Road-, Air- and Water-based Future Internet Experimentation

Project Acronym:	RAWFIE		
Contract Number:	645220		
Starting date:	Jan 1st 2015	Ending date:	Dec 31st, 2018

Deliverable Number and Title	D4.9 - Pilot Experimentation Scenarios for Validation and Testing		
Confidentiality	PU	Deliverable type[1]	R
Deliverable File	RAWFIE-D4.9-Pilot Experimentation Scenarios for Validation and Testing_v01.docx	Date	30.06.2017
Approval Status[2]	WP leader, 1st Reviewer, 2nd Reviewer	Version	1.0
Contact Person	Ph. Dallemagne	Organization	CSEM
Phone		E-Mail	Philippe.Dallemagne@csem.ch

[1] Deliverable type: P(Prototype), R (Report), O (Other)
[2] Approval Status: WP leader, 1st Reviewer, 2nd Reviewer, Advisory Board

AUTHORS TABLE

Name	Company	E-Mail
Ph. Dallemagne	CSEM	Philippe.Dallemagne@csem.ch
Nikolaos Pringouris	HAI	Pringgouris.nikolaos@haicorp.com
Kiriakos Georgouleas	HAI	Georgouleas.Kiriakos@haicorp.com
Giovanni Tusa	IES	g.tusa@iessolutions.eu
Kakia Panagidi	UoA	kakiap@di.uoa.gr
Kostantinos Kolomvatsos	UoA	kostasks@di.uoa.gr
Miquel Cantero	ROBOTNIK	mcantero@robotnik.es
Marcel Heckel	FRAUNHOFER	Marcel.Heckel@ivi.fraunhofer.de
Cveta Dimitrova	ABERON	cveta.dimitrova@aberon.bg
Vasil Kumanov	ABERON	vasil.kumanov@aberon.bg
Kiriakos Georgouleas	HAI	GEORGOULEAS.Kiriakos@haicorp.com
Savvas Chatzchristofis	CERTH	savvash@gmail.com
Ricardo Martins	MST Oceanscan	rasm@oceanscan-mst.com
Lionel Blondé	HESSO	lionel.blonde@hesge.ch

REVIEWERS TABLE

Name	Company	E-Mail
Kakia Panagidi	UoA	kakiap@di.uoa.gr
Thanasis Kapoutsis	CERTH	athakapo@iti.gr

DISTRIBUTION

Name / Role	Company	Level of confidentiality[3]	Type of deliverable
All		PU	R

[3] Deliverable Distribution: PU (Public, can be distributed to everyone), CO (Confidential, for use by consortium members only), RE (Restricted, available to a group specified by the Project Advisory Board).

CHANGE HISTORY

Version	Date	Reason for Change	Pages/Sections Affected
0.3	26.07.2017	Ready for internal review	N/A
0.7	10.08.2017	Ready for upload	N/A

Abstract:

This deliverable describes the definition of the testing and validation scenarios of the RAWFIE platform. It defines the test and validation plan as a set of scenarios that are performed in WP6 as well as the definition of the metrics and the success criteria.

Keywords:

Verification validation tests scenarios end users metrics success criteria

Table of Contents

List of Tables

Foreword

The first version of the deliverable "Pilot Experimentation Scenarios for Validation and Testing" (D4.3) introduced the plan and the approach that is followed to perform and document the tests for verification and validation of the RAWFIE system. The second iteration of this deliverable focuses on the needs of the stakeholders that will participate in the context of the Open Calls.

While retaining most of the previous work reflected in D4.3, D4.6 added the description of the scenarios and uses cases that corresponds to the needs of newcomers and their uses of the RAWFIE system. D4.9 updates D4.6 by taking into account the refinements brought to the validation scenarios and to the metrics and by considering the ongoing validation process. In particular:

- the verification tests were revised according to the requirements mentioned in D3.3 and updated components requirements mapping tables and updated features taken from D4.8.

as far as the validation metrics and success criteria are concerned, the original metrics tables from D4.6 are kept untouched. Nevertheless, metrics and success criteria have been extensively reworked over the different releases of the document.

-

- the validation scenarios are tightly linked with the corresponding results presented in D6.4. In principle, we kept the same validation scenario descriptions as in D4.6 highlighting those which are already completely or partially done.

This third iteration of the document completes the process of identification and description of the scenarios for validation and testing.

Taking into account the iterative process adopted in the project, and therefore the fact that each deliverable type, and so the one reporting on *"Pilot Experimentation Scenarios for Validation and Testing",* is submitted at regular intervals corresponding to the different cycles of requirements, design, verification and validation planning and implementation, in the next iterations of this deliverable the consortium will take the actions needed to follow the recommendations received after every review.

The D4.3 document included the complete list of verification tests that were identified as relevant during the first cycle, at a very early stage of the project, to ensure an extensive component and system test campaign. After the first and second implementation rounds, some tests may prove unnecessary and should be deleted from subsequent versions of the document. The open call lead to the selection of several proposals from various new RAWFIE stakeholders, which the consortium analysed and reported in D4.6. The analysis is done from several perspectives: the needs and requirements expressed by newcomers, the identified satisfaction levels and the corresponding metrics and the typical scenarios and use

cases. D4.6 justifies the presence of scenarios and tests from the user and needs perspectives (in particular by tracing back to the requirements). In D4.9, the verification and validation tests described in Section 5 and Section 6, will be kept only if they relate to any specific requirement appearing respectively in D3.2 and then D3.3.

Updated or new requirements coming from WP3, are in turn reflected in the functionalities described in the architecture and design deliverables (D4.4, D4.5, D4.7, D4.8). Tests related to functionalities that are not explicitly mentioned in those deliverables, will not be considered as well, or existing tests will be updated accordingly.

It should also be noted that, with the preparation of deliverable D6.1, the consortium took the opportunity to proceed with the update or the removal of all tests that were not applicable anymore, after the first implementation cycle was completed. As recommended and already stated in the first release of this deliverable (D4.3), the consortium defines in D4.6 the success criteria for the evaluation of the platform, and refines them in deliverables D4.9. Requirements linked to a given scenario are mentioned in its description. The orphan requirements are also identified and listed.

We understand that any scenario that is not linked to any requirement may lack of a justification. However, most of the requirements are coming from all kinds of stakeholders, in particular experimenters and testbed owners; many of these requirements are high-level and most of them are addressed in validation scenarios. Verification tests may look sometimes disconnected from the actual requirements, but they represent important steps for the technical verification of a component or combination of components.

Executive Summary

This deliverable is based on the results of T4.1 for what concerns the definition of the testing and validation scenarios of the RAWFIE platform. It describes the test and validation methodology and it defines a set of test scenarios that will be performed in WP6 as well as the definition of the success criteria.

In D3.1, the end users have specified the RAWFIE requirements at all possible levels (component, system, etc.) and many categories (functional, non-functional, etc.). These requirements shall be met by the RAWFIE testbeds, with respect to their achievements or specific success criteria. It defines the minimum set of requirements to be met by the testbed and specifies the scenarios that are sufficient to validate the testbed, with respect to requirement subsets.

The test and validation scenarios deal with the global features of the RAWFIE system. They cover the test and validation of the Open interface framework, the interoperability of different sets of entities (testbeds, UxV, etc.) and the management of the RAWFIE federation.

