml

14. Schedule the execution of a product tool to a certain file of the session. In this case, an XML editor shall display the contents of the geoConf.xml file at the end of the session execution. For that, right-click on the geoConf.xml file and select the “Schedule” option.
15. Create the session by clicking on the “Accept” button.
16. Check that the session identifier “Session01” appears under the “Sessions” folder at the left frame of the HMI. Double-click on it so that the window containing the information of the simulation is displayed.
17. Check that the session contains all the information set in the previous steps.

5.12. Test Procedure TP-0100: Session edition and deletion

5.12.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0180: Session edition
- Test case TC-0190: Session deletion

5.12.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- Test TP-0090 has been successfully executed. Therefore, "Session01" exists in the system (openSF file system and database).

5.12.3. Procedure Steps

1. Access the "Repository" tab from the left frame of the window and click on "Sessions" option.
2. Check that "Session01" appears on the list that is displayed. Select it and click on the "Edit" button.
3. Check that a window appears containing the information of the session created in TP-0090.
4. Verify that in the "General properties" the simulation identifier is displayed in read-only mode. For the rest of fields enter the following information:

Description	: New description for the session
Author	: New Tester

5. Modify the session by clicking on the "Accept" button.
6. Repeat steps 1 to 3 and verify that the "General properties" area contains the information entered in step 4.
7. Execute TP-0090 to create a new session with exactly the same information as "Session01" but with a new identifier called "Session02".
8. Access the "Repository" tab from the left frame of the window and click on "Sessions" option.
9. Delete the session "Session02" by selecting it from the list and clicking on the "Delete" button. Confirm the operation when requested by the HMI.
10. Check that the session "Session02" is no longer displayed under the "Sessions" folder from the left frame.

5.13. Test Procedure TP-0110: Session execution

5.13.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0200: Session execution
- Test case TC-0260: Session results visualisation

5.13.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- Test TP-0090 has been successfully executed. Therefore, “Session01” exists in the system (openSF file system and database).⁹
- A tool has been created in openSF. In this case the generic text editor gedit¹⁰ has been associated to text/XML files.
- All input data and configuration files specific for this test are available.

5.13.3. Procedure Steps

1. Access the “Repository” tab from the left frame of the window. Select the “Sessions→Session01” option, press the mouse’s right button and select the “Run” option.
2. Check that a new window is displayed containing the following information:
 - The window title includes the running identifier composed by the session identifier plus the data and time the simulation was launched;
 - The session progress;
 - The model progress;
 - The list of log messages that are produced, each one with the date and time with the format YYYY-MM-DD HH:mm:ss:sss;
 - A button to abort the execution.
3. Wait until the simulation session ends, and keep track of the time spent for the execution. In the meantime, check that the log window can be maximised for a better reading.
4. Check that the XML editor is launched with the contents of the geoConf.xml file.
5. Access the “Executions” tab from the left frame of the main window and check that a new session exists with the same simulation session identifier from step 2 (plus the date and time the execution began as YYYY-MM-DD HH:mm:ss:sss).

⁹ The identifier of the session to run shall be referred throughout the procedure as session01 although it can be applicable to any session appearing in the session list.

¹⁰ Gedit is the default text editor for the Linux platform with gnome desktop. For Windows, Notepad can be used, and for Mac the TextEdit tool.

6. Double click on this new session identifier (or alternatively select it, press mouse's right button and click on the View option) and check that a new window is displayed.
7. Check that the window has the following contents:
 - In the "General properties" area:
 - The "Identifier", "Description" and "Author" fields contain the same information as in the simulation window.
 - The "Date/Time" field contains the same date and time that is part of the session identifier, and the "Duration" field contains approximately the same time period as the duration obtained in step 3.
 - The status of the simulation result is "Successful".
 - In the "Session setup" area:
 - All files contained in the "Output" tab have the "Available" status.
 - The log messages appearing in the "Log" tab are the same as those produced during the execution.
8. Check that the output files organised per model in a dedicated directory.

5.14. Test Procedure TP-0120: Failed session execution

5.14.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0210: Failed session execution

5.14.2. Pre-conditions

- Same as TP-0110, except that for the data and configuration files, not all of them are available.

5.14.3. Procedure Steps

1. Access the "Repository" tab from the left frame of the window. Select the "Sessions->E2E_test_session" option, press the mouse's right button and select the "Run" option..
2. Check that openSF displays a message indicating that files are missing, and requests the user to either continue or cancel the execution. If not, enter a wrong path in the Session Edition window (change Input Files path to "wrong" and press enter).
3. Click on the option to continue the execution.
4. Check that the session terminates the session's execution, as one of the models cannot read the inputs.¹¹

¹¹ This should be the nominal behaviour of the model's execution although DEIMOS is not responsible for the models' implementation.

5.15. Test Procedure TP-0130: Session plots visualisation

5.15.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0270: Session plots visualisation

5.15.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- Test TP-0090 has been successfully executed. Therefore, “Session01” exists in the system (openSF file system and database).¹² The session has generated output files susceptible to be open with visualisation tools.
- A tool associated to openSF files exists in the system. In this case the generic text editor gedit¹³ has been associated to text/XML files generated by the simulation execution

5.15.3. Procedure Steps

1. Access the “Executions” tab from the left frame of the window. Select the “Results→Session01 (date-time)” element and right-click on it.
2. Check that a pop-up menu appears and contains a “View” option.
3. Select this option.
4. Check that a new window appears containing all the information about the execution of that session.
5. Select the “Output” tab and unfold the tree structure to locate a text file and right-click on it to visualise it with the generic text editor.

¹² The identifier of the session to run shall be referred throughout the procedure as session01 although it can be applicable to any session appearing in the session list.

¹³ Gedit is the default text editor for the Linux platform with gnome desktop. For Windows, Notepad can be used, and for Mac the TextEdit tool.

5.16. Test Procedure TP-0140: Abort session execution

5.16.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0220: Session abortion

5.16.2. Pre-conditions

- Same as TP-0110.

5.16.3. Procedure Steps

1. Access the “Repository” tab from the left frame of the window. Select the “Sessions→Session01” option, press the mouse’s right button and select the “Run” option.
2. Check that a new window is displayed containing the following information:
 - The window title includes the running identifier composed by the session identifier plus the data and time the simulation was launched;
 - The session progress;
 - The model progress;
 - The list of log messages that are produced;
 - A button to abort the execution.
3. Check that a new window is displayed including the “Abort” button.
4. Click on the “Abort” button to abort the execution before the execution finishes.
5. Check that a confirmation dialogue box is displayed for the operation. Confirm the abortion of the execution.
6. Access the “Executions” tab from the left frame of the main window and check that a new session exists with the same simulation session identifier.
7. Double click on this new session identifier (or alternatively select it, press mouse’s right button and click on the View option) and check that a new window is displayed.
8. Check in the “General properties” area that the status of the simulation result is “Aborted”.

5.17. Test Procedure TP-0150: Iterative session execution

5.17.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0230: Iterative session execution

5.17.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- “E2E_test_session” exists in the system (openSF file system and database).¹⁴. This session must have at least two models with at least one parameter in each configuration.

5.17.3. Procedure Steps

1. Access the “Repository” tab from the left frame of the window. Select the “Sessions->E2E_test_session” option, press the mouse’s right button and select the “Edit” option.
2. Check that a window is displayed containing the information about the selected session.
3. Access the “Parameters” on the bottom part of the window and select two parameters from one model (selection is possible left-clicking over a parameter identifier while pressing the control button).
4. Press the “Iterate parameters” button in the top-right corner.
5. Check that a new dialog is displayed containing the selected parameters.
6. Alter every value data field entering two valid values (with the same type of the parameter) separated by commas or blank spaces.
7. Preview the combination of the parameters pressing the “Preview” button. A tree showing the new session is displayed in the right-most side of the dialog.
8. Accept the combination by pressing the “Accept” button. This dialog must be closed now.
9. Press the “Run” button to start the execution.
10. Press the “Execute all” button.
11. Check that a new window is displayed containing the following information:
 - The window title includes the running identifier composed by the session identifier plus the data and time the simulation was launched;
 - The session progress;
 - The model progress;
 - The list of log messages that are produced;
 - A button to abort the execution.

¹⁴ The identifier of the session to run shall be referred throughout the procedure as session001 although it can be applicable to any session under the “Sessions” node.

Note that since openSF version 3 a session execution window appears per each iteration due to the new approach in the Session Management and Parallelization.

12. Wait until the simulation session ends, and keep track of the time spent for the execution.
13. Check that the final result of the session execution is “Successful”.
14. Access the “Executions” tab from the left frame of the main window and check that a set of new sessions exist with the same session identifier from step 2 (plus the date and time the execution began in parenthesis).
15. Double click on this new session identifier (or alternatively select it, press mouse’s right button and click on the View option) and check that a new window is displayed.
16. Check that the window has the following contents:
 - In the “General properties” area:
 - The “Identifier”, “Description” and “Author” fields contain the same information as in the simulation window.
 - The “Date/Time” field contains the same date and time that is part of the session identifier, and the “Duration” contains the time duration of the execution.
 - The status of the simulation result is “Successful”.
 - In the “Session setup” area:
 - All files contained in the “Output” tab have the “Available” status. There is at least one output file generated by each model execution with different parameter values.
 - The log messages appearing in the “Log” tab are the same as those produced during the execution.

5.18. Test Procedure TP-0160: Session script execution

5.18.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0240: Session script execution

5.18.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- Test TP-0090 has been successfully executed. Therefore, “session01” exists in the system (openSF file system and database).¹⁵
- Test TP-0150 has been successfully executed for the same session as the one for this procedure.
- The environment variable \$TEST_HOME must be correctly set and available.
- A script file exists for the session in the \$OPENSF_HOME/sessions/session01 (<date and time>) directory.
- Every input and configuration file is present and available in the directories specified in the session definition.

5.18.3. Procedure Steps

17. Open a shell tool of the Operating System.
18. Change the current directory to locate the \$OPENSF_HOME/sessions/session01 (<date and time>) directory, check the existence of the script file and execute it.
19. Check that the execution of the script file repeats the actions taken in the interactive session execution (TP-0090).
20. Check that log messages are shown in the standard output and the output files are successfully generated and identical as those generated in TP-0150.

¹⁵ The identifier of the session to run shall be referred throughout the procedure as session01 although it can be applicable to any session appearing in the session list.

5.19. Test Procedure TP-0170: Session execution with breakpoints

5.19.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0250: Session execution with breakpoints

5.19.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- Test TP-0090 has been successfully executed. Therefore, “session01” exists in the system (openSF file system and database).¹⁶
- All input data and configuration files specific for this test are available.

5.19.3. Procedure Steps

1. Set the breakpoint in a simulation of the session to run. For this, access the “Repository” tab from the left frame of the window and click on “Sessions” option.
2. Check that “session01” appears on the list that is displayed. Select it and click on the “Edit” button.
3. Access the simulation area and select the model where the breakpoint wants to be set.
4. Click on the “Accept” button to save the changes.
5. Edit the session “session01” again. Check that the system shows that the selected model has a breakpoint set.
6. Run the session and wait until the execution stops. Check that session has stopped at the end of the model selected in step 3.
7. Resume the session execution by pressing the “Resume” button. Wait until the execution ends.

¹⁶ The identifier of the session to run shall be referred throughout the procedure as session01 although it can be applicable to any session appearing in the session list.

5.20. Test Procedure TP-0180: Web access for new users

5.20.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0300: Web access for new users

5.20.2. Pre-conditions

- Email of the ESA member in charge of authorising the user registration is set.

5.20.3. Procedure Steps

1. Open a web browser and access the web site entering the following URL:

```
http://opensf.deimos-space.com
```

2. Check that the wiki page for openSF is displayed, including an option for the registration of new users. Click on this link.
3. Check that a new page is displayed presenting a dedicated form with the needed data files for the user registration.
4. Fill in the fields with the following data, and click on the “Accept” button:

```
Name      : myname  
Last name : mylastname  
Login     : user99  
Country  : Spain  
Organisation : DEIMOS  
email    : ricardo.moyano@deimos-space.com
```

5. Check that the ESA user and DEIMOS openSF team responsible for the user authorisation receive an e-mail with the request for registration with the information provided by the user in step 4.

5.21. Test Procedure TP-0190: Web access for delivery download

5.21.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0310: Web access for delivery download

5.21.2. Pre-conditions

- A registered user accesses the openSF web site
- At least one openSF delivery exists for download.

5.21.3. Procedure Steps

1. Open a web browser and access the web site entering the following URL:

`http://opensf.deimos-space.com`

2. Check that the wiki page for openSF is displayed, including the option to download openSF deliveries.
3. Click on the option to download an openSF release. Check that the web site requests the user Login/password couple to proceed with the download.
4. Fill in the fields with the valid user's data.
5. Check that the download proceeds requesting the user to specify the destination folder.
6. Check that the openSF release has been downloaded at the user's target folder.