The test scenarios are used during the system integration and testing, in particular of the different font-ends, middle-tier, non-functional services (e.g. storage) and the operational entities (e.g. UxV, testbeds, and environment). The validation covers the entire RAWFIE Federation life-cycle, but it focuses on the deployment and operation phases.

Verification takes place during the development (e.g. in the way of unit tests) and on completion of development (integration tests), before the system is delivered to the pilot users. The purpose of verification is to ensure that each component works as expected and RAWFIE prototype components are related correctly through all expected scenarios. The verification process also offers an opportunity to test RAWFIE under extreme conditions such as realistic volumes of data, to give an indication of theoretical performance and ensure that the system is scalable to a sufficient degree when it is deployed for the users.

In order to verify components, the Consortium has identified all components of the system and verification scenarios for each of them has been prepared. Verification needs to be carried out on each component by way of unit tests to be sure that the required functionality is achieved in the way that is expected, and on the whole system to ensure that it achieves the required functionality, performance and reliability.

Evaluation takes place once RAWFIE prototype has been deployed for the pilot users to assess how the system performs under live scenarios. Evaluation covers areas such as the usability of the user interfaces, the type, quantity and quality of the data provided and overall use and usability of the system. The system will be evaluated following the metrics and success criteria defined in this document.

Main Section

1 Introduction

1.1 Scope of D4.9

This deliverable specifies the verification and validation scenarios to be exercised on a RAWFIE testbed and the success criteria used for the evaluation of its implementation. Validation scenarios aim at checking if the system works as expected from the End Users point of view (System Validation). They can be refined and enhanced at a later stage in cooperation with WP6, and have to be strictly linked to the Use Cases defined within WP3. This document also prepares the approach for Components and Integrated Prototype Testing (System Verification) for Task 6.1 (e.g. functional and performance tests, and so on). Finally, it describes the Verification vs. Validation activities and approaches.

D4.9 is an input for organising, driving and evaluating the work done in WP6, in particular:

- Task 6.1 Prototype Integration, Testing and customization
- Task 6.2 Evaluation and Platform Validation

The document covers:

- What needs to be tested (complete testbeds, subsystems, etc.);
- Who will test (users, stakeholders, RAWFIE partners, EAB, etc.);
- How tests are performed (tools, means, metrics, criteria, etc.).

1.2 Abbreviations

Table 1: Abbreviations

Abbreviation	Meaning
ACCS	Accounting Service
AT	Aerial Testbed
AUV	Autonomous Underwater Vehicle
BS	Booking Service
BT	Booking Tool
DoW	Description of Work
EAT	Experiment Authoring Tool
EC	Experiment Controller
ECV	EDL Compiler and Validator
EDL	Experiment Description Language
EMT	Experiment Monitoring Tool
EST	Early sub-system tests
LS	Launching Service
MT	Maritime Testbed
MM	Monitoring Manager
NC	Network Controller
PA	Platform Administrator
PT-DAA-E	Data Analysis Engine
PT-DAA-T	Data Analysis Tool
RC	Resource Controller
RET	Resource Explorer Tool
SYMS	System Monitoring Service
SMT	System Monitoring Tool
TD	Testbed Directory
TM	Testbed Manager
TO	Tesbed Operator
UAV	Unmanned Aerial Vehicle
UM	UxV Manufacturer
URS	Users & Rights Service
UD	User Defined
UGV	Unmanned Ground Vehicle
USV	Unmanned Surface Vehicle
UxP	UxV Proximity component
UxV	Unmanned aerial/ground/surface Vehicle
UxVNT	UxV Navigation Tool

VE	Visualisation Engine
VT	Vehicular Testbed
VT (scenario)	Visualisation Tool
WP	Web Portal
WT	Wiki Tool

2 Object of the validation and testing

The RAWFIE system is made of a set of sub-systems, components, processes, etc. and, thus, it should be thoroughly validated and tested. Only through an efficient verification and validation process, possible problems and malfunctions will be revealed and corrected in order to secure the efficient execution of the RAWFIE platform. A set of scenarios have been defined to verify the properties of the RAWFIE system during the development, to verify that the RAWFIE system and components comply with the specifications and to evaluate the degree of achievement with respect to the expected performance. The RAWFIE consortium aims to secure the efficient execution of the system in two axes: (a) the *verification* of the available components and the integrated system, (b) the *validation / evaluation* of the whole system.

The verification process aims at revealing potential problems. A set of template for describing components and system integration tests that must be passed (functional tests, performance tests, etc.) will be defined. Integration tests are required for the outcomes of third party projects. The infrastructure of the testbeds being part of RAWFIE will be subject to a number of mandatory tests in order to be integrated to RAWFIE platform. Such tests cover the data generated from the testbed monitoring services and security policy screening. The UxV additions perform a number of software tests to check the integration of the message bus components (consumers, producers) with the RAWFIE data exchange backbone. UxV additions will be checked through a number of testing experiments of increasing complexity authored and executed in the RAWFIE platform. Verification scenarios are adopted to verify that the platform and the single components (as implemented within WP5) properly meet the requirements from the technical perspective (system verification). The system validation and evaluation process aims to reveal if the system also meets the defined requirements and performs as expected from the end users' perspective. Similarly to the verification process, the validation will be built on top of a set of templates for describing the validation scenarios. The establishment of the scenario descriptions and specifications was initially based on the analysis of the user requirements defined in D3.1/D3.3 and the related metrics and expected performance (success criteria); the analysis of the proposals received by the consortium in the frame of the first RAWFIE Open Call revealed new use cases and scenarios, which were considered as additional "user defined" scenarios.

Nota bene: The template used for describing the scenarios already includes a "status" of the capability for RAWFIE to pass it, although the verification scenarios are defined for the experimentation phase. This field is currently a placeholder for the upcoming tests that will

be performed, since we will complete these templates across the entire project lifetime and probably beyond it. Whenever the verification (or validation) will be done, we will update the status.

2.1 Verification

Verification takes place during the development (e.g., in the way of unit tests) and on completion of development (integration tests) before the system is delivered to the pilot users. The purpose of verification is to ensure that each component works as expected and RAWFIE prototype components are interacting correctly through all expected scenarios. The verification process also offers an opportunity to test RAWFIE under extreme conditions such as realistic volumes of data to give an indication of the theoretical performance and ensure that the system is scalable to a sufficient degree when deployed for the users. The aim is to answer questions related to if the developed components meet the initial requirements and if they are built in the right way. In order to verify the available components, the consortium has identified all components of the system and verification scenarios for each of them has been prepared. Verification needs to be carried out on each component by way of unit tests to be sure that the required functionality is achieved in the way that is expected, and on the whole system to ensure that it achieves the required functionality, performance and reliability. Verification will help to lower the number of defects in early as well as in late stages of development and lead to better understanding of the components. Finally, it will reduce the chances of failures in the software implementation.

2.2 Validation and evaluation

Validation and evaluation takes place once the RAWFIE prototype has been deployed for the pilots to assess how the system performs under live scenarios. Evaluation covers areas such as the usability of the user interfaces, type, quantity and quality of the data provided and the overall use and the usability of the system. The system will be evaluated adopting the metrics defined in this document. The discussed process will execute extensive evaluations in order to assess the overall effectiveness and efficiency of the RAWFIE solution and to prove its added-value in a real environment. The validation campaign will include formal tests of the RAWFIE platform against the requirements set, as well as against the use cases' objectives. Validation sessions and templates, based on requirements will take place, expecting to bring valuable information about general user acceptance and usability of the provided infrastructure. Performance or other technical issues will be thoroughly evaluated. The activity will conclude with the preparation of a report summarizing the system evaluation and providing an assessment of its readiness for operational use.