5.22. Test Procedure TP-0200: Web access for SPR management

5.22.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0320: Web access for SPR management

5.22.2. Pre-conditions

- A registered user accesses the openSF web site

5.22.3. Procedure Steps

- Open a web browser and access the web site entering the following URL:

```
http://opensf.deimos-space.com
```

- Check that the link to Mantis bug reporting exists in the wiki page.
- Click on the option to access Mantis. Check that the web site requests the user Login/password couple to proceed with the download.
- Fill in the fields with the valid user's data.
- Check that the list of current SPR (named Issues) is presented to the user. Click on the option to add a new Issue.
- Check an empty form is displayed so that the user can describe the SPR. Fill in the fields with the following data, and click on the "Accept" button:

```
Id          : SPR01
Title       : Functionality X does not work
Description : Description why functionality X does not work
Severity    : major
Author      : RIMM
```

- Check that the SPR has been added to the list of Issues.

5.23. Test Procedure TP-0210: IDL model creation

5.23.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0060: Model creation
- Test case TC-0350: Model languages support

5.23.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- The model identified as “IDLModel” does not exist in the system (openSF file system and database).
- Stages allocated to the model exist in the system, by executing TP-0020 for them.
- IO descriptors needed for the model exist in the system: “InputGeneric” and “OutputGeneric” by executing TP-0030 for each of them.

For InputGeneric, the information needed is

- Identifier : InputGeneric
- Description : Generic input for IDL and Matlab tests
- Files: Four files named input1, input2, input3, input4

For OutputGeneric, the information needed is:

- Identifier : OutputGeneric
- Description : Generic output for IDL and Matlab tests
- Files: Four files named output1, output2, output3, output4

- The environment variable \$TEST_HOME is correctly set and available.
- The data files for the model to create should exist at \$TEST_HOME/data/ directory.
- A licensed version of IDL (runtime, development or virtual machine) is not required for this step.

5.23.3. Procedure Steps

1. Create the model. For this, access the “Repository” menu and select the “New model” option from the menu bar placed on top of the HMI.
2. Check that a multi-tabbed window appears containing the information needed to create a new model.
3. Enter the following information in the “General” tab:

Model Id	: IDLModel
Model version:	1.0
Description	: IDL model
Author	: Tester

Stage : Select the “IDLStage” stage from the pulldown menu that is presented for the field.

Binary file : Select “\$TEST_HOME\bin\test_IDL.pro” or “\$TEST_HOME\bin\test_idl.sav” from the file dialogue presented for the field.

4. Fill in the Default file field for the XML Configuration file by selecting “\$TEST_HOME\data\conf\exampleFile.xml”.
5. Click on the “Input/Output” tab and from the “Input Descriptor” pulldown menu and select the “InputGeneric” item.
6. From the “Output descriptor” pulldown menu select the “OutputGeneric” item.
7. Create the model by clicking on the “Accept” button.
8. Check that the model identifier “IDLModel (1.0)” (model identifier plus the version in parenthesis) appears under the “Models” folder at the left frame of the HMI.

5.24. Test Procedure TP-0220: Session execution with models in IDL

5.24.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0120: Simulation creation
- Test case TC-0160: Session creation
- Test case TC-0200: Session execution
- Test case TC-0350: Model languages support

5.24.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- Test TP-0210 has been successfully executed. Therefore “IDLModel (1.0)” exists in the system. It can be found under Models folder in the left frame of the HMI.
- IDL version 6.2 or later is available and accessible through the system path.
- A licensed version of IDL (runtime, development or virtual machine) is not mandatory for this test.
- All input data and configuration files specific for this test are available.

5.24.3. Procedure Steps

Simulation Creation

1. Create a new simulation “IdlSimulation” following the instructions in TP-0070:
2. Create the simulation. For this, access the “Repository” menu and select the “New simulation” option from the menu bar placed on top of the HMI.
3. Check that a window appears containing the information needed to create a new simulation.
4. Enter the following information in the “General properties” area:

Identifier	: IdlSimulation
Author	: Tester
Description	: Simulation for IDL model
Start Stage	: Select “IDLStage” from the pulldown menu that is presented for the field.
Stop Stage	: Select “IDLStage” from the pulldown menu that is presented for the field.

5. From the “Simulation stages” area check that:
 - The simulation stage selected is “IDLStage”
 - The “Models available” table presents the list of models associated to “IDL”. Check that “IDLModel (1.0)” is part of the list.
 - The “Current Simulation Definition” list presents no models.

6. Select “IDLModel (1.0)” from the “Models available” table and click on the “Next stage” button.
7. Check that the model selected on the previous stage is collected in the final list presented in the “Current simulation definition” area. Click on the “Accept” button to create the simulation.
8. Check that the simulation identifier “IDLSimulation” appears under the “Simulations” folder at the left frame of the HMI. Double-click on it so that the window containing the information of the simulation is displayed.
9. Check that the simulation contains all the information set in previous steps.

Session Creation

10. Create a new session “IDLSession” following the instructions in TP-0090.
11. Create the session. For this, access the “Repository” menu and select the “New session” option from the menu bar placed on top of the HMI.
12. Check that the system presents a window asking the user to create a new session or to import the settings from an existing folder. Click on the “Create new” button.
13. Check that a window appears containing the information needed to create a new session.
14. Enter the following information in the “General properties” area:

```
Identifier : IDLSession
Description : Session for testing purposes.
Author : Tester
```

15. From the “Simulations set” area add as the simulations to the session by using the “Add” button and selecting the corresponding simulation (“IDLSimulation”)
16. Load the input data files for each simulation by clicking on the “Input” tab from the “Session setup” area. (In this example is not necessary, because the IDL model does not need input files)
17. Access the “Configuration” tab from the “Session setup” area. For each configuration file existing in \$TEST_HOME/data/conf/, double click on the “File instance” of the corresponding model. Check that a file dialogue is displayed requesting the input file. Select the corresponding file from \$TEST_HOME/data/conf directory:

Global/Model	Identifier
Global	GlobalConfiguration.xml
IdlModel	exampleFile.xml

18. Create the session by clicking on the “Accept” button.
19. Check that the session identifier “IDLSession” appears under the “Session” folder at the left frame of the HMI. Double-click on it so that the window containing the information of the simulation is displayed.
20. Check that the session contains all the information set in the previous steps.

Session Execution

21. Access the “Repository” tab from the left frame of the window. Select the “Sessions→IDLSession” option, press the mouse’s right button and select the “Run” option.

22. Check that openSF displays a message indicating that files are missing, and requests the user to either continue or cancel the execution.
23. Click on the option to continue the execution.
24. Check that a new window is displayed containing the following information:
 - The window title includes the running identifier composed by the session identifier plus the data and time the simulation was launched;
 - The session progress;
 - The model progress;
 - The list of log messages that are produced;
 - A button to abort the execution.
25. Wait until the simulation session ends, and keep track of the time spent for the execution.
26. Access the “Executions” tab from the left frame of the main window and check that a new session exists with the same simulation session identifier from step 2 (plus the date and time the execution began).
27. Double click on this new session identifier (or alternatively select it, press mouse’s right button and click on the View option) and check that a new window is displayed.
28. Check that the window has the following contents:
 - In the “General properties” area:
 - The “Identifier”, “Description” and “Author” fields contain the same information as in the simulation window.
 - The “Date/Time” field contains the same date and time that is part of the session identifier, and the “Duration” field contains approximately the same time period as the duration obtained in step 3.
 - The status of the simulation result is “Successful”.
 - In the “Session setup” area:
 - All files contained in the” Output” tab have the “Available” status.
 - The log messages appearing in the “Log” tab are the same as those produced during the execution.

5.25. Test Procedure TP-0230: Matlab model creation

5.25.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0060: Model creation
- Test case TC-0350: Model languages support

5.25.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- The model identified as “MatlabModel” does not exist in the system (openSF file system and database).
- Stages allocated to the model exist in the system, by executing TP-0020 for them.
- IO descriptors needed for the model exist in the system: “InputGeneric” and “OutputGeneric” by executing TP-0030 for each of them.

For InputGeneric, the information needed is

- Identifier : InputGeneric
- Description : Generic input for IDL and Matlab tests
- Files: Four files named input1, input2, input3, input4

For OutputGeneric, the information needed is:

- Identifier : OutputGeneric
- Description : Generic output for IDL and Matlab tests
- Files: Four files named output1, output2, output3, output4

- The environment variable \$TEST_HOME is correctly set and available.
- The data files for the model to create should exist at \$TEST_HOME/data/ directory.
- A licensed version of Matlab is not required for this step.

5.25.3. Procedure Steps

1. Create the model. For this, access the “Repository” menu and select the “New model” option from the menu bar placed on top of the HMI.
2. Check that a multi-tabbed window appears containing the information needed to create a new model.
3. Enter the following information in the “General” tab:

Model Id	: MatlabModel
Model version:	1.0
Description	: Matlab model

Author : Tester
Stage : Select the MatlabStage stage from the pulldown menu that is presented for the field.
Binary file : Select “\$TEST_HOME\bin\testModel.m” from the file dialogue presented for the field.

4. Fill in the Default file field for the XML Configuration file by selecting “\$TEST_HOME\data\conf\exampleFile.xml”.
5. Click on the “Input/Output” tab and from the “Input Descriptor” pulldown menu and select the “InputGeneric” item.
6. From the “Output descriptor” pulldown menu select the “OutputGeneric” item.
7. Create the model by clicking on the “Accept” button.
8. Check that the model identifier “MatlabModel (1.0)” (model identifier plus the version in parenthesis) appears under the “Models” folder at the left frame of the HMI.

5.26. Test Procedure TP-0240: Session execution with models in Matlab

5.26.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0200: Session execution
- Test case TC-0120: Simulation creation
- Test case TC-0160: Session creation
- Test case TC-0350: Model languages support

5.26.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- Test TP-0230 has been successfully executed. Therefore “MatlabModel (1.0)” exists in the system. It can be found under Models folder in the left frame of the HMI.
- A Matlab version R2008a or later with the corresponding license is available and accessible through the system path.
- All input data and configuration files specific for this test are available.

5.26.3. Procedure Steps

Simulation Creation

1. Create a new simulation “MatlabSimulation” following the instructions in TP-0070:
2. Create the simulation. For this, access the “Repository” menu and select the “New simulation” option from the menu bar placed on top of the HMI.
3. Check that a window appears containing the information needed to create a new simulation.
4. Enter the following information in the “General properties” area:

Identifier	: MatlabSimulation
Author	: Tester
Description	: Simulation for Matlab model
Start Stage	: Select “MatlabStage” from the pulldown menu that is presented for the field.
Stop Stage	: Select “MatlabStage” from the pulldown menu that is presented for the field.

5. From the “Simulation stages” area check that:
 - The simulation stage selected is “MatlabStage”
 - The “Models available” table presents the list of models associated to “Matlab”. Check that “MatlabModel (1.0)” is part of the list.
 - The “Current Simulation Definition” list presents no models.

6. Select “MatlabModel (1.0)” from the “Models available” table and click on the “Next stage” button.
7. Check that the model selected on the previous stage is collected in the final list presented in the “Current simulation definition” area. Click on the “Accept” button to create the simulation.
9. Check that the simulation identifier “MatlabSimulation” appears under the “Simulations” folder at the left frame of the HMI. Double-click on it so that the window containing the information of the simulation is displayed.
10. Check that the simulation contains all the information set in previous steps.

Session Creation

11. Create a new session “MatlabSession” following the instructions in TP-0090.
12. Create the session. For this, access the “Repository” menu and select the “New session” option from the menu bar placed on top of the HMI.
13. Check that the system presents a window asking the user to create a new session or to import the settings from an existing folder. Click on the “Create new” button.
14. Check that a window appears containing the information needed to create a new session.
15. Enter the following information in the “General properties” area:

Identifier : MatlabSession
Description : Session for testing purposes.
Author : Tester

16. From the “Simulations set” area add as the simulations to the session by using the “Add” button and selecting the corresponding simulation (“MatlabSimulation”)
17. Load the input data files for each simulation by clicking on the “Input” tab from the “Session setup” area. (In this example is not necessary, because the Matlab model does not need input files)
18. Access the “Configuration” tab from the “Session setup” area. For each configuration file existing in \$TEST_HOME/data/conf, double click on the “File instance” of the corresponding model. Check that a file dialogue is displayed requesting the input file. Select the corresponding file from \$TEST_HOME/data/conf directory:

Global/Model	Identifier
Global	GlobalConfiguration.xml
MatlabModel	exampleFile.xml

19. Create the session by clicking on the “Accept” button.
20. Check that the session identifier “MatlabSession” appears under the “Session” folder at the left frame of the HMI. Double-click on it so that the window containing the information of the simulation is displayed.
21. Check that the session contains all the information set in the previous steps.

Session Execution

22. Access the “Repository” tab from the left frame of the window. Select the “Sessions→MatlabSession” option, press the mouse’s right button and select the “Run” option.