2.3 RAWFIE federation lifecycle

The RAWFIE federation lifecycle is tested through specific scenarios that 'see' the framework as a black box. The aim is to identify if the system works appropriately through a high level evaluation. At first, the tests will identify if a set of different testbeds are smoothly attached to the RAWFIE architecture. The test scenarios will define the type, the number and the location of the testbeds. Accordingly, a specific EDL script will be defined that covers the

entire set of the available components and testbeds. For instance, the script will define requirements for the parallel execution of different types of testbeds in the same experiment. In combination with the stress tests, the specific approach is judged very efficient as it will identify possible problems in the RAWFIE architecture. In general, the federation lifecycle will be evaluated through a number of major phases that include: user and testbed registration, authoring, booking, launching and evaluation of an experiment. In the upcoming sections, a set of validation scenarios are provided that cover all the discussed phases accompanied by a set of metrics that will reveal the performance of the framework.

2.4 Verification and validation infrastructure and procedures

Verification will ensure that RAWFIE components meet the defined requirements while the validation phase will check if the system meets the high level requirements as defined by the consortium. Requirements are verified and the implemented components and the system are evaluated against the defined requirements. In addition, the validation process will ensure that all requirements are adequately tested or demonstrated, and that test results are as expected and can be repeated to verify correct implementation of the RAWFIE components. The consortium will follow a specific plan that follows these guidelines and it will help to ensure that the provided components can consistently meet a high level of quality and performance requirements. In short, the verification and the validation plans are as follows:

- Verification plan. For each component and sub-components the tests will manage to reveal their performance. Specific objectives will be defined for each (sub-) component and a detailed description of the verification scenario will be provided. Moreover, pre-requisites and the expected results will undertake the role of identifying if the component meets the defined requirements. Finally, specific testing scenarios could be devoted to identify the appropriate communication between components in order to secure the efficient data transfer throughout the RAWFIE architecture. The discussed plan will be realized during the implementation process in order to identify possible problems early in the development process.

- Validation plan. A set of validation scenarios will be adopted to reveal the performance of the platform. These scenarios mainly focus on testing from the stakeholder's point of view. Hence, in each scenario the main stakeholders will be defined and a detailed description will elaborate on the adopted steps. In addition, the involved (sub-) components will be referred in order to have a view on the part of the RAWFIE architecture that is evaluated. It should be noted that these scenarios will be evaluated against the already defined requirements.

2.4.1 Non regression and stress tests

The aim of non-regression and stress tests is to identify possible errors in the RAWFIE architecture. These errors could be caused by a number of issues like wrong interfaces design and / or implementation, insufficient data passed to / from each component and so on.

Non regression tests will be realized on the RAWFIE prototype. As it is very difficult to have a large set of UxVs during validation, specific routines undertake the role of producing data related to UxVs behaviour (e.g., location, measurements, status of resources). Hence, the consortium is performing large scale validation producing large amounts of data in high rates. The discussed routines are launched / combined with the prototype and represent the behaviour of RAWFIE nodes / testbeds. A post-processing tool undertakes the responsibility of analysing the derived behaviour of the system based on a set of metrics. For instance, the number of errors, the data transferred, the time required to complete an 'action' and so on are some useful metrics that could be adopted to measure the performance of the system. In addition, the consortium has adopted an approach that will take into consideration the 'footprint' of each test. This means that every validation scenario is combined with a specific 'view' of the system. For instance, specific tests are realized either from the experimenter point of view or from the testbed perspective. In other words, the 'footprint' combines each test with what is tested (i.e., RAWFIE architecture). Finally, specific reports are realized to describe the outcome of the process.

Stress tests are also performed during the validation phase: for example the test done in Skaramagas experienced some stress conditions, in which the devices were working in high wind conditions that were dangerous for operation or the wifi network was in low quality, leading to network disconnections. These are the kind of circumstances experienced during real life tests that can define the bottom line of the system capabilities.

Based on the aforementioned routines, the consortium provides extensive tests in order to reveal the performance of the platform. The aim is to bring the framework close to its limits. Fails and means for fast recovering will be realized leading to a high quality system. Stress tests are realized in the following axes: (a) high number of users (b) high number of bookings, (c) high number of concurrent connections to the system, (d) high number of testbeds / nodes, (e) high load, (f) unpredictable events like taking a testbed / node or the DBMS offline and restarting it, etc. These tests focus on unpredictable events randomly generated during the framework execution and put emphasis on robustness, availability, and error handling under a heavy load, rather than on what would be considered correct behaviour under normal circumstances.

2.4.2 EDL Testing

The EDL testing is a special process in the verification – validation process. The reason is that EDL tests should reveal the efficiency of the system when communicating with experimenters not only through the provided functionality perspective but also through the easiness that an experimenter can create, compile and run an experiment. The aim of the EDL testing is to reveal if the scenario defined by the experimenter is smoothly processed and produces the appropriate outcomes to be adopted by the remaining RAWFIE components. Specific tests will be realized concerning important characteristics of the EDL as well as the functionalities provided by the editors. For instance, the testing process will involve two aspects: (a) the experimenter side and (b) the components side. From the experimenter point of view, the provided editors and their functionalities should be easily initiated and

commands (i.e., EDL scripts) should be efficiently translated based on the underlying EDL model. In RAWFIE, experimenters that create an experiment will need to provide a short high level description of the experiment and its purpose. The second aspect involves the definition of specific commands in the test script that will reveal if the RAWFIE components are smoothly combined. This will also test the connection between components in order to have an efficient execution of the experiment.

The test scenarios will be realized based on the defined use cases and reveal if an experimenter is capable of easily define an experiment in the EDL terms. For instance, with test scenarios, critical questions will be answered like: Can the experiment easily define the application logic of his/her experiment? How easily the experimenter can define an experiment that realizes a complex algorithm? Moreover, the test scenarios will check if the EDL script is efficiently translated based on the underlying model and, accordingly, be compiled and validated. Syntactic and semantic errors will be incorporated in the test scenarios in order to reveal if the system is capable of identifying the errors and return specific messages to the experimenter. Successful fulfilment of the compilation and the script validation process will be realized through a number of files / models assigned to specific RAWFIE components. These files / models are necessary to, finally, execute the experiment.

3 Stakeholders and actors

Stakeholders to be considered in the validation plan, include both end users that are interested in the experimentation of specific technologies, as well as personnel of specialised, NGO or GO organisations, that can use the RAWFIE platform and testbeds for simulating specific mission scenarios linked to their by day operations. All these types of actors, some of them identified in D3.1/D3.2, are in the following represented by the common category “Experimenters”. Those who are the main candidate for evaluating the appropriateness of the RAWFIE platform and testbeds to support their requirements are:

- Experimenters:
    - Users who belong to the federation. They must be acknowledged by the federation partners. As said, these include different stakeholders like e.g.:
        - Governmental Organizations responsible for SAR operations
        - Non-Governmental Organizations aiding SAR operations
        - Command and Control Operation centres
- RAWFIE Admin:
    - Administrator of RAWFIE frontend and middleware framework. These are owned and maintained by the RAWFIE consortium
- Testbed Operators:

  - Owners and managers of testbed facilities

- UxV Manufacturers:
  - Suppliers of UxVs resources

4 Metrics

4.1 Introduction

In D4.9, the metrics has been kept identical to those presented in D4.6. They are however reproduced in D4.9 for information. Note also the following information, taken from D4.6:

- The metric types as defined in D3.2 are: PERF= performance, FUNC=functional, USE=usability, DATA=data. It has to be noted however, that metrics linked to one or more D3.2 requirements are not necessarily of the same type of the requirement/s they are linked to. A specific validation metric could be, for example, of type USE because focussed on usability from the validators (end users) perspective, while being connected to functional requirements

The following metrics categories are still taken into account:

- PLATFORM – metrics related to the whole RAWFIE frontend and middleware platform behaviour
- TESTBED – metrics related to testbeds availability / information
- INTERCONNECTIVITY – metrics related mainly to communication performances
- UxV - metrics related to UxVs availability / information

Once the system has been verified and deployed at the testbed sites, a period of evaluation will take place during which the abovementioned metrics will be assessed either by quantifiable measurements or by way of questionnaires/interviews. It should be noted that not all of the defined validation metrics can be directly and explicitly expressed in the validation scenarios described in the following of the document.

Nevertheless, the needed actions will be put in place for being able to measure them and evaluate them against the related success criteria. This applies for example, to the metrics related to the monitoring and acquisition of particular parameters / statistics (errors, notifications, etc.), as well as to most of the usability related ones (type = USE), for which dedicated questionnaires will be prepared before running the validation sessions, to be submitted to the validators.

For the description of other specific metrics attributes like required or beneficial, hard or soft, please refer to the previous deliverable D4.3 and D4.6.

4.2 Metric template definition

Below is the new, updated version of the metrics definition / description template

Table 2: Metrics template

Metric category/ Type/ ID / Tag	Description	Required or Beneficial Hard or Soft	Mean for measurement	Validator stakeholder	Success Criteria	Req. Id (D3.1-D3.2)

4.3 Success criteria

Success criteria are quantitative or qualitative values (or set or ranges of values) for relevant metrics, against which the actual characteristics or performance indicators of the system and components are compared. A typical criterion is a threshold against which the performance indicator of the tested element is compared (e.g. “the temperature of the motor shall not exceed 90°C during the experiment”).

The success criteria are usually combined to perform the evaluation of a given element. For example, an element will be successfully evaluated if it meets the criteria A and B and C. Another element may be successfully evaluated if it meets the criteria B and C or F.

For any given metrics, the success criteria may vary depending on the components under evaluation, or on the experiment under execution. To this intent, a template is provided to specify criteria for any component or system to be evaluated.

4.4 Platform metrics

Table 3: Platform metrics.

Metric category/ Type / ID / Tag	Description	Required or Beneficial Hard or Soft	Mean for measurement	Validator stakeholder	Success Criteria	Req. Id (D3.1-D3.2)
PLATFORM / PERF	Measures the performances of the system as a whole according to specific sub-criteria described in the following					
PLATFORM / PERF / 1 / STABLE SYSTEM	Measures the system uptime and detect system downtimes	Required Hard	System monitoring System logs	RAWFIE Admin, Testbed Operator, Experimenter	Downtime < 2%	PT-SYM-T-001 PT-SYM-T-004
PLATFORM / PERF / 2 / ERRORS	Counts RAWFIE platform errors and crashes	Required Hard/Soft	System monitoring System logs Tickets received from the end users	RAWFIE Admin Experimenter	The target is to keep the number of received tickets to the minimum possible, i.e. under a threshold of the 5% of the total number of executed experiments	PT-EXP-C-009
PLATFORM / PERF / 3 / SCALABILITY	Number of concurrent running experiments. Number of users interacting with the platform (e.g. for creating experiments, visualise and analyse results, and so on.	Required Hard	System statistics and monitoring (e.g. users' accesses). Stress tests by launching a certain number of experiments (the maximum allowed by the available testbeds & resources) in parallel. Registering the number of successfully executed experiments, and date/time of execution.	RAWFIE Admin	By design, the target (success criteria) is to have a platform that can scale horizontally, provided the needed server instances are setup in the Cloud environment. Therefore this metric should only be dependent on the number of available testbeds and UxVs at each given time	PT-NF-006 from D3.1

PLATFORM / PERF / 4 / RECOVERY TIME	Records the time needed to recover the system operations after the shutdown / failure of specific parts which are needed for normal system use (i.e. Web frontend server, Middle Tier server/services, Message Bus servers cluster, and so on). Testbeds and UxVs unavailability is excluded as they are independent from the central platform	Required Hard	System monitoring Statistics, collected through dedicated tests for simulating the unavailability of specific services / servers	RAWFIE Admin	The system should be operational again after one or more server / services shutdown, in less than 5 minutes. This should happen either automatically (thanks to the used cloud facilities and setup), or event manually in case of problems affecting the functionality of specific services, requiring a technical intervention. In this latter case, the time is calculated starting from when the problem causing the shutdown of the server / service has been solved, and the operator himself has started again the affected servers / services	PT-SYM-S-004
PLATFORM / PERF / 5 / LATENCY/ RESULTS UPDATE TIME	Latency between the real execution of commands or the acquisition of measurements and results, and the update of the same info in the visualisation tools	Required Hard	System monitoring and statistics / logs	RAWFIE Admin Experimenter	< 5 seconds	
PLATFORM / PERF / 6 / LATENCY/ BOOKING TIME	Time for the user to receive the notification of "experiment booked" after completing the request procedure through the UI (e.g. completion of all queries for selecting the needed testbeds and resources also in a federated environment).	Required / Hard	System monitoring and statistics / logs	RAWFIE Admin Experimenter	< 30 seconds	

PLATFORM / USE	Measures the usability of the system as a whole, or of different GUI tools and functions, according to specific sub-criteria (provided notifications, ease of access, clarity, engagement, motivation, etc..) described in the following					PT-WEB-P-001
PLATFORM / USE / 7 / NOTIFICATION	Measures the quality and usefulness of the notifications provided by the different system tools	Required Soft	End users' questionnaires / interviews, aimed at checking whether the notifications provided by specific GUI tools, are understandable and properly provided.	Experimenter	Through the answers to specific questions (for each GUI tool), users will be asked to give a score from 1 to 5. The metric will be considered as positively evaluated if an average score of at least 3.5 will be reached.	PT-BOO-T-010 PT-BOO-T-010 PT-EXV-S-001 PT-BOO-S-011 PT-LAU-S-008 PT-LAU-S-012 PT-EXP-C-008 PT-EXP-C-009
PLATFORM / USE / 8 / ROLES	RAWFIE platform shall support various roles with different privileges at every level of access	Required Soft	End users' questionnaires / interviews, aimed at checking whether the role management is provided as end users' expect.	Experimenter	Through the answers to specific questions (for each GUI tool), users will be asked to give a score from 1 to 5. The metric will be considered as positively evaluated if an average score of at least 3.5 will be reached.	PT-WEB-P-002 PT-SYM-T-003 PT-USR-S-001 PT-USR-S-002
PLATFORM / USE / 9 / VISUALISATION / BALANCE	End user estimate the distribution of the optical weight in the GUI (number of objects) in a picture via questionnaires	Required Soft	Questionnaire	Experimenter	Through the answers to specific questions (for each GUI tool), users will be asked to give a score from 1 to 5. The metric will be considered as positively evaluated if an average score of at least 3.5 will be reached.	PT-BOO-T-009