23. Check that openSF displays a message indicating that files are missing, and requests the user to either continue or cancel the execution.
24. Click on the option to continue the execution.
25. Check that a new window is displayed containing the following information:
 - The window title includes the running identifier composed by the session identifier plus the data and time the simulation was launched;
 - The session progress;
 - The model progress;
 - The list of log messages that are produced;
 - A button to abort the execution.
26. Wait until the simulation session ends, and keep track of the time spent for the execution.
27. Access the “Executions” tab from the left frame of the main window and check that a new session exists with the same simulation session identifier from step 2 (plus the date and time the execution began).
28. Double click on this new session identifier (or alternatively select it, press mouse’s right button and click on the View option) and check that a new window is displayed.
29. Check that the window has the following contents:
 - In the “General properties” area:
 - The “Identifier”, “Description” and “Author” fields contain the same information as in the simulation window.
 - The “Date/Time” field contains the same date and time that is part of the session identifier, and the “Duration” field contains approximately the same time period as the duration obtained in step 3.
 - The status of the simulation result is “Successful”.
 - In the “Session setup” area:
 - All files contained in the” Output” tab have the “Available” status.
 - The log messages appearing in the “Log” tab are the same as those produced during the execution.

5.27. Test Procedure TP-0250: Repository creation

5.27.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0330: Multi-repository capabilities
- Test case TC-0340: Multi-repository management

5.27.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.

5.27.3. Procedure Steps

1. Access the “System→Databases” menu from the openSF top menu bar.
2. Check that a new window is displayed listing the available databases within openSF application.
3. Check that at least there is one database listed, identified by “openSF” (default one).
4. Click on the “New” button.
5. Check that a new form is displayed so that the user can enter the new database information. Most fields have default data, so the user only has to fill in the ‘Database Name’ field. Fill in the fields with the following data, and click on the “OK” button:

Note that values marked as default shall be already filled.

Database Name	: MissionRep01
User	: <i>openSF</i> (default)
Password	: <i>openSF</i> (default)
Address	: <i>localhost</i> (default)
Script	: <i>test/data/scripts/openSFdb.sql</i> (default)

6. Check that the new database is now listed in the table with all databases, identified by “MissionRep01”
7. In order to ease the testing of the next procedures, create a new database repeating the steps 1 to 5 changing only the “Database Name” to “MissionRep02”.
8. Check that the new database is now listed in the table with all databases, identified by “MissionRep02”
9. Check that openSF is now connected to the new database. For this, in the label ‘connected to’ must appear the name of the new database.

5.28. Test Procedure TP-0260: Repository deletion

5.28.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0330: Multi-repository capabilities
- Test case TC-0340: Multi-repository management

5.28.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- Test TP-0250 has been successfully executed. Therefore, “MissionRep02” exists in the system (openSF file system and database).

5.28.3. Procedure Steps

1. Access the “System” menu from the top menu bar of the openSF main window and click on “Databases” option.
2. Check that “MissionRep02” appears on the list that is displayed.
3. Delete the database “MissionRep02” by selecting the database from the list and clicking on the “Delete” button.
4. Check that a dialog appears on screen asking if it is desired to delete the associated database. Click on “Yes, delete”.
5. Check that the database “MissionRep02” is no longer displayed in the databases list.
6. Check that the label ‘connected to’ is empty.

5.29. Test Procedure TP-0270: Repository switching

5.29.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0330: Multi-repository capabilities
- Test case TC-0340: Multi-repository management

5.29.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- Test TP-0250 has been successfully executed. Therefore, "MissionRep01" exists in the system (openSF file system and database).

5.29.3. Procedure Steps

1. Access the "System" menu from the top menu bar of the openSF main window and click on "Databases" option.
2. Check that "MissionRep01" appears on the list that is displayed. Select it and click on the "Connect" button.
3. Check that in the label 'connected to' appears the name of the database selected. In this case, "MissionRep01".
4. Click on the "OK" button.
5. Verify that all the simulation entities are now empty. ("Stages", "Descriptors", "Models", "Simulations", "Sessions" and "Tools")
6. Repeat again steps 1 to 4 but this time switching to openSF default database identified by "openSF"

5.30. Test Procedure TP-0280: Parameter Editor - Parameter creation

5.30.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0360: Parameter management system
- Test case TC-0380: Parameter validation
- Test case TC-0390: Parameter editor integration

5.30.2. Pre-conditions

- The environment variable \$TEST_HOME must be correctly set and available.
- The environment variable \$OPENSF_HOME must be correctly set and available.
- All configuration files specific for this test are available.
- openSF is executed and the HMI is displayed on the screen.

5.30.3. Procedure Steps

1. Access the Parameter Editor window from the openSF GUI by clicking on *Menu-System-ParameterEditor* button or by using Ctrl+P shortcut.
2. Check that the *Parameter Editor* main window appears on screen. Note that there is a “Parameter Files” frame with a “Load File” button.. Click on this button and check that a file selector frame appears on screen. Go to \$TEST_HOME/data/conf/ and click on “globalConfiguration.xml”. Repeat this step for, “geoConf.xml” and “ossConf.xml” files that can be found in the same folder.
3. Check that three new tabs appear on screen showing the parameters contained on the recently opened files.
4. Check that file list is available in the right side of the window.
5. Click on the “geoConf” tab and check that parameters contained in this file are listed in a tree-like view. Check that there are two parameters listed.
 - Nbands: Integer
 - Iterations: Integer
6. Click on the root node of the “geoConf.xml” parameters tree.
7. Click on “New parameter” button and check that a new window appears with fields for entering parameter attributes.
8. Enter the following parameter attributes:

```
Name      : testParam
Type      : Integer
Description : Test parameter
Dims     : 2 x 2
```

Value	: 3 3 3 3
Min	: 2
Max	: 4
Units	: meters

9. Click on “Accept” to create the new parameter.
10. Check that a parameter identified by “testParam” appears just under the root node of the “geoConf.xml” parameters tree.
11. In order to ease next test procedures create two new parameters repeating steps 6 to 9 entering the following information:

Name	: zenithAngle
Type	: FLOAT
Description	: Solar Zenith Angle
Dims	: 1 x 1
Value	: 30
Min	: 0
Max	: 90
Units	: degrees

Name	: azimuthAngle
Type	: FLOAT
Description	: Solar Zenith Angle
Dims	: 1 x 1
Value	: 10Min : 30
Max	: 180
Units	: degrees

14. Save changes pressing the “Save All” button found in right side panel under “Parameters File” title to save the changes into the file system.¹⁷

¹⁷ An extra checking can be performed, opening *geoConf.xml* with your desired text editor and checking that the new parameters are now in the file. It can be found under *\$TEST_HOME/data/conf/* folder.

5.31. Test Procedure TP-0290: Parameter Editor - Parameter edition, deletion

5.31.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0360: Parameter management system
- Test case TC-0380: Parameter validation

5.31.2. Pre-conditions

- Parameter Editor* package is included within openSF.
- Parameter Editor* is running and it appears on screen.
- Test TP-0280 has been successfully executed. Therefore, “testParam” exists in the “geoConf.xml” file.
- The environment variable \$TEST_HOME must be correctly set and available.
- All configuration files specific for this test are available.

5.31.3. Procedure Steps

1. Check that the *Parameter Editor* main window appears on screen. Note that there is a “Configuration Files” frame with a tab for every configuration file involved in the session. In this case check that three tabs are available “global_conf.xml”, “geoConf.xml” and “ossConf.xml”.
2. Double Click on the “testParam” node from the “geoConf.xml” parameters tree.
3. Check that a window is displayed showing the parameter attributes in text fields.
4. Change the following values in the parameter edition window:

```
Name       : newTestParam
Description : New description for test parameter
Value      : 2 2 2 2
```

5. Click on “Accept” button to edit parameter attributes.
6. Check that the “geoConf.xml” parameters tree contains a parameter identified by “newTestParam”
7. Repeat steps 2 and 3 to check that parameter attributes are the desired ones.
8. Click now on “newTestParam” node from “geoConf.xml” parameters tree.
9. Click on “Delete” button to remove “newTestParam” from “geoConf.xml” parameters tree.
10. Check that “newTestParam” no longer exist within the system.
11. Press “Save All” button found in right side panel under “Parameters File” title to save the changes into the file system.¹⁸

¹⁸ An extra checking can be performed, opening *geoConf.xml* with your desired text editor and checking that the deleted parameter no longer exist within the file. It can be found under *\$TEST_HOME/conf/* folder.

5.32. Test Procedure TP-0300: Parameter Editor - Parameter consistency checking

5.32.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0360: Parameter management system
- Test case TC-0380: Parameter validation

5.32.2. Pre-conditions

- Parameter Editor* package is included within openSF.
- Parameter Editor* is running and it appears on screen.
- Test TP-0280 has been successfully executed. Therefore, “zenithAngle” and “azimuthAngle” exists in the “geoConf.xml” file.
- The environment variable \$TEST_HOME must be correctly set and available.
- All configuration files specific for this test are available.

5.32.3. Procedure Steps

1. Check that the *Parameter Editor* main window appears on screen. Note that there is a “Configuration Files” frame with a tab for every configuration file involved in the session. In this case check that three tabs are available “global_conf.xml”, “geoConf.xml” and “ossConf.xml”.
2. Check that “zenithAngle” and “azimuthAngle” can be found under the “geoConf.xml” parameters tree.
3. Check that the icon under column “Validity” for parameter “azimuthAngle” is a warning signal and the text is “OutOfRange”
4. Check that the “Log Terminal” frame contains at least the following message:
 - Warning *azimuthAngle* is out of range value=10 min=30

5.33. Test Procedure TP-0310: Parameter Editor - Rule creation

5.33.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0360: Parameter management system
- Test case TC-0370: Rules definition
- Test case TC-0380: Parameter validation

5.33.2. Pre-conditions

- Parameter Editor* package is included within openSF.
- Parameter Editor* is running and it appears on screen.
- Test TP-0280 has been successfully executed. Therefore, “zenithAngle” and “azimuthAngle” exists in the “geoConf.xml” file.
- The environment variable \$TEST_HOME must be correctly set and available.
- All configuration files specific for this test are available.