PLATFORM / USE / 10 / VISUALISATION / SIMPLICITY	Experimenter evaluates if the objects appearing to the screen are the minimum needed and easily accessible	Required Soft	Questionnaire	Experimenter	Through the answers to specific questions (for each GUI tool), users will be asked to give a score from 1 to 5. The metric will be considered as positively evaluated if an average score of at least 3.5 will be reached.	PT-BOO-T-009 PT-VIS-E-002
PLATFORM / USE / 11 / VISUALISATION / CONSISTENCY	Experimenter evaluates if similar actions lead to similar results and the elements in the GUI (fonts, patterns, tables) are similar to all pages	Required Soft	Questionnaire	Experimenter	Through the answers to specific questions (for each GUI tool), users will be asked to give a score from 1 to 5. The metric will be considered as positively evaluated if an average score of at least 3.5 will be reached.	PT-BOO-T-009 PT-LAU-S-003
PLATFORM / USE / 12 / VISUALISATION / UTILITY	Experimenter evaluates the utility of the different tools in order to define, manage and execute an experiment	Required Soft	Questionnaire	Experimenter	Through the answers to specific questions (for each GUI tool), users will be asked to give a score from 1 to 5. The metric will be considered as positively evaluated if an average score of at least 3.5 will be reached.	PT-EXA-T-002 PT-VIS-E-002
PLATFORM / USE / 13 / GUIDANCE	Experimenter tests if help guidance or error messages appear in order to guide him/her to the right option	Required Soft	Questionnaire	Experimenter	Through the answers to specific questions (for each GUI tool), users will be asked to give a score from 1 to 5. The metric will be considered as positively evaluated if an average score of at least 3.5 will be reached.	PT-EXA-T-002
PLATFORM / USE / 14 / FILTERING	Usefulness and efficiency of provided filtering functionalities of the different tools	Required Soft	Questionnaire	Experimenter	Through the answers to specific questions (for each GUI tool), users will be asked to give a score from 1 to 5. The metric will be considered as positively evaluated if an average score of at least 3.5 will be reached.	PT-SYM-T-003 PT-REE-T-003 PT-EXA-T-006 PT-VIS-T-005 PT-DAA-T-004 PT-DIR-S-002

PLATFORM / USE / 15 / EXPERIMENTS STATISTICS	It should be possible to check if the same or similar experiment configuration (parameters) lead to problems (UxV collisions, crashes, system failures, etc.) in the past	Beneficial Hard/Soft	System monitoring, logs, questionnaire	RAWFIE Admin, Experimenter	RAWFIE Admin validate the quality and quantity of provided information from past experiment. Through the answers to specific questions (for each GUI tool), users will be asked to give a score from 1 to 5. The metric will be considered as positively evaluated if an average score of at least 3.5 will be reached.	PT-DAA-T-002
PLATFORM / FUNC / 16 / STORAGE	System ability to store experiment data in case of comm. link failure between the testbed and the upstream components deployed in the cloud	Required Hard	System Monitoring, logs Check stored data	RAWFIE Admin	The system should be able to provide, for visualisation and analysis purposes, all (100%) results related to the experiments that were running when the link communication failure happened	PT-GEN-R-004 PT-VIS-E-004
PLATFORM / FUNC / 17 / EXTENSIBILITY	This metric is aimed at assessing how easy is to extend the platform in terms of: A) new services / functionalities; B) New testbeds and UxVs provided the architectural guidelines and requirements are respected by new testbed and UxVs owners, and with or without (SFA based) federation	Required Soft/Hard	Conceptual evaluation by RAWFIE technicians and stakeholders	RAWFIE Admin Testbed Operators UxVs Manufactures	The different architectural elements (from Frontend Tier to MiddleTier services to testbeds and UxVs) should be easily “plugged”, form the software perspective, with the minimum effort, by just using configuration capabilities and APIs that are provided by the RAWFIE platform components themselves	

4.5 Testbed metrics

Table 4: Testbed metrics

Metric category/ Type / ID / Tag	Description	Required or Beneficial Hard or Soft	Mean for measurement	Validator stakeholder	Success Criteria	Req. Id (D3.2)
TESTBED / DATA / 1 / INFORMATION	Capability of a testbed to provide the users, through the RAWFIE platform, information relevant for booking and running experiments, such as: weather conditions, UxV availability and capabilities, sensors, whole testbed availability time	Required Hard/Soft	Testbed monitoring (and finally notifications to the users) Users' questionnaires This can also be calculated via integration tests. We have different measures from the manufacturers about the bottom line of executing an experiment	RAWFIE Admin, Experimenter	Weather conditions, overall testbed status, as well as information on UxVs and sensors, should be updated at least **daily** by the Testbed Operator during the periods when the Testbed is up and running. And made available for the experimenter the 100% of the time.	TB-MOM-001 TB-MOM-002 TB-MOM-003 TB-MOM-004 PT-EXP-C-006 PT-EXP-C-008 PT-SYM-002 TB-GEN-R-001 TB-GEN-002 TB-MAN-003
TESTBED / FUNC / 2 / SECURITY	Capability of the Testbed to provide a secure environment, with firewall rules for avoiding harmful accesses to the rest of the RAWFIE platform. It could also be based on a DMZ containing only the Testbed components which need to be reached from the rest of RAWFIE components	Required Hard	Dedicated security tests	Admin	The success criteria is defined as the set of rules that will need to be satisfied in order to avoid unauthorised accesses to the different components, at both RAWFIE platform and Testbed side.	TB-PRO-002

TESTBED / FUNC / 3 / AVAILABILITY	Measure the Testbeds availability for performing experiments, in a certain period of time	Required Hard	System monitoring & notifications. Users' experience	Admin, Experimenter	Success criteria will be that the amount of days of testbed availability in total, will be exactly as declared by RAWFIE Testbed Operators at the beginning and in the Open Calls proposals. Downtime for maintenance, as well as other planned unavailability which may prevent the execution of the experiments should be communicated in advance, at least 2 days before.	
TESTBED / USE / 4 / CONSISTENCY	This metric is intended to measure if the remote users of the scenario were able to perform their tests as they expected (e.g. the run experiment was exactly what they asked for)	Required Soft	Users' experience	Experimenter	Through the answers to specific related questions, users will be asked to give a score from 1 to 5. The metric will be considered as positively evaluated if an average score of at least 3.5 will be reached.	PT-EXP-C-009

4.6 UxV metrics

Table 5: UxV metrics

Metric category/ Type / ID / Tag	Description	Required Or Beneficial Hard Or Soft	Mean for measurement	Validator stakeholder	Success Criteria	Req. Id (D3.2)
UxV / FUNC / 1 / COHERENCE	Actual route vs. plan	Required Hard/Soft	Statistics of the UxV collected during the experiment	RAWFIE Admin, Experimenter	Through the answers to specific related questions, users will be asked to give a score from 1 to 5 to this metric. The metric will be considered as positively evaluated if an average score of at least 3.5 will be reached. For a more rational evaluation, the following formulas could define a threshold depending on the application requirements and tesbed accuracy: Max deviation from the route in meter below the threshold. Or in percentage of the accuracy. Average deviation from the route in meter below the threshold. Or in percentage of the accuracy.	TB-REC-003 TB-REC-004 TB-REC-005
UxV / FUNC/ 2 / MISSION ACHIEVEMENT	Actual mission achievement	Required Hard/Soft	Experiment statistics: rate of achieved vs. assigned objectives	RAWFIE Admin, Experimenter	Through the answers to specific related questions, users will be asked to give a score from 1 to 5 to this metric. The metric will be considered as positively evaluated if an average score of at least 3.5 will be reached. Rate of achieved vs. assigned objectives greater than a given threshold .	TB-REC-003 TB-REC-004 TB-REC-005
UxV / PERF / 4 / BATTERY LIFETIME	Counts battery lifetime per experiment	Required Hard	System Monitoring. UxV node parameters and status	RAWFIE Admin, UxV Manufactors	Battery autonomy of each device should be between 15 and 30 minutes	UXV-NOD-002

4.7 Interconnectivity (aka. communication) metrics

Communication metrics are related to traditional networking and communication parameters like throughput, end-to-end delay (latency), and maximum allowed communication distance with the described below might be applied both to the *local* communication between the UxVs and the Resource Controller at the testbed side, as well as the remote communication with the rest of RAWFIE platform.

Metric type/ ID/ Tag	Description	Required or Beneficial Hard or Soft	Mean for measurement	Validator stakeholder	Success criteria	Req. Id (D3.2)
INTERCONNECTIVITY / PERF / 1 / AGGREGATED THROUGHPUT	Aggregated data throughput for the whole RAWFIE platform, expressed as the maximum number of messages processed in the unit of time	Required Hard	System monitoring. Components measurements. By the mean of stress tests, messages of different fixed size (e.g. typical average sized RAWFIE messages) will be processed for a given workflow (e.g. a given validation scenario). At the end of the test, the total processed number of messages in the given amount of time is retrieved, and the conversion in bytes per second is finally realised	RAWFIE Admin Testbed Operator (for performing validation scenarios)	The actual, acceptable throughput for the correct execution of realistic experimentation scenarios is part of the research activities. The validation will, in this case, aimed at A) assessing the performances of the provided integration and communication solution The aggregated throughput will be calculated for different workflows (e.g. corresponding to some of the validation scenarios)	
INTERCONNECTIVITY / PERF / 2 / COMPONENTS THROUGHPUT	Data throughput ensured by different RAWFIE components, for both the intra-testbed communication (especially Resource Controller-to-UxVs) and inter-tier communication	Required Hard	System monitoring. Components measurements. By the mean of stress tests, messages of different fixed size (e.g. typical average sized RAWFIE messages) will be processed for a given communication scenario (e.g. between 2 components in a validation scenario). At the end of the test, the total processed number of messages in the given amount of time is retrieved, and the conversion in bytes per second is finally realised	RAWFIE Admin Testbed Operator	See the previous metric	

INTERCONNECTIVITY / PERF / 3 / END-TO-END DELAY	The total time it takes for a packet to reach its destination after being sent. Especially relevant for the communication between the resource controller and the UxVs (local, testbed level), but in general for any other kind of components' communication scenario	Required Hard	Stress tests are performed by continuously sending packets of fixed size (e.g. average size of RAWFIE messages exchanged between the Resource Controller and the UxVs). Each packet is sent with a timestamp (the sender and the receiving entities (where the involved components are running are synchronised with the same time). At the receiver side, for each received packet, the difference between the receiving time and the original timestamp is calculated	RAWFIE Admin Testbed Operator	The actual, acceptable end-to-end delay for UxV controlling is part of the research activities. The validation will, in this case, aimed at A) assessing the performances of the provided integration and communication solution and B) finding outcomes of the impact of the latency in semi-autonomous devices controlling	

5 Verification

The verification of components is included in this chapter in an attempt to capture, from the earliest stage of the project, as most input as possible discussing the scenarios and tests about the verification and validation.

5.1 Verification scenarios

5.1.1 Frontend Tier

The Front-end tier mostly consists in User interfaces, in particular the Web Portal GUI elements.

5.1.1.1 Web Portal

Table 6: Verification test of the Web Portal - Login/ Logout

Test ID**: WP01**	Conducted by**:**	Date**:**	Test Category**: Verification Tests (front end tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Web Portal - Login/ Logout***		
Preconditions	• User registered in the User & Rights repository		
Related Requirements	PT-WEB-P-001, PT-WEB-P-002		
Tools Used			

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	user opens RAWFIE any web page	redirect to login page, login form displayed		
2	user enters invalid credentials and submits the form	error message displayed		
3	user enters valid credentials and submits the form	redirect to start page		
4	user press the logout button	redirect to login page, login form displayed, logout message displayed		

Table 7: Verification test of the Web Portal – Language selection

Test ID: **WP02**	Conducted by:	Date:	Test Category: **Verification Tests (front end tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Web Portal – Language selection***		
Preconditions	• Translation available		
Related Requirements	PT-WEB-P-001		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	user opens RAWFIE any web page	web page with language selection displayed,		
2	user changes the language	web page displayed in the selected language		

Table 8: Verification test of the Web Portal – User registration

Test ID: **WP03**	Conducted by:	Date:	Test Category: **Verification Tests (front end tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Web Portal – User registration***		
Preconditions	• Admin login available • No pending registration request		
Related Requirements	PT-WEB-P-002		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Browser 1: login as administrator and open user management page	management page displayed		
2	Browser 1: Navigate to registration requests page	No registration request displayed		
3	Browser 2: Open register form, fill in form (login credentials, personal data, etc.) and submit	Registration request stored and confirmation shown to the user.		
4	Browser 2:Try to login with the submitted login credentials	Login failed. Display message that user is looked		
5	Browser 1: Reload registration requests page	The new registration request is show		
6	Browser 1: Accept the new user	The new user is now unlooked		
7	Browser 2: Try to login with the submitted login credentials	Login successful.		
8	Browser 1: Navigate to the user list and delete the new user	User deleted		
9	Browser 2: Logout and try to login with the submitted login credentials	Login failed. Show invalid credentials messages		

5.1.1.2 Wiki Tool

Table 9: Verification test of the Wiki Tool – Component Help

Test ID: **WT01**	Conducted by:	Date:	Test Category: **Verification Tests (front end tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Wiki Tool – Component help***		
Preconditions	• Help pages added to the Wiki		
Related Requirements	PT-WIK-001, PT-WIK-003		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Login to the Web Portal and open Resource Explorer	Resource Explorer page displayed		
2	Click on the Help icon	Wiki Tool opened with the article about Resource Explorer		
3	Change display language in the Wiki	Wiki article displayed in another language		
4	Repeat step 2 of other pages (like Visualization Tool, Booking tool, etc.)	Wiki Tool opened with the article about other tools		

Table 10: Verification test of the Wiki Tool – Editing

Test ID: **WT02**	Conducted by:	Date:	Test Category: **Verification Tests (front end tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Wiki Tool – Editing***		
Preconditions	• User for Wiki management defined		
Related Requirements	PT-WIK-001, PT-WIK-002, PT-WIK-004		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Login to the Web Portal as normal experimenter and open a page in the Wiki Tool	Wiki page displayed		
2	Try to edit the page	Editing not possible due to missing rights		
3	Login as administrator and assign the Wiki manager right to the user	The user has now the Wiki manager right		
4	Login as the first user and open a page in the Wiki Tool	Wiki page displayed		
5	Try to edit the page	Editing allowed and changes are save		