5.33.3. Procedure Steps

1. Note that in the right side panel of the *ParameterEditor* a “Load File” button is present under “Rules File” title.
2. Click on “Load File” button and check that a file selector frame appears on screen. Select \$TEST_HOME/data/conf/ossConf.xml.
3. Check that the file is listed under “Rules File” frame from the right side panel.
4. Double click on “\$TEST_HOME/data/conf/ossConf.xml” in the “Rules File” frame. Check that a new frame appears on screen showing the content of “ossConf.xml”.
5. Delete all content of the file and write this on the text editor:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<rulesFile>
  <rulesDefinition>
    <rule id="ruleID">
      <condition type="EQUALS">
        <operandA name="geoConf.azimuthAngle" type="attribute" value="value"/>
        <operandB name="geoConf.zenithAngle" type="attribute" value="value"/>
      </condition>
    </rule>
  </rulesDefinition>
</rulesFile>
```

6. Save the file as “rulesFile.xml” under \$TEST_HOME/data/conf folder and close the text editor. Ensure that you click on “Save As” button of the text editor instead of “Save” button to avoid file overwriting.

5.34. Test Procedure TP-0320: Parameter Editor - Parameter validation

5.34.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0360: Parameter management system
- Test case TC-0370: Rules definition
- Test case TC-0380: Parameter validation

5.34.2. Pre-conditions

- Parameter Editor* package is included within openSF.
- Parameter Editor* is running and it appears on screen.
- Test TP-0280 has been successfully executed. Therefore, “zenithAngle” and “azimuthAngle” exists in the “geoConf.xml” file.
- Test TP-0310 has been successfully executed. Therefore, “rulesFile.xml” exists and can be located under *\$TEST_HOME/data/conf* folder.
- The environment variable *\$TEST_HOME* must be correctly set and available.
- All configuration files specific for this test are available.
- A external text editor is available

5.34.3. Procedure Steps

1. Check that the *Parameter Editor* main window appears on screen. Note that there is a “Configuration Files” frame with a tab for every configuration file involved in the session. In this case check that three tabs are available “global_conf.xml”, “geoConf.xml” and “ossConf.xml”.
2. Click on the “Load file” button in the ”Rules File” frame from the right side panel. Go to *\$TEST_HOME/data/conf* folder and select “rulesFile.xml”. Accept the selection for loading the rules file in the system.
3. Click on “Validate” button and wait until some message appears on the “Log Terminal” within *Parameter Editor* main window.
4. Check that the “Log Terminal” frame contains at least the following messages:
 - Validation successfully completed (1 warning 1 error)
 - Error file not compliant with *ruleID*, *azimuthAngle* and *zenithAngle* value must be equal.
 - Warning *azimuthAngle* is out of range value=10 min=30

5.35. Test Procedure TP-0330: Parameter Perturbation

5.35.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0400: Parameter perturbation

5.35.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- “E2E_test_session” exists in the system.¹⁹
- All input data and configuration files specific for this test are available.

5.35.3. Procedure Steps

1. Access the “Repository” tab from the left frame of the window. Select the “Sessions→E2E_test_session” option, press the mouse’s right button and select the “Edit” option.
2. Check that the session’ data appears in the main window.
3. Select the *Parameters* tab in *Session Setup* panel (bottom part of the window). Select a configuration parameter (FLOAT/INTEGER) and press the “Parameter Perturbation” button.
4. Check that a new window appears presenting the *Parameter Perturbation* form. Check that it is constituted by the following 3 areas: Perturbation Addition toolbar, Perturbation Tree and Execution Summary.
5. On the Perturbation Tree area the parameter selected shall appear with the message “*No perturbation*”.
6. On the Perturbation Tree select the “*No perturbation*” node and click on “+” button located in the Addition Toolbar (bottom of the screen). A new window shall pop up with the Perturbation Selection options.
7. On the Perturbation Selection window select *PerturbationType Random* and *normal* distribution. Check that a list of properties shall automatically appears with the following attributes:
 - seed
 - mu
 - sigma
8. Enter the following values in the perturbation properties list:

seed	: 1
mu	: 0.0
sigma	: 0.1

9. Click on the “Add” button to accept the perturbation and add it to the Perturbation Tree.

¹⁹ The identifier of the session to run shall be referred throughout the procedure as session01 although it can be applicable to any session appearing in the session list.

10. Check that the added property appears on the Perturbation Tree area.

11. Under the Perturbation Tree set the following data

Number of shots : 5

12. Click on “Preview” button. A new window shall appear for selecting perturbed execution mode. Select first option “*1. Statistical Execution Mode (N model executions)*” and press “Ok”

13. Check that the perturbation interface now shows the execution outline, reporting the number of shots for the model perturbation.

14. Accept the perturbation data set for the selected parameter by pressing the “Accept” button.

15. Run the session by pressing the “Run” button in the bottom of the session edition window.

16. On the “Parallel shots” dialog that appears select option “Yes, parallelize”.

17. Check that for the executed model with the perturbed parameter 5 intermediate output files have been generated corresponding to the number of shots defined in step 11.

5.36. Test Procedure TP-0340: MATLAB errors

5.36.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0410: MATLAB errors

5.36.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- All input data and configuration files specific for this test are available.

5.36.3. Procedure Steps

1. Access the “Repository” tab from the left frame of the window. Select the session that is input to test case TC-0410, “*MatlabFail_test_session*” from the openSF validation database, press the mouse’s right button and select the “Run” option.
2. Wait until the session execution ends.
3. Check that the log collects messages generated by the MATLAB model included in the session.

5.37. Test Procedure TP-0350: Shortcuts

5.37.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0420: Shortcuts

5.37.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.

5.37.3. Procedure Steps

1. Upon the openSF GUI, click on the CTRL-T keys. Check that the Tools window is activated.
2. Click on the CTRL-O keys. Check that the openSF configuration window is activated.
3. Click on the CTRL-P keys. Check that the Parameter Editor window is activated.
4. Click on the CTRL-H keys. Check that the Help window is activated.
5. Click on the CTRL-A keys. Check that the About window is activated.
6. Click on the CTRL-X keys. Check that openSF exits.
7. Click on the CTRL-D keys. Check that the Descriptors window is activated.
8. Click on the CTRL-G keys. Check that the Stages window is activated.
9. Click on the CTRL-M keys. Check that the Models window is activated.
10. Click on the CTRL-I keys. Check that the Simulations window is activated.
11. Click on the CTRL-S keys. Check that the Sessions window is activated.
12. Click on the CTRL-R keys. Check that the Executions Results window is activated.
13. Click on the CTRL-L keys. Check that the Logs window is activated.

5.38. Test Procedure TP-0360: Backup/Restore

5.38.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0430: Backup/Restore capability

5.38.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.

5.38.3. Procedure Steps

1. Upon the openSF GUI, click on the Repository tab and select the option to perform a backup of the system by clicking on Menu-System-Databases.
2. The database management interface shall appear. Click on *Backup* button (top-right corner)
3. Check that the system presents a dialogue box requesting the operator to confirm or change the name of the SQL script that shall be generated as a result of the backup.
4. Press the OK button to proceed with the operation with the default name.
5. Perform some database change operations on openSF, such as deleting sessions, IO descriptors, etc.
6. Click on the Repository tab and select the option to perform a restore by creating a New Database. See TP-0250, Repository Creation (5.27).
7. Select the SQL file saved in step 3.
8. Check that the status of the system (i.e. data repository) is the same as before step 4.

5.39. Test Procedure TP-0370: Import/Export

5.39.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0440: Import/export capability

5.39.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- At least one session exists in the openSF data repository.

5.39.3. Procedure Steps

1. Upon the openSF GUI, click on the Repository tab and select the option to perform an export of a session by right-clicking on a session and selecting “Export” option.
2. Check that the system informs about the successful export.
3. Access the “Repository” tab from the left frame of the window and click on “Sessions” option.
4. Delete the session selected in step 2 for the export by selecting it from the list and clicking on the “Delete” button. Confirm the operation when requested by the HMI.
5. Check that the session is no longer displayed under the “Sessions” folder from the left frame.
6. Click on the Repository tab and select the option to perform a session import by clicking on ”Menu-Repository-Import Session” ... button.
7. Select the SQL script and ZIP file that were saved in the export operation in step 1 and proceed with the operation.
8. Check that the session has been correctly imported into the system.

5.40. Test Procedure TP-0380: GUI closure

5.40.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0450: GUI closure

5.40.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- A session is running.

5.40.3. Procedure Steps

1. While the session is running, click on the option to quit the openSF.
2. Check that a dialogue box is presented warning the user about the running session.
3. Press the option to proceed to quit the GUI.
4. Execute openSF again.
5. Check that there is no trace of the running session.

5.41. Test Procedure TP-0390: Parallelization

5.41.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0460: Parallelization
- Test case TC-0470: Removal of logs from DB

5.41.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.

5.41.3. Procedure Steps

1. Access the “*System->Show Configuration*” menu from the openSF top menu bar.
2. Check that a new window is displayed containing tab “*System Configuration*” and select it.
3. Set the “Maximum Execution Threads” parameter to 1.
4. Run the input session with the Parallelisation maximum number of execution threads set to 1.
5. Wait until the session ends. Take note of the execution time.
6. Run the same session but setting the Parallelisation maximum number of execution threads to more than 1 (preferably to the same number of processor cores of the machine setting the parameter to 0).
7. Access the “*System->CPU Usage*” menu from the openSF top menu bar to open the CPU Usage window and check that the processes are running in parallel.
8. Wait until the session ends. Check that the running time of the second run is lower than the first.
9. Check that the outputs of both runs are identical.
10. Check that for both executions the log messages have been stored in a single file located in the top level sessions’ output directory.

5.42. Test Procedure TP-0400: Flexible session management

5.42.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0480: Flexible session management

5.42.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.

5.42.3. Procedure Steps

1. Access the “Repository” tab from the left frame of the window. Select a session from the “Sessions” option, press the mouse’s right button and select the “Edit” option.
2. On the window that is displayed containing the session’s information, check that there are options in a model’s contextual menu to:
 - change the version of a given model;
 - run a session from a certain point (model);
 - by-pass the execution of a given model;
3. On the window that is displayed containing the session’s information, check that there is a checkbox in the session definition to remove the intermediate data generated during the session’s run.
4. Select a model and activate the model’s contextual menu (right-click). Check that the contextual menu is showing the versions of the model allowing the user to select one.
5. Select a model version different from the one originally set in the session.
6. Select another model and select to bypass it. Check that openSF updates the files status indicating the user to provide the inputs needed for the next (or remaining) models of the session to be run.
7. Select the output files of the bypassed model.
8. Execute the session by pressing the “Run” button on the main window, Wait until the session ends.
9. Check that the session has been executed with the model version indicated in step 4.
10. Check that the session has been executed without running the bypassed model indicated in step 6.
11. Re-run the same session with the following features:
 - Select to re-run the session from a previous point: select any model, activate its contextual menu (right-click) and select the option to run the session from that model.
 - Select the option to remove intermediate data during the execution of the session.
12. Press the “Run” button. Wait until the session ends.
13. Check that the session has been executed starting from the model selected in step 10.
14. Check that the session’s output directory does not contain intermediate data.

5.43. Test Procedure TP-0410: Model export/import

5.43.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0490: Model export/import

5.43.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.

5.43.3. Procedure Steps

1. Access the “Executions” tab from the left frame of the window. Select a session from the “Sessions” option, press the mouse’s right button and select the “View” option.
2. On the window that is displayed containing the session’s information, check that there are options to import and export the data relative to a model by activating the model’s contextual menu (right-click).
3. Select any model and press the option to perform an export.
4. Check that openSF generates in the Home directory as output a compressed file including the configuration and input data files used by the model during the session execution.
5. Access the “Executions” tab from the left frame of the window. Select another execution form the same session selected in step 1.
6. Select the same model as in step 3 and press the option to perform an import.
7. Check that openSF presents a dialogue box requesting the user to load the compressed file to import. Select the file generated in step 4. Click on the “Open” button.
8. Check that openSF replaces the configuration and input files with the ones contained in the compressed file.

5.44. Test Procedure TP-0420: Copy elements

5.44.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0500: Copy elements

5.44.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.

5.44.3. Procedure Steps

1. Access the “Repository” tab from the left frame of the window and click on “IO descriptors” and select any “IO descriptor” option. Click the right button and check that there is an option to copy it. Select it.
2. Check that openSF present a dialogue requesting a user to provide a unique name. Enter the same name as the copied IO descriptor.
3. Check that openSF raises an error message indicating that the IO descriptor already exists. Enter a new name.
4. Check that the new IO descriptor is created and is part of the repository.
5. Repeat steps 1 to 4 for stages, models, simulations and sessions.

5.45. Test Procedure TP-0430: Element definition from XML

5.45.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0510: Element definition from XML

5.45.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.

5.45.3. Procedure Steps

1. Access the “Databases” dialog option from the “System” menu.
2. Check that there is an option to import the definition of elements, Click on this option.
3. Check that openSF present a dialogue box requesting the user to select the XML file including the definitions. Select the invalid file and click on the “OK” button:
4. Check that openSF raises an error message indicating that the input file is incorrect.
5. Repeat steps 2 and 3 but selecting the XML file with the openSF element definitions.
6. Check that openSF raises an error message indicating that imported elements do not have unique names.
7. Correct the XML file ensuring that the element definition names are distinct.
8. Repeat steps 2 and 3 selecting the corrected XML file.
9. Check that the elements included in the XML file have been created and are part of the openSF repository.

5.46. Test Procedure TP-0440: Error generation capabilities

5.46.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0520: Error generation capabilities

5.46.2. Pre-conditions

N/A

5.46.3. Procedure Steps

1. Open a shell tool of the Operating System.
2. Ensure that OPENSF_HOME is configured correctly (typically pointing to /home/<user>/openSF).
3. Configure LD_LIBRARY_PATH to include the OSFEG libraries by issuing command 'export LD_LIBRARY_PATH=\$OPENSF_HOME/OSFEG/lib:\$LD_LIBRARY_PATH'.
4. Change to directory \$OPENSF_HOME/OSFEG/test.
5. Run the test program including the C++ OSFEG library functions by executing script OSFEGTest.sh.
6. Check that the test program generates as output file report.html.
7. Display the contents of report.html in any browser installed in the system (e.g. issuing command 'firefox report.html');
8. Check that the outputs obtained for each function available in the OSFEG libraries match the ones included in the reference files (available in \$OPENSF_HOME/OSFEG/test/reference) obtained by Matlab for the same function definitions.

5.47. Test Procedure TP-0450: Migration procedure to openSF V3

5.47.1. Objective

This procedure covers the execution of the following test cases:

- ❑ Test case TC-0530: Migration from openSF V2 to openSF V3

5.47.2. Pre-conditions

- ❑ Two distinct openSF instances are available one with openSF v2.X and another one with openSF v3.

5.47.3. Procedure Steps

1. Access the data exportation directory in openSF V3 installation (typically in \$OPENSF_HOME/data/xml/export).
2. Execute the exportation script supplying the appropriate parameters: (1) the database name, (2) the database user, (3) the database user password, (4) the intended XML output file name, and (5) the intended output log file name.
3. Check that an XML file and a log file have been created in the current directory.
4. Copy the output log file to the openSF v3 installation top level session's output directory.
5. Launch openSF v3 and access the HMI.
6. Access the "Databases" dialog option from the "System" menu.
7. Click the option to import the definition of elements.
8. Check that openSF presents a dialogue box requesting the user to select the XML file including the definitions. Select the output XML file obtained in step 3 and click on the "OK" button.
9. Check that the elements included in the XML file have been created and are part of the openSF v3 repository.
10. Check that the exported log messages are part of the openSF v3 repository.

5.48. Test Procedure TP-0460: Performance test - E2E Session execution

5.48.1. Objective

This procedure covers the execution of the following test cases:

- ❑ Test case TC-0540: Performance improvement of openSF V3

5.48.2. Pre-conditions

- ❑ Two distinct openSF instances are available one with openSF v2.X and another one with openSF v3 both installed with the validation data set.

5.48.3. Procedure Steps

1. Manually determine the total duration time of steps 2 to 6 below:
2. Run the openSF V2.X tool.
3. Check that under “Repository” tab from the left frame there is a session under “Sessions” tag named “E2E_test_session”
4. Access the “Repository” tab from the left frame of the window. Select the “Sessions→E2E_test_session” option, press the mouse’s right button and select the “Run” option.
5. Check that a new window is displayed containing the following information:
 - The window title includes the running identifier composed by the session identifier plus the data and time the simulation was launched;
 - The session progress;
 - The model progress;
 - The list of log messages that are produced;
6. Wait until the simulation session ends, and keep track of the time spent for the execution.
7. Run the openSF V3 tool.
8. Configure the parallelization option to be disabled.
9. Exit the openSF V3 tool.
10. Manually determine the total duration time of steps 11 to 15 below:
11. Run the openSF V3 tool.
12. Check that under “Repository” tab from the left frame there is a session under “Sessions” tag named “E2E_test_session”
13. Access the “Repository” tab from the left frame of the window. Select the “Sessions→E2E_test_session” option, press the mouse’s right button and select the “Run” option.
14. Check that a new window is displayed containing the following information:

- The window title includes the running identifier composed by the session identifier plus the data and time the simulation was launched;
 - The session progress;
 - The model progress;
 - The list of log messages that are produced;
15. Wait until the simulation session ends, and keep track of the time spent for the execution.
 16. Configure the parallelization option to be enabled.
 17. Repeat steps 9 to 15 above
 18. Compare the overhead duration time obtained in steps 1 and 10.
 19. Compare the session execution time obtained in steps 6 and 15.

5.49. Test Procedure TP-0470: Python model creation

5.49.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0060: Model creation
- Test case TC-0350: Model languages support

5.49.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- A Python executable file named `PythonModel.py` exists in the `$TEST_HOME\bin` folder.
- The model identified as “PythonModel” does not exist in the system (openSF file system and database).
- Stages allocated to the model exist in the system, by executing TP-0020 for them.
- IO descriptors needed for the model exist in the system: “InputPython” and “OutputPython” by executing TP-0030 for each of them.

For InputPython, the information needed is

- Identifier : `InputPython`
- Description : Generic input for Python models
- Files: Four files named `input1`, `input2`, `input3`, `input4`

For OutputPython, the information needed is:

- Identifier : `OutputPython`
- Description : Generic output for Python models
- Files: Four files named `output1`, `output2`, `output3`, `output4`

- The environment variable `$TEST_HOME` is correctly set and available.
- The data files for the model to create should exist at `$TEST_HOME/data/` directory.

5.49.3. Procedure Steps

1. Create the model. For this, access the “Repository” menu and select the “New model” option from the menu bar placed on top of the HMI.
2. Check that a multi-tabbed window appears containing the information needed to create a new model.
3. Enter the following information in the “General” tab:

Model Id	: <code>PythonModel</code>
Model version:	<code>1.0</code>
Description	: <code>Python model</code>
Author	: <code>Tester</code>

Stage : Select the stage corresponding to the model from the pulldown menu that is presented for the field.

Binary file : Select “\$TEST_HOME\bin\testModel.py” from the file dialogue presented for the field.

4. Fill in the Default file field for the XML Configuration file by selecting “\$TEST_HOME\data\conf\exampleFile.xml”.
5. Click on the “Input/Output” tab and from the “Input Descriptor” pulldown menu and select the “InputGeneric” item.
6. From the “Output descriptor” pulldown menu select the “OutputGeneric” item.
7. Create the model by clicking on the “Accept” button.
8. Check that the model identifier “PythonModel (1.0)” (model identifier plus the version in parenthesis) appears under the “Models” folder at the left frame of the HMI.

5.50. Test Procedure TP-0480: Session execution with models in Python

5.50.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0200: Session execution
- Test case TC-0160: Session creation
- Test case TC-0350: Model languages support

5.50.2. Pre-conditions

- openSF is executed and the HMI is displayed on the screen.
- Test TP-0470 has been successfully executed. Therefore “PythonModel (1.0)” exists in the system. It can be found under Models folder in the left frame of the HMI.
- All input data and configuration files specific for this test are available.

5.50.3. Procedure Steps

Session Creation

1. Create a new session named “PythonSession”. For this, access the “Repository” menu and select the “New session” option from the menu bar placed on top of the HMI.
2. Check that the system presents a window asking the user to create a new session or to import the settings from an existing folder. Click on the “Create new” button.
3. Check that a window appears containing the information needed to create a new session.
4. Enter the following information in the “General properties” area:

```
Identifier : PythonSession
Description : Session for testing a model written in Python.
Author : Tester
```

5. From the “Models set” area add the model to the session by using the “Add” button and selecting the corresponding model (“PythonModel”)
6. Load the input data files for the model by clicking on the “Input” tab from the “Model setup” area. (In this example is not necessary, because the Python model does not need input files)
7. Access the “Configuration” tab from the “Session setup” area. Double click on the “File instance” of the corresponding model. Check that a file dialogue is displayed requesting the input file. Select the corresponding file from \$TEST_HOME/data/conf directory:

Global/Model	Identifier
Global	GlobalConfiguration.xml
PythonModel	exampleFile.xml

8. Create the session by clicking on the “Accept” button.

9. Check that the session identifier “PythonSession” appears under the “Session” folder at the left frame of the HMI. Double-click on it so that the window containing the information of the simulation is displayed.

10. Check that the session contains all the information set in the previous steps.

Session Execution

11. Access the “Repository” tab from the left frame of the window. Select the “Sessions→PythonSession” option, press the mouse’s right button and select the “Run” option.

12. Check that openSF displays a message indicating that files are missing, and requests the user to either continue or cancel the execution.

13. Click on the option to continue the execution.

14. Check that a new window is displayed containing the following information:

- The window title includes the running identifier composed by the session identifier plus the data and time the simulation was launched;
- The session progress;
- The model progress;
- The list of log messages that are produced;
- A button to abort the execution.

15. Wait until the simulation session ends, and keep track of the time spent for the execution.

16. Access the “Executions” tab from the left frame of the main window and check that a new session exists with the same simulation session identifier from step 2 (plus the date and time the execution began).

17. Double click on this new session identifier (or alternatively select it, press mouse’s right button and click on the View option) and check that a new window is displayed.

18. Check that the window has the following contents:

- In the “General properties” area:
 - The “Identifier”, “Description” and “Author” fields contain the same information as in the simulation window.
 - The “Date/Time” field contains the same date and time that is part of the session identifier, and the “Duration” field contains approximately the same time period as the duration obtained in step 15.
 - The status of the simulation result is “Successful”.
- In the “Session setup” area:
 - All files contained in the “Output” tab have the “Available” status.
 - The log messages appearing in the “Log” tab are the same as those produced during the execution.

5.51. Test Procedure TP-1000: Regression Test - E2E Session execution

5.51.1. Objective

This procedure covers the execution of the following test cases:

- Test case TC-0010: Installation
- Test case TC-0200: Session execution
- Test case TC-0530: Migration from openSF V2 to openSF V3
- Test case TC-0540: Performance improvement of openSF V3
- Test case TC-1000: Regression Test

5.51.2. Pre-conditions

- The Operational platform (operational machine plus the target operating system) is available.
- The Software User Manual [SUM] contains the procedure to install the software.

5.51.3. Procedure Steps

1. Install the openSF software in accordance with the installation procedures identified in the [SUM].
2. Ensure that during installation process “Validation Database” is selected
3. Perform a visual inspection of the openSF target directory, and check that the openSF directory structure has been correctly installed at the expected directory.
4. Perform a visual inspection of openSF COTS target directory and check that the COTS have been installed at the expected directory.
5. Run the openSF tool by launching a command shell window and typing in the following command at the \$OPENSF_HOME:

```
./openSF
```

6. Check that the openSF tool includes the functionalities of the latest openSF version from the top menu and at the left frame of the tool.
7. Check that under “Repository” tab from the left frame there is a session under “Sessions” tag named “E2E_test_session”
8. Access the “Repository” tab from the left frame of the window. Select the “Sessions→E2E_test_session” option, press the mouse’s right button and select the “Run” option.
9. Check that a new window is displayed containing the following information:
 - The window title includes the running identifier composed by the session identifier plus the data and time the simulation was launched;
 - The session progress;
 - The model progress;
 - The list of log messages that are produced;

- A button to abort the execution.
10. Wait until the simulation session ends, and keep track of the time spent for the execution.
 11. Access the “Executions” tab from the left frame of the main window and check that a new session exists with the same simulation session identifier from step 2 (plus the date and time the execution began).
 12. Double click on this new session identifier (or alternatively select it, press mouse’s right button and click on the View option) and check that a new window is displayed.
 13. Check that the window has the following contents:
 - In the “General properties” area:
 - The “Identifier”, “Description” and “Author” fields contain the same information as in the simulation window.
 - The “Date/Time” field contains the same date and time that is part of the session identifier, and the “Duration” field contains approximately the same time period as the duration obtained in step 3.
 - The status of the simulation result is “Successful”.
 - In the “Session setup” area:
 - All files contained in the” Output” tab have the “Available” status.
 - The log messages appearing in the “Log” tab are the same as those produced during the execution.

6. TRACEABILITY MATRICES

This section presents the forward and backward traceability between the following elements:

- Test cases and the system requirements from [SRD]
- Test procedures and test cases
- Test procedures to software version

6.1. Traceability between Validation Cases and System Requirements

6.1.1. Traceability Matrix from Test Cases to SRD Requirements

Table 6-1: Traceability Matrix, Test Cases vs. SRD Requirements

Test Case	SRD Requirements
TC-0010: Installation	SR-FUN-0050
	SR-INS-0010
TC-0020: Configuration set-up	
	SR-FUN-0340
	SR-FUN-0350
	SR-FUN-0360
	SR-FUN-0380
TC-0030: Configuration edition	SR-OPE-0010
	SR-FUN-0390
	SR-FUN-0400
	SR-FUN-0410
	SR-FUN-0420
TC.0035: Tool assignment	SR-OPE-0010
	SR-FUN-0030
TC-0040: I/O descriptors creation	SR-FUN-1240
	SR-FUN-0140
	SR-FUN-0150
	SR-FUN-0180
TC-0050: I/O descriptors deletion	SR-OPE-0010
	SR-FUN-0160
	SR-FUN-0170
	SR-SEC-0010
TC-0060: Model creation	SR-OPE-0010
	SR-FUN-0010
	SR-FUN-0060
	SR-FUN-0070

Test Case	SRD Requirements
	SR-FUN-0080
	SR-FUN-0090
	SR-FUN-0130
	SR-FUN-0140
	SR-FUN-0190
	SR-INT-0020
	SR-OPE-0010
TC-0070: Incorrect model creation	SR-FUN-0200
	SR-FUN-0210
	SR-OPE-0010
TC-0080: Model edition	SR-FUN-0220
	SR-FUN-0230
	SR-FUN-0240
	SR-FUN-0250
	SR-SEC-0010
	SR-INT-0020
	SR-OPE-0010
TC-0090: Model version creation	SR-FUN-0260
	SR-FUN-0270
	SR-FUN-0280
	SR-FUN-0290
	SR-OPE-0010
TC-0100: Incorrect model version creation	SR-FUN-0300
	SR-OPE-0010
TC-0110: Model deletion	SR-FUN-0310
	SR-FUN-0320
	SR-FUN-0330
	SR-SEC-0010
	SR-OPE-0010
TC-0120: Simulation creation	SR-FUN-0430
	SR-FUN-0440
	SR-FUN-0450
	SR-FUN-0460
	SR-FUN-0470
	SR-FUN-0480
	SR-FUN-0490
	SR-INT-0010
	SR-OPE-0010
TC-0130: Incorrect simulation creation	SR-FUN-0440
	SR-FUN-0500
	SR-FUN-0510
	SR-OPE-0010
TC-0140: Simulation edition	SR-FUN-0520
	SR-FUN-0530

Test Case	SRD Requirements
	SR-FUN-0540
	SR-FUN-0550
	SR-SEC-0010
	SR-INT-0010
	SR-OPE-0010
TC-0150: Simulation deletion	SR-FUN-0560
	SR-FUN-0570
	SR-FUN-0580
	SR-SEC-0010
	SR-OPE-0010
TC-0160: Session creation	
	SR-FUN-0590
	SR-FUN-0600
	SR-FUN-0610
	SR-FUN-0620
	SR-FUN-0630
	SR-FUN-0640
	SR-FUN-0760
	SR-FUN-0780
	SR-FUN-0790
	SR-FUN-0800
	SR-FUN-0810
	SR-OPE-0010
TC-0170: Incorrect session creation	SR-FUN-0650
	SR-FUN-0660
	SR-OPE-0010
TC-0180: Session edition	SR-FUN-0670
	SR-FUN-0680
	SR-FUN-0690
	SR-FUN-0700
	SR-FUN-0710
	SR-FUN-0720
	SR-FUN-0780
	SR-FUN-0790
	SR-FUN-0800
	SR-FUN-0810
	SR-SEC-0010
	SR-OPE-0010
TC-0190: Session deletion	SR-FUN-0730
	SR-FUN-0740
	SR-FUN-050
	SR-SEC-0100
	SR-OPE-0010
TC-0200: Session execution	SR-FUN-0010

Test Case	SRD Requirements
	SR-FUN-0840
	SR-FUN-0850
	SR-FUN-0860
	SR-FUN-0870
	SR-FUN-0880
	SR-FUN-0900
	SR-FUN-1250
	SR-FUN-1260
	SR-FUN-1290
	SR-FUN-1300
	SR-FUN-1330
	SR-INT-0030
	SR-OPE-0010
	SR-OPE-0020
TC-0210: Failed session execution	SR-FUN-0840
	SR-FUN-0850
	SR-FUN-0860
	SR-FUN-0910
	SR-INT-0030
	SR-OPE-0010
	SR-OPE-0020
TC-0220: Session abortion	SR-FUN-0890
	SR-INT-0030
	SR-OPE-0010
	SR-OPE-0020
TC-0230: Iterative session execution	SR-FUN-0820
	SR-FUN-0830
	SR-FUN-0840
	SR-FUN-0850
	SR-FUN-0860
	SR-FUN-0870
	SR-FUN-0880
	SR-FUN-0900
	SR-INT-0030
	SR-OPE-0010
	SR-OPE-0020
	SR-OPE-0050
TC-0240: Session script execution	SR-OPE-0020
	SR-OPE-0030
	SR-OPE-0040
TC-0250: Session execution with breakpoints	SR-FUN-0010
	SR-FUN-0020
	SR-FUN-0770
	SR-OPE-0010

Test Case	SRD Requirements
TC-0260: Session results visualisation	SR-FUN-0010
	SR-FUN-0940
	SR-FUN-0950
	SR-FUN-0960
	SR-FUN-0970
	SR-FUN-0980
	SR-FUN-1010
	SR-FUN-1020
	SR-FUN-1280
	SR-FUN-1300
	SR-FUN-1310
	SR-FUN-1320
	SR-FUN-1330
	SR-OPE-0010
TC-0270: Session plots visualisation	SR-FUN-0010
	SR-FUN-0990
	SR-FUN-1000
	SR-OPE-0010
TC-0280: Multi-language model session execution	SR-RES-0010
TC-0290: Multi-platform session execution	SR-IMP-0040
TC-0300: Web access for new users	SR-FUN-1030
	SR-FUN-1070
	SR-FUN-1080
	SR-FUN-1090
TC-0310: Web access for delivery download	SR-FUN-1030
	SR-FUN-1060
TC-0320: Web access for SPR management	SR-FUN-1030
	SR-FUN-1040
	SR-FUN-1050
TC-0330: Multi-repository capabilities	SR-FUN-1100
	SR-FUN-1110
TC-0340: Multi-repository management	SR-FUN-1100
	SR-FUN-1110
TC-0350: Model languages support	SR-RES-0050
TC-0360: Parameter management system	SR-FUN-0790
	SR-FUN-1120
	SR-FUN-1130
	SR-FUN-1140
	SR-FUN-1150
	SR-FUN-1160
	SR-FUN-1170
TC-0370: Rules definition	SR-FUN-1120
	SR-FUN-1140
	SR-FUN-1160
TC-0380: Parameter validation	SR-FUN-0790
	SR-FUN-1120
	SR-FUN-1140

Test Case	SRD Requirements
	SR-FUN-1160
TC-0390: Parameter editor integration	SR-FUN-1180
	SR-FUN-1190
	SR-FUN-1200
TC-0400: Parameter perturbation	SR-FUN-1200
	SR-FUN-1210
	SR-FUN-1220
	SR-FUN-1230
TC-0410: MATLAB errors	SR-FUN-1270
	SR-FUN-1280
	SR-FUN-1340
TC-0420: Shortcuts	SR-FUN-1350
TC-0430: Backup/Restore capability	SR-FUN-1360
	SR-FUN-1370
TC-0440: Import/export capability	SR-FUN-1380
	SR-FUN-1390
	SR-FUN-1400
	SR-FUN-1410
	SR-FUN-1420
	SR-FUN-1430
TC-0450: GUI closure	SR-FUN-1440
	SR-OPE-0060
TC-0460: Parallelization	SR-OPE-0070
	SR-FUN-1450
	SR-FUN-1460
TC-0470: Removal of logs from DB	SR-FUN-1470
	SR-FUN-1345
TC-0480: Flexible session management	SR-FUN-1480
	SR-FUN-1490
	SR-FUN-1500
	SR-FUN-1510
	SR-FUN-1520
	SR-FUN-1530
TC-0490: Model export/import	SR-FUN-1540
	SR-FUN-1550
	SR-FUN-1560
	SR-FUN-1570
	SR-FUN-1580
	SR-FUN-1590
	SR-FUN-1600
	SR-FUN-1610
TC-0500: Copy elements	SR-FUN-1620
	SR-FUN-1630
TC-0510: Element definition from XML	SR-FUN-1640

Test Case	SRD Requirements
	SR-FUN-1650
	SR-FUN-1660
	SR-FUN-1670
TC-0520: Error generation capabilities	SR-FUN-1680
	SR-FUN-1690
	SR-FUN-1700
	SR-FUN-1710
	SR-FUN-1720
	SR-FUN-1730
TC-0530: Migration from openSF V2 to openSF V3	SR-FUN-1345
	SR-FUN-1640
	SR-FUN-1650
	SR-FUN-1660
	SR-FUN-1670
TC-0540: Performance improvement of openSF V3	SR-FUN-1345
	SR-FUN-1450
	SR-FUN-1460
	SR-FUN-1470
TC-1000: Regression Test	SR-FUN-0010
	SR-FUN-0840
	SR-FUN-0850
	SR-FUN-0860
	SR-FUN-0870
	SR-FUN-0880
	SR-FUN-0900
	SR-INT-0030
	SR-OPE-0010
	SR-OPE-0020
	SR-FUN-0050
	SR-INS-0010

6.1.2. Traceability Matrix from SRD Requirements to Test cases

Table 6-2: Traceability Matrix, SRD Requirements vs. Test Cases

SRD Requirements	Test Case
SR-FUN-0010	TC-0060: Model creation
	TC-0200: Session execution
	TC-0250: Session execution with breakpoints
	TC-0260: Session results visualisation
	TC-0270: Session plots visualisation
	TC-1000: Regression Test
SR-FUN-0020	TC-0250: Session execution with breakpoints
SR-FUN-0030	TC-0035: Tool assignment
SR-FUN-0050	TC-0010: Installation
	TC-1000: Regression Test

SRD Requirements	Test Case
SR-FUN-0060	TC-0060: Model creation
SR-FUN-0070	TC-0060: Model creation
SR-FUN-0080	TC-0060: Model creation
SR-FUN-0090	TC-0060: Model creation
SR-FUN-0130	TC-0060: Model creation
SR-FUN-0140	TC-0040: I/O descriptors creation TC-0060: Model creation
SR-FUN-0150	TC-0040: I/O descriptors creation
SR-FUN-0160	TC-0050: I/O descriptors deletion
SR-FUN-0170	TC-0050: I/O descriptors deletion
SR-FUN-0180	TC-0040: I/O descriptors creation
SR-FUN-0190	TC-0060: Model creation
SR-FUN-0200	TC-0070: Incorrect model creation
SR-FUN-0210	TC-0070: Incorrect model creation
SR-FUN-0220	TC-0080: Model edition
SR-FUN-0230	TC-0080: Model edition
SR-FUN-0240	TC-0080: Model edition
SR-FUN-0250	TC-0080: Model edition
SR-FUN-0260	TC-0090: Model version creation
SR-FUN-0270	TC-0090: Model version creation
SR-FUN-0280	TC-0090: Model version creation
SR-FUN-0290	TC-0090: Model version creation
SR-FUN-0300	TC-0100: Incorrect model version creation
SR-FUN-0310	TC-0110: Model deletion
SR-FUN-0320	TC-0110: Model deletion
SR-FUN-0330	TC-0110: Model deletion
SR-FUN-0340	TC-0020: Configuration set-up
SR-FUN-0350	TC-0020: Configuration set-up
SR-FUN-0360	TC-0020: Configuration set-up
SR-FUN-0380	TC-0020: Configuration set-up
SR-FUN-0390	TC-0030: Configuration edition
SR-FUN-0400	TC-0030: Configuration edition
SR-FUN-0410	TC-0030: Configuration edition
SR-FUN-0420	TC-0030: Configuration edition
SR-FUN-0430	TC-0120: Simulation creation
SR-FUN-0440	TC-0120: Simulation creation TC-0130: Incorrect simulation creation
SR-FUN-0450	TC-0120: Simulation creation
SR-FUN-0460	TC-0120: Simulation creation
SR-FUN-0470	TC-0120: Simulation creation
SR-FUN-0480	TC-0120: Simulation creation
SR-FUN-0490	TC-0120: Simulation creation
SR-FUN-0500	TC-0130: Incorrect simulation creation
SR-FUN-0510	TC-0130: Incorrect simulation creation

SRD Requirements	Test Case
SR-FUN-0520	TC-0140: Simulation edition
SR-FUN-0530	TC-0140: Simulation edition
SR-FUN-0540	TC-0140: Simulation edition
	TC-0140: Simulation edition
SR-FUN-0550	TC-0140: Simulation edition
SR-FUN-0560	TC-0150: Simulation deletion
SR-FUN-0570	TC-0150: Simulation deletion
SR-FUN-0580	TC-0150: Simulation deletion
SR-FUN-0590	TC-0160: Session creation
SR-FUN-0600	TC-0160: Session creation
SR-FUN-0610	TC-0160: Session creation
SR-FUN-0620	TC-0160: Session creation
SR-FUN-0630	TC-0160: Session creation
SR-FUN-0640	TC-0160: Session creation
SR-FUN-0650	TC-0170: Incorrect session creation
SR-FUN-0660	TC-0170: Incorrect session creation
SR-FUN-0670	TC-0180: Session edition
SR-FUN-0680	TC-0180: Session edition
SR-FUN-0690	TC-0180: Session edition
SR-FUN-0700	TC-0180: Session edition
SR-FUN-0710	TC-0180: Session edition
SR-FUN-0720	TC-0180: Session edition
SR-FUN-0730	TC-0190: Session deletion
SR-FUN-0740	TC-0190: Session deletion
SR-FUN-0750	TC-0190: Session deletion
SR-FUN-0760	TC-0160: Session creation
SR-FUN-0770	TC-0250: Session execution with breakpoints
SR-FUN-0780	TC-0160: Session creation
	TC-0180: Session edition
SR-FUN-0790	TC-0160: Session creation
	TC-0180: Session edition
	TC-0360: Parameter management system
	TC-0380: Parameter validation
SR-FUN-0800	TC-0160: Session creation
	TC-0180: Session edition
SR-FUN-0810	TC-0160: Session creation
	TC-0180: Session edition
SR-FUN-0820	TC-0230: Iterative session execution
SR-FUN-0830	TC-0230: Iterative session execution
SR-FUN-0840	TC-0200: Session execution
	TC-0210: Failed session execution
	TC-0230: Iterative session execution
	TC-1000: Regression Test
SR-FUN-0850	TC-0200: Session execution

SRD Requirements	Test Case
	TC-0210: Failed session execution
	TC-0230: Iterative session execution