5.1.1.3 Resource Explorer Tool

Table 11: Verification test of the Browse testbeds and UxVs and start booking

Test ID**: RET01**	Conducted by**:**	Date**:**	Test Category**: Verification Tests (front end tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Browse testbeds and UxVs and start booking***		
Preconditions	• connection to the Testbeds Directory Service OK • data about testbeds and UxVs available		
Related Requirements	PT-REE-T-001, PT-REE-T-003, PT-REE-T-004		
Tools Used			

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	user opens Resource Explorer Tool in the Web Portal	Resource Explorer Tool displays a view with all available testbeds		
2	User set some filter parameters too find a testbed fitting to its needs	Resource Explorer Tool displays only the testbeds fitting to the filter		
3	user selects a testbed	Resource Explorer Tool displays all testbed details and a list of available UxVs		
4	user selects a UxV	Resource Explorer Tool displays all UxVs details		
5	user starts booking	Booking Tool opened with the selected resources		

5.1.1.4 Booking Tool

Booking Tool requirements PT-BOO-T-015 is implemented by integration of the tool to the Web Portal which ensures authorized access is only available.

Test Procedures BT01, BT02 have been updated with extra steps added. Test Procedures BT03, BT04 remain unchanged compared to what was defined in the previous version of the deliverable (D4.6). Test procedure BT05 is new.

Table 12: Verification test of the Booking Tool Calendar View and its display options

Test ID: **BT01**	Conducted by:	Date:	Test Category: **Verification Tests (web tier)**
Hardware Configuration	-		
Software Configuration	-		
Test Name:	***Booking Tool Calendar View and display options***		
Preconditions	• connection to the Booking Service ok • user has logged in the web portal • reservations of different status exist in the Master DB		
Related Requirements	PT-BOO-T-001 PT-BOO-T-003 PT-BOO-T-006 PT-BOO-T-010 PT-BOO-T-015 PT-BOO-T-016 PT-BOO-S-008		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Click of Bookings menu item	Navigation to Booking Tool (Calendar View)		
		Calendar view displays by default the present week with all defined bookings		
2	Switch Calendar display to display week, month, day interval via the appropriate options	Calendar view changes to present the selected interval with all defined bookings		
3	Navigate back and forth in time via the provided navigation buttons (for every selection made in step 2)	Calendar view changes to previous or future date time intervals and displays even past reservations		
4	Verify by inspection of existing reservations that only reservations of certain status are visible in the Calendar View	Reservation of status PENDING, OK or REJECTED should only be displayed		
5	While in Calendar view, switch between different testbeds by changing selection in the corresponding combo box	Reservations only for the selected testbeds are available		new step added in D4.9
6	(Repeat action in step 5)	While selecting different testbeds verify also that the displayed Calendar timeslots adhere to the testbed operational hours as defined in the Testbed DB table		new step added in D4.9
7	Check filtering of calendar displayed events by setting/modifying the filter textbox and clicking the apply button	Based on filter options certain booking events may become visible or invisible		new step added in D4.9

Table 13: Verification test of the Booking Tool Calendar View Interactions

Test ID: **BT02**	Conducted by:	Date:	Test Category: **Verification Tests (web tier)**
Hardware Configuration	-		
Software Configuration	-		
Test Name:	***Booking Tool Calendar View Interactions***		
Preconditions	• connection to the Booking Service ok • user has logged in the web portal • reservations of different status exist in the Master DB		
Related Requirements	PT-BOO-T-001 PT-BOO-T-003 PT-BOO-T-005 PT-BOO-T-006 PT-BOO-T-016 PT-BOO-S-002 PT-BOO-S-004		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Click on an empty calendar timeslot (result should depend on the relevance of the timeslot to the present time)	If click occurs on a past timeslot a popup warning is displayed		
		If click occurs on a future timeslot the "Create Reservation" window opens		
2	Click on an existing reservation (result should depend on the relevance of the reservation to the present time)	If click occurs on a past reservation the "Edit Reservation" window opens but no further actions are offered to the user		
3	(see also test BT04)	If click occurs on a future reservation the "Edit Reservation" window opens and the user can perform certain actions on the reservation. Displayed actions depend on user role and reservation status		
4	verify the displayed color for each reservation (click existing reservations)	Coloring of reservation should differ based on the reservation status (shown in the Edit Reservation window)		
5	Perform steps 1-3 after selecting different testbeds in the provided drop down list	Verify that when a testbed is selected in the corresponding Calendar view drop down box then only resources from this		new step added in D4.9

		specific testbed are displayed in all popup windows (Create/Edit/View reservations)		
7	verify the time options available during reservation edit/create	The time steps for begin and end time should not fall outside the testbed defined operation hours		new step added in D4.9

Table 14: Verification test of the Booking Tool Create Reservation

Test ID: **BT03**	Conducted by:	Date:	Test Category: **Verification Tests (web tier)**
Hardware Configuration	-		
Software Configuration	-		
Test Name:	***Booking Tool Create Reservation***		
Preconditions	• connection to the Booking Service ok • user has logged in the web portal • user has clicked on an empty future timeslot		
Related Requirements	PT-BOO-T-001 PT-BOO-T-003 PT-BOO-T-004 PT-BOO-T-009 PT-BOO-T-010 PT-BOO-T-011 PT-BOO-S-006		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	User edits the field of the "Create Reservation" form so that no time overlapping with other reservation exists and presses the OK button (no conflicts scenario)	Reservation is created and displayed in the Calendar View. Reservation is put in PENDING state		
2	User edits the field of the "Create Reservation" form so that a time overlapping with other reservation exists and presses the OK button (possible conflict scenario)	If no common resources exist with the overlapping reservations then the new reservation is created and displayed in the Calendar View. Reservation is put in PENDING state		
		If common resources exist with the overlapping reservations then the new reservation is not created and a warning message is displayed		Result may depend on status of pre-existing reservation

Table 15: Verification test of the Booking Tool Edit Reservation Actions

Test ID: **BT04**	Conducted by:	Date:	Test Category: **Verification Tests (web tier)**
Hardware Configuration	-		
Software Configuration	-		
Test Name:	***Booking Tool Edit Reservation Actions***		
Preconditions	• connection to the Booking Service ok • user has logged in the web portal • user has clicked on an existing future reservation		
Related Requirements	PT-BOO-T-003 PT-BOO-T-005 PT-BOO-T-007 PT-BOO-T-008 PT-BOO-T-010 PT-BOO-T-011 PT-BOO-T-013 PT-BOO-T-014 PT-BOO-S-006 PT-NF-002		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	The actions available to the Edit Reservation window depend on the: • status of reservation • user • role of the user			
	status=PENDING user= owner of reservation role= EXPERIMENTER	Actions available: OK, CANCEL DELETE		
	status=OK user= owner of reservation role= EXPERIMENTER	Actions available: OK, CANCEL DELETE		
	status=REJECTED user= owner of reservation role= EXPERIMENTER	Actions available: OK, CANCEL DELETE		
	status=PENDING user= owner of reservation role= TESTBED_OP	Actions available: OK, CANCEL, DELETE, APPROVE, REJECT		
	status=PENDING user= not owner of reservation role= TESTBED_OP	Actions available: CANCEL, APPROVE, REJECT		
	status=OK user= owner of reservation role= TESTBED_OP	Actions available: CANCEL, DELETE, REJECT		
	status=OK user= not owner of reservation role= TESTBED_OP	Actions available: CANCEL, REJECT		
	status=REJECTED user= owner of reservation role= TESTBED_OP	Actions available: CANCEL, DELETE, APPROVE		
	status= REJECTED user= not owner of reservation role= TESTBED_OP	Actions available: CANCEL, APPROVE		