	TC-1000: Regression Test
SR-FUN-0860	TC-0200: Session execution
	TC-0210: Failed session execution
	TC-0230: Iterative session execution
	TC-1000: Regression Test
SR-FUN-0870	TC-0200: Session execution
	TC-0230: Iterative session execution
	TC-1000: Regression Test
SR-FUN-0880	TC-0200: Session execution
	TC-0230: Iterative session execution
	TC-1000: Regression Test
SR-FUN-0890	TC-0220: Session abortion
SR-FUN-0900	TC-0200: Session execution
	TC-0230: Iterative session execution
	TC-1000: Regression Test
SR-FUN-0910	TC-0210: Failed session execution
SR-FUN-0940	TC-0260: Session results visualisation
SR-FUN-0950	TC-0260: Session results visualisation
SR-FUN-0960	TC-0260: Session results visualisation
SR-FUN-0970	TC-0260: Session results visualisation
SR-FUN-0980	TC-0260: Session results visualisation
SR-FUN-0990	TC-0270: Session plots visualisation
SR-FUN-1000	TC-0270: Session plots visualisation
SR-FUN-1010	TC-0260: Session results visualisation
SR-FUN-1020	TC-0260: Session results visualisation
SR-FUN-1030	TC-0300: Web access for new users TC-0310: Web access for delivery download TC-0320: Web access for SPR management
SR-FUN-1040	TC-0320: Web access for SPR management
SR-FUN-1050	TC-0320: Web access for SPR management
SR-FUN-1060	TC-0310: Web access for delivery download
SR-FUN-1070	TC-0300: Web access for new users
SR-FUN-1080	TC-0300: Web access for new users
SR-FUN-1090	TC-0300: Web access for new users
SR-FUN-1100	TC-0330: Multi-repository capabilities TC-0340: Multi-repository management
SR-FUN-1110	TC-0330: Multi-repository capabilities TC-0340: Multi-repository management
SR-FUN-1120	TC-0360: Parameter management system TC-0370: Rules definition TC-0380: Parameter validation
SR-FUN-1130	TC-0360: Parameter management system
SR-FUN-1140	TC-0360: Parameter management system

SRD Requirements	Test Case
	TC-0370: Rules definition TC-0380: Parameter validation
SR-FUN-1150	TC-0360: Parameter management system
SR-FUN-1160	TC-0360: Parameter management system TC-0370: Rules definition TC-0380: Parameter validation
SR-FUN-1170	TC-0360: Parameter management system
SR-FUN-1180	TC-0390: Parameter editor integration
SR-FUN-1190	TC-0390: Parameter editor integration
SR-FUN-1200	TC-0390: Parameter editor integration TC-0400: Parameter perturbation
SR-FUN-1210	TC-0400: Parameter perturbation
SR-FUN-1220	TC-0400: Parameter perturbation
SR-FUN-1230	TC-0400: Parameter perturbation
SR-FUN-1240	TC-0035: Tool assignment
SR-FUN-1250	TC-0200: Session execution
SR-FUN-1260	TC-0200: Session execution
SR-FUN-1270	TC-0410: MATLAB errors
SR-FUN-1280	TC-0260: Session results visualisation TC-0410: MATLAB errors
SR-FUN-1290	TC-0200: Session execution
SR-FUN-1300	TC-0200: Session execution TC-0260: Session results visualisation
SR-FUN-1310	TC-0260: Session results visualisation
SR-FUN-1320	TC-0260: Session results visualisation
SR-FUN-1330	TC-0200: Session execution TC-0260: Session results visualisation
SR-FUN-1340	TC-0410: MATLAB errors
SR-FUN-1345	TC-0470: Removal of logs from DB TC-0530: Migration from openSF V2 to openSF V3 TC-0540: Performance improvement of openSF V3
SR-FUN-1350	TC-0420: Shortcuts
SR-FUN-1360	TC-0430: Backup/Restore capability
SR-FUN-1370	TC-0430: Backup/Restore capability
SR-FUN-1380	TC-0440: Import/export capability
SR-FUN-1390	TC-0440: Import/export capability
SR-FUN-1400	TC-0440: Import/export capability
SR-FUN-1410	TC-0440: Import/export capability
SR-FUN-1420	TC-0440: Import/export capability
SR-FUN-1430	TC-0440: Import/export capability
SR-FUN-1440	TC-0440: Import/export capability
SR-FUN-1450	TC-0460: Parallelization TC-0540: Performance improvement of openSF V3
SR-FUN-1460	TC-0460: Parallelization TC-0540: Performance improvement of openSF V3
SR-FUN-1470	TC-0460: Parallelization

SRD Requirements	Test Case
	TC-0540: Performance improvement of openSF V3
SR-FUN-1480	TC-0480: Flexible session management
SR-FUN-1490	TC-0480: Flexible session management
SR-FUN-1500	TC-0480: Flexible session management
SR-FUN-1510	TC-0480: Flexible session management
SR-FUN-1520	TC-0480: Flexible session management
SR-FUN-1530	TC-0480: Flexible session management
SR-FUN-1540	TC-0490: Model export/import
SR-FUN-1550	TC-0490: Model export/import
SR-FUN-1560	TC-0490: Model export/import
SR-FUN-1570	TC-0490: Model export/import
SR-FUN-1580	TC-0490: Model export/import
SR-FUN-1590	TC-0490: Model export/import
SR-FUN-1600	TC-0490: Model export/import
SR-FUN-1610	TC-0490: Model export/import
SR-FUN-1620	TC-0500: Copy elements
SR-FUN-1630	TC-0500: Copy elements
SR-FUN-1640	TC-0510: Element definition from XML TC-0530: Migration from openSF V2 to openSF V3
SR-FUN-1650	TC-0510: Element definition from XML TC-0530: Migration from openSF V2 to openSF V3
SR-FUN-1660	TC-0510: Element definition from XML TC-0530: Migration from openSF V2 to openSF V3
SR-FUN-1670	TC-0510: Element definition from XML TC-0530: Migration from openSF V2 to openSF V3
SR-FUN-1680	TC-0520: Error generation capabilities
SR-FUN-1690	TC-0520: Error generation capabilities
SR-FUN-1700	TC-0520: Error generation capabilities
SR-FUN-1710	TC-0520: Error generation capabilities
SR-FUN-1720	TC-0520: Error generation capabilities
SR-FUN-1730	TC-0520: Error generation capabilities
SR-IMP-0040	TC-0290: Multi-platform session execution
SR-INS-0010	TC-0010: Installation TC-1000: Regression Test
SR-INT-0010	TC-0120: Simulation creation TC-0140: Simulation edition
SR-INT-0020	TC-0060: Model creation TC-0080: Model edition
SR-INT-0030	TC-0200: Session execution TC-0210: Failed session execution TC-0220: Session abortion TC-0230: Iterative session execution TC-1000: Regression Test
SR-OPE-0010	TC-0020: Configuration set-up TC-0030: Configuration edition

SRD Requirements	Test Case
	TC-0040: I/O descriptors creation
	TC-0050: I/O descriptors deletion
	TC-0060: Model creation
	TC-0070: Incorrect model creation
	TC-0080: Model edition
	TC-0090: Model version creation
	TC-0100: Incorrect model version creation
	TC-0110: Model deletion
	TC-0120: Simulation creation
	TC-0130: Incorrect simulation creation
	TC-0140: Simulation edition
	TC-0150: Simulation deletion
	TC-0160: Session creation
	TC-0170: Incorrect session creation
	TC-0180: Session edition
	TC-0190: Session deletion
	TC-0200: Session execution
	TC-0210: Failed session execution
	TC-0220: Session abortion
	TC-0230: Iterative session execution
	TC-0250: Session execution with breakpoints
	TC-0260: Session results visualisation
	TC-0270: Session plots visualisation
TC-1000: Regression Test	
SR-OPE-0020	TC-0200: Session execution
	TC-0210: Failed session execution
	TC-0220: Session abortion
	TC-0230: Iterative session execution
	TC-0240: Session script execution
	TC-1000: Regression Test
SR-OPE-0030	TC-0240: Session script execution
SR-OPE-0040	TC-0240: Session script execution
SR-OPE-0050	TC-0230: Iterative session execution
SR-OPE-0060	TC-0450: GUI closure
SR-OPE-0070	TC-0450: GUI closure
SR-RES-0010	TC-0280: Multi-language model session execution
SR-SEC-0010	TC-0050: I/O descriptors deletion
	TC-0080: Model edition
	TC-0110: Model deletion
	TC-0140: Simulation edition
	TC-0150: Simulation deletion
	TC-0180: Session edition
	TC-0190: Session deletion
SR-RES-0050	TC-0350: Model languages support

6.2. Traceability between Test Procedures and Test Cases

6.2.1. Traceability Matrix from Test Procedures to Test Cases

Table 6-3: Traceability Matrix Test Procedures vs. Test Cases

Test Procedure	Test Case
TP-0010: Installation	TC-0010: Installation
TP-0020: Configuration set-up	TC-0020: Configuration set-up
	TC-0030: Configuration edition
TP-0030: I/O descriptor management	TC-0040: I/O descriptors creation
	TC-0050: I/O descriptors deletion
TP-0040: Model creation	TC-0060: Model creation
	TC-0070: Incorrect model creation
TP-0050: Model edition and deletion	TC-0080: Model edition
	TC-0110: Model deletion
TP-0060: Model version creation	TC-0090: Model version creation
	TC-0100: Incorrect model version creation
TP-0070: Simulation creation	TC-0120: Simulation creation
	TC-0130: Incorrect simulation creation
TP-0080: Simulation edition and deletion	TC-0140: Simulation edition
	TC-0150: Simulation deletion
TP-0085: Tool assignment	TC-0035: Tool assignment
TP-0090: Session creation	TC-0160: Session creation
	TC-0170: Incorrect session creation
TP-0100: Session edition and deletion	TC-0180: Session edition
	TC-0190: Session deletion
TP-0110: Session execution	TC-0200: Session execution
	TC-0260: Session results visualisation
TP-0120: Failed session execution	TC-0210: Failed session execution
TP-0130: Session plots visualisation	TC-0270: Session plots visualisation
TP-0140: Abort session execution	TC-0220: Session abortion
TP-0150: Iterative session execution	TC-0230: Iterative session execution
TP-0160: Session script execution	TC-0240: Session script execution
TP-0170: Session execution with breakpoints	TC-0250: Session execution with breakpoints
TP-0180: Web access for new users	TC-0300: Web access for new users
TP-0190: Web access for delivery download	TC-0310: Web access for delivery download
TP-0200: Web access for SPR management	TC-0320: Web access for SPR management
TP-0210: IDL model creation	TC-0060: Model creation
	TC-0350: Model languages support.

Test Procedure	Test Case
TP-0220: Session execution with models in IDL	TC-0200: Session execution TC-0120: Simulation creation TC-0160: Session creation TC-0350: Model languages support.
TP-0230: Session execution with models in Matlab and IDL	TC-0060: Model creation TC-0350: Model languages support.
TP-0240: Session execution with models in Matlab and IDL	TC-0200: Session execution TC-0120: Simulation creation TC-0160: Session creation TC-0350: Model languages support.
TP-0250: Repository creation	TC-0330: Multi-repository capabilities. TC-0340: Multi-repository management.
TP-0260: Repository deletion	TC-0330: Multi-repository capabilities. TC-0340: Multi-repository management.
TP-0270: Repository switching	TC-0330: Multi-repository capabilities. TC-0340: Multi-repository management.
TP-0280: Parameter Editor – Parameter creation	TC-0360: Parameter management system. TC-0380: Parameter validation.
TP-0290: Parameter Editor – Parameter edition and deletion	TC-0360: Parameter management system. TC-0380: Parameter validation.
TP-0300: Parameter Editor – Parameter consistency checking	TC-0360: Parameter management system. TC-0380: Parameter validation.
TP-0310: Parameter Editor – Rule creation	TC-0360: Parameter management system. TC-0370: Rules definition. TC-0380: Parameter validation.
TP-0320: Parameter Editor – Parameter validation	TC-0360: Parameter management system. TC-0370: Rules definition. TC-0380: Parameter validation.
TP-0330: Parameter Perturbation	TC-0400: Parameter perturbation
TP-0340: MATLAB errors	TC-0410: MATLAB errors
TP-0350: Shortcuts	TC-0420: Shortcuts
TP-0360: Backup/Restore	TC-0430: Backup/Restore capability
TP-0370: Import/Export	TC-0440: Import/Export capability
TP-0380: GUI closure	TC-0450: GUI closure
TP-0390: Parallelization	TC-0460: Parallelization TC-0470: Removal of logs from DB
TP-0400: Flexible session management	TC-0480: Flexible session management
TP-0410: Model export/import	TC-0490: Model export/import
TP-0420: Copy elements	TC-0500: Copy elements
TP-0430: Element definition from XML	TC-0510: Element definition from XML
TP-0440: Error generation capabilities	TC-0520: Error generation capabilities
TP-0450: Migration procedure to openSF V3	TC-0530: Migration from openSF V2 to openSF V3

Test Procedure	Test Case
TP-0460: Performance test – E2E Session execution	TC-0540: Performance improvement of openSF V3
TP-0470: Python model creation	TC-0060: Model creation TC-0350: Model languages support
TP-0480: Session execution with models in Python	TC-0160: Session creation TC-0200: Session execution TC-0350: Model languages support
TP-1000: Regression Test – E2E Session execution	TC-0010: Installation TC-0200: Session execution TC-1000: Regression Test

6.2.2. Traceability Matrix from Test Cases to Test Procedures

Table 6-4: Traceability Matrix, Test Cases vs. Test Procedures

Test Case	Test Procedure
TC-0010: Installation	TP-0010: Installation TP-1000: Regression Test – E2E Session execution
TC-0020: Configuration set-up	TP-0020: Configuration set-up
TC-0030: Configuration edition	TP-0020: Configuration set-up
TC-0035: Tool assignment	TP-0085: Tool assignment
TC-0040: I/O descriptors creation	TP-0030: I/O descriptor management
TC-0050: I/O descriptors deletion	TP-0030: I/O descriptor management
TC-0060: Model creation	TP-0040: Model creation TP-0210: IDL model creation TP-0240: Matlab model creation TP-0470: Python model creation
TC-0070: Incorrect model creation	TP-0040: Model creation
TC-0080: Model edition	TP-0050: Model edition and deletion
TC-0090: Model version creation	TP-0060: Model version creation
TC-0100: Incorrect model version creation	TP-0060: Model version creation
TC-0110: Model deletion	TP-0050: Model edition and deletion
TC-0120: Simulation creation	TP-0070: Simulation creation TP-0220: Session execution with models in IDL TP-0240: Session execution with models in Matlab
TC-0130: Incorrect simulation creation	TP-0070: Simulation creation
TC-0140: Simulation edition	TP-0080: Simulation edition and deletion
TC-0150: Simulation deletion	TP-0080: Simulation edition and deletion
TC-0160: Session creation	TP-0090: Session creation TP-0220: Session execution with models in IDL TP-0240: Session execution with models in Matlab

Test Case	Test Procedure
	TP-0480: Session execution with models in Python
TC-0170: Incorrect session creation	TP-0090: Session creation
TC-0180: Session edition	TP-0100: Session edition and deletion
TC-0190: Session deletion	TP-0100: Session edition and deletion
TC-0200: Session execution	TP-0110: Session execution TP-0220: Session execution with models in IDL TP-0240: Session execution with models in Matlab TP-0480: Session execution with models in Python TP-1000: Regression Test – E2E Session execution
TC-0210: Failed session execution	TP-0120: Failed session execution
TC-0220: Session abortion	TP-0140: Abort session execution
TC-0230: Iterative session execution	TP-0150: Iterative session execution
TC-0240: Session script execution	TP-0160: Session script execution
TC-0250: Session execution with breakpoints	TP-0170: Session execution with breakpoints
TC-0260: Session results visualisation	TP-0110: Session execution
TC-0270: Session plots visualisation	TP-0130: Session plots visualisation
TC-0300: Web access for new users	TP-0180: Web access for new users
TC-0310: Web access for delivery download	TP-0190: Web access for delivery download
TC-0320: Web access for SPR management	TP-0200: Web access for SPR management
TC-0330: Multi-repository capabilities	TP-0250: Repository creation TP-0260: Repository edition and deletion TP-0270: Repository Switching
TC-0340: Multi-repository management	TP-0250: Repository creation TP-0260: Repository edition and deletion TP-0270: Repository Switching
TC-0350: Model languages support	TP-0210: IDL model creation TP-0220: Session execution with models in IDL TP-0230: Matlab model creation TP-0240: Session execution with models in Matlab TP-0470: Python model creation TP-0480: Session execution with models in Python
TC-0360: Parameter management system	TP-0280: Parameter Editor – Parameter creation TP-0290: Parameter Editor – Parameter edition and deletion TP-0310: Parameter Editor – Rule creation TP-0320: Parameter Editor – Parameter validation
TC-0370: Rules definition	TP-0310: Parameter Editor – Rule creation TP-0320: Parameter Editor – Parameter validation

Test Case	Test Procedure
TC-0380: Parameter validation	TP-0280: Parameter Editor – Parameter creation TP-0290: Parameter Editor – Parameter edition and deletion TP-0310: Parameter Editor – Rule creation TP-0320: Parameter Editor – Parameter validation
TC-0400: Parameter perturbation	TP-0330: Parameter Perturbation
TC-0410: MATLAB errors	TP-0340: MATLAB errors
TC-0420: Shortcuts	TP-0350: Shortcuts
TC-0430: Backup/Restore capability	TP-0360: Backup/Restore
TC-0440: Import/Export capability	TP-0370: Import/Export
TC-0450: GUI closure	TP-0380: GUI closure
TC-0460: Parallelization	TP-0390: Parallelization
TC-0470: Removal of logs from DB	TP-0390: Parallelization
TC-0480: Flexible session management	TP-0400: Flexible session management
TC-0490: Model export/import	TP-0410: Model export/import
TC-0500: Copy elements	TP-0420: Copy elements
TC-0510: Element definition from XML	TP-0430: Element definition from XML
TC-0520: Error generation capabilities	TP-0440: Error generation capabilities
TC-0530: Migration from openSF V2 to openSF V3	TP-0450: Migration procedure to openSF V3
TC-0540: Performance improvement of openSF V3	TP-0460: Performance test – E2E Session execution
TC-1000: Regression Test	TP-1000: Regression Test – E2E Session execution

6.3. Traceability between Test Procedures and Software Versions

Traceability matrices below present the traceability and consequently the validity of test procedures regarding the openSF software version. Current openSF versions are:

- openSF version 1.0, including minor releases associated (v1.1 and v1.2)
- openSF version 2.0, including minor releases associated (v2.1)
- openSF version 2.2, including the extended capabilities requested in MR02;
- openSF version 3.0, including the extended capabilities requested in [CR1].
- openSF version 3.1, with minor releases associated.

6.3.1. Traceability Matrix from Test Procedures to Software Versions

Table 6-5: Traceability Matrix, Test Procedures vs. Software Versions

Test Procedure	Software Version
TP-0010: Installation	1.0, 3.0

Test Procedure	Software Version
TP-0020: Configuration set-up	1.0
TP-0030: I/O descriptor management	1.0
TP-0040: Model creation	1.0
TP-0050: Model edition and deletion	1.0
TP-0060: Model version creation	1.0
TP-0070: Simulation creation	1.0
TP-0080: Simulation edition and deletion	1.0
TP-0085: Tool assignment	1.0
TP-0090: Session creation	1.0
TP-0100: Session edition and deletion	1.0
TP-0110: Session execution	1.0 2.2
TP-0120: Failed session execution	1.0, 3.0, 3.1
TP-0130: Session plots visualisation	1.0
TP-0140: Abort session execution	1.0
TP-0150: Iterative session execution	1.0, 3.0
TP-0160: Session script execution	1.0
TP-0170: Session execution with breakpoints	1.0, 3.0
TP-0180: Web access for new users	1.0
TP-0190: Web access for delivery download	1.0
TP-0200: Web access for SPR management	1.0
TP-0210: IDL model creation	2.0
TP-0220: Session execution with models in IDL	2.0
TP-0230: Matlab model creation	2.0
TP-0240: Session execution with models in Matlab	2.0
TP-0250: Repository creation	2.0
TP-0260: Repository edition and deletion	2.0
TP-0270: Repository switching	2.0, 3.0
TP-0280: Parameter Editor – Parameter creation	2.0
TP-0290: Parameter Editor – Parameter edition and deletion	2.0
TP-0300: Parameter Editor – Parameter consistency checking	2.0, 3.0
TP-0310: Parameter Editor – Rule creation	2.0
TP-0320: Parameter Editor – Parameter validation	2.0
TP-0330: Parameter Perturbation	2.2
TP-0340: MATLAB errors	2.2
TP-0350: Shortcuts	2.2
TP-0360: Backup/Restore	2.2, 3.0
TP-0370: Import/Export	2.2
TP-0380: GUI closure	2.2

Test Procedure	Software Version
TP-0390: Parallelization	3.0, 3.1
TP-0400: Flexible session management	3.0
TP-0410: Model export/import	3.0, 3.1
TP-0420: Copy elements	3.0
TP-0430: Element definition from XML	3.0
TP-0440: Error generation capabilities	3.0
TP-0450: Migration procedure to openSF V3	3.0, 3.1
TP-0460: Performance test – E2E Session execution	3.0, 3.1
TP-1000: Regression Test – E2E Session execution	2.0, 3.0, 3.1

6.3.2. Traceability Matrix from Software Versions to Test Procedures

Table 6-6: Traceability Matrix, Software Versions vs. Test Procedures

Software Version	Test Procedure
1.0	TP-0010: Installation
	TP-0020: Configuration set-up
	TP-0030: I/O descriptor management
	TP-0040: Model creation
	TP-0050: Model edition and deletion
	TP-0060: Model version creation
	TP-0070: Simulation creation
	TP-0080: Simulation edition and deletion
	TP-0085: Tool assignment
	TP-0090: Session creation
	TP-0100: Session edition and deletion
	TP-0110: Session execution
	TP-0120: Failed session execution
	TP-0130: Session plots visualisation
	TP-0140: Abort session execution
	TP-0150: Iterative session execution
	TP-0160: Session script execution
	TP-0170: Session execution with breakpoints
	TP-0180: Web access for new users
	TP-0190: Web access for delivery download
TP-0200: Web access for SPR management	
2.0	TP-0210: Matlab and IDL model creation
	TP-0220: Session execution with models in Matlab and IDL
	TP-0230: Matlab model creation
	TP-0240: Session execution with models in Matlab
	TP-0250: Repository creation

Software Version	Test Procedure
	TP-0260: Repository edition and deletion
	TP-0270: Repository switching
	TP-0280: Parameter Editor – Parameter creation
	TP-0290: Parameter Editor – Parameter edition and deletion
	TP-0300: Parameter Editor – Parameter consistency checking
	TP-0310: Parameter Editor – Rule creation
	TP-0320: Parameter Editor – Parameter validation
	TP-1000: Regression Test – E2E Session execution
2.2	TP-0110: Session execution
	TP-0330: Parameter Perturbation
	TP-0340: MATLAB errors
	TP-0350: Shortcuts
	TP-0360: Backup/Restore
	TP-0370: Import/Export
	TP-0380: GUI closure
3.0	TP-0390: Parallelization
	TP-0400: Flexible session management
	TP-0410: Model export/import
	TP-0420: Copy elements
	TP-0430: Element definition from XML
	TP-0440: Error generation capabilities
	TP-0450: Migration procedure to openSF V3
	TP-0460: Performance test – E2E Session execution
	TP-1000: Regression Test – E2E Session execution
3.1	TP-0120: Failed session execution
	TP-0390: Parallelization
	TP-0410: Model export/import
	TP-0450: Migration procedure to openSF V3
	TP-0460: Performance test - E2E Session execution
	TP-1000: Regression Test - E2E Session execution

7. OPENSF TEST PLAN

This section presents the test plan for each openSF version (starting with Version 3).

7.1. openSF V3 test plan

Table 7-1: Test Cases and Test Procedures for openSF V3

Test Type	Test Case	Test Procedure
Regression Test	TC-0010: Installation	TP-0010: Installation
Regression Test	TC-0210: Failed session execution	TP-0120: Failed session execution
Regression Test	TC-0230: Iterative session execution	TP-0150: Iterative session execution
Regression Test	TC-0250: Session execution with breakpoints	TP-0170: Session execution with breakpoints
Regression Test	TC-0330: Multi-repository capabilities. TC-0340: Multi-repository management.	TP-0270: Repository switching
Regression Test	TC-0400: Parameter perturbation	TP-0330: Parameter perturbation
Regression Test	TC-0430: Backup/Restore capability	TP-0360: Backup/Restore
Regression Test	TC-1000: Regression Test	TP-1000: Regression Test - E2E Session execution
New Test	TC-0460: Parallelization TC-0470: Removal of logs from DB	TP-0390: Parallelization
New Test	TC-0480: Flexible session management	TP-0400: Flexible session management
New Test	TC-0490: Model export/import	TP-0410: Model export/import
New Test	TC-0500: Copy elements	TP-0420: Copy elements
New Test	TC-0510: Element definition from XML	TP-0430: Element definition from XML
New Test	TC-0520: Error generation capabilities	TP-0440: Error generation capabilities
New Test	TC-0530: Migration from openSF V2 to openSF V3	TP-0450: Migration procedure to openSF V3
New Test	TC-0540: Performance improvement of openSF V3	TP-0460: Performance test - E2E Session execution

7.2. openSF V3.1 test plan

Table 7-2: Test Cases and Test Procedures for openSF V3.1

Test Type	Test Case	Test Procedure
SPR related Test (SPR-AR-01, SPR-AR-02)	TC-0210: Failed session execution	TP-0120: Failed session execution
SPR related Test (SPR-AR-03)	TC-0460: Parallelization	TP-0390: Parallelization
SPR related Test (SPR-AR-04)	TC-0490: Model export/import	TP-0410: Model export/import
SPR related Test (SPR-AR-05)	TC-0530: Migration from openSF V2 to openSF V3	TP-0450: Migration procedure to openSF V3
AR related Test (inconclusive status)	TC-0540: Performance improvement of openSF V3	TP-0460: Performance test - E2E Session execution
Regression Test	TC-1000: Regression Test	TP-1000: Regression Test - E2E Session execution

End of document