	user= not owner of reservation	No actions available		
2	Owner of reservation performs changes to the reservation and presses OK button	If the changes do NOT introduce conflicts in both timeslots and selected resources, then the reservation is successfully updated and the UI refreshed to display the changes		
		If the changes do introduce conflicts in both timeslots and selected resources, then a warning message appears and no further action is performed		
3	Owner of reservation presses DELETE button	If reservation does not refer to a currently running experiment, then it is put in a CANCELLED state and removed from the UI		
4	User with TESTBED_OP role presses APPROVE button	If no resource conflicts with already created reservation exists then reservation status becomes OK and color changes appropriately in the Calendar view		
5	User with TESTBED_OP role presses REJECT button	reservation status becomes REJECTED and color changes appropriately in the Calendar view		

Table 16: Verification test of the Booking Tool SFA integration

Test ID: **BT05**	Conducted by:	Date:	Test Category: **Verification Tests (web tier)**
Hardware Configuration	-		
Software Configuration	-		
Test Name:	***Booking Tool SFA Integration***		
Preconditions	• connection to the Booking Service ok • connection to the SFA Aggregate Manager ok • user has logged in the web portal • user has clicked on an empty future timeslot		
Related Requirements	PT-BOO-T-002		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Replicate all steps defined in BT03 (creation of the reservation)	Verify by the SFA UI (i.e. MySlice) that there exists a reservation for the involved resources in the Aggregate Manager data store		
2	Replicate steps 3 & 4 of BT04	Verify the status of reservation is also updated in Aggregate Manager		
3	Perform a reservation of resources from the MySlice interface`	After refreshing the calendar view, verify that a reservation exists for these resources		

5.1.1.5 Experiment Authoring Tool

Table 17: Verification test of the in-Textual Editor Experiments definition

Test ID**: EAT01**	Conducted by**:**	Date**:**	Test Category**: Verification Tests (front end tier – middle tier)**
Hardware Configuration	-		
Software Configuration			
Test Name:	***Define Experiments in the Textual Editor***		
Preconditions	• User entered in the RAWFIE Portal		
Related Requirements	PT-EXA-T-001, PT-EXA-T-002, PT-EXA-T-003, PT-EXA-T-004, PT-EXA-T-005, PT-EXA-T-007, PT-EXA-T-008, PT-EXA-T-009, PT-EXA-T-010, PT-EXA-T-011, PT-EXA-T-012, PT-EXA-T-013, PT-EXA-T-014, PT-EXA-T-015, PT-EXA-T-016		
Tools Used	• RAWFIE Web Portal • RAWFIE Textual Editor		

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Access to the Textual Editor through the RAWFIE Web Portal	Redirection to the Textual Editor interface		
2	Write an experiment	Experiment is presented in the editor		
3	Utilize code completion, content assist and compilation	The editor responds with specific drop down lists, messages, etc.		
4	Define erroneous commands in the experiment workflow	The editor responds with error messages and indication for correcting the error		
5	Save the experiment	The experiment is stored in the database and specific files are produced to be adopted by the remaining RAWFIE components		

Table 18: Verification test of the Textual Editor Experiments Update

Test ID: **EAT02**	Conducted by:	Date:	Test Category: **Verification Tests (front end tier – middle tier)**
Hardware Configuration	-		
Software Configuration			
Test Name:	***Update Experiments in the Textual Editor***		
Preconditions	• User entered in the RAWFIE Portal		
Related Requirements	PT-EXA-T-001, PT-EXA-T-002, PT-EXA-T-003, PT-EXA-T-004, PT-EXA-T-005, PT-EXA-T-007, PT-EXA-T-008, PT-EXA-T-009, PT-EXA-T-010, PT-EXA-T-011, PT-EXA-T-012, PT-EXA-T-013, PT-EXA-T-014, PT-EXA-T-015, PT-EXA-T-016		
Tools Used	• RAWFIE Web Portal • RAWFIE Textual Editor		

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Access to the Textual Editor through the RAWFIE Web Portal	Redirection to the Textual Editor interface		
2	Open an already defined experiment	Experiment is presented in the editor		
3	Makes changes in the experiment workflow	The experiment is updated		
4	Save the experiment	The experiment is stored in the database and specific files are produced to be adopted by the remaining RAWFIE components		

Table 19: Verification test of the in-Visual Editor Experiments Define

Test ID: **EAT03**	Conducted by:	Date:	Test Category: **Verification Tests (front end tier – middle tier)**
Hardware Configuration	-		
Software Configuration	•		
Test Name:	***Define Experiments in the Visual Editor***		
Preconditions	• User entered in the RAWFIE Portal		
Related Requirements	PT-EXA-T-001, PT-EXA-T-002, PT-EXA-T-003, PT-EXA-T-004, PT-EXA-T-005, PT-EXA-T-007, PT-EXA-T-008, PT-EXA-T-009, PT-EXA-T-010, PT-EXA-T-011, PT-EXA-T-012, PT-EXA-T-013, PT-EXA-T-014, PT-EXA-T-015, PT-EXA-T-016		
Tools Used	• RAWFIE Web Portal • RAWFIE Visual Editor		

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Access to the Visual Editor through the RAWFIE Web Portal	Redirection to the Visual Editor interface		
2	Access the available toolbar	Specific windows are presented		
3	Create an experiment by utilizing the available tools	The experimenter can define waypoints and experiment information by clicking and designing in the visual editor		
4	Define erroneous commands	The authoring tool responds with error messages and indication for correcting the error		
5	Save the experiment	The experiment is stored in the database and specific files are produced to be adopted by the remaining RAWFIE components		

Table 20: Verification test of the in-Visual Editor Experiments Update

Test ID: **EAT04**	Conducted by:	Date:	Test Category: **Verification Tests (front end tier – middle tier)**
Hardware Configuration	-		
Software Configuration			
Test Name:	***Update Experiments in the Visual Editor***		
Preconditions	• User entered in the RAWFIE Portal		
Related Requirements	PT-EXA-T-001, PT-EXA-T-002, PT-EXA-T-003, PT-EXA-T-004, PT-EXA-T-005, PT-EXA-T-007, PT-EXA-T-008, PT-EXA-T-009, PT-EXA-T-010, PT-EXA-T-011, PT-EXA-T-012, PT-EXA-T-013, PT-EXA-T-014, PT-EXA-T-015, PT-EXA-T-016		
Tools Used	• RAWFIE Web Portal • RAWFIE Visual Editor		

Step	Action	Expected Result	Status	Remarks
1	Access to the Visual Editor through the RAWFIE Web Portal	Redirection to the Visual Editor interface		
2	Open an already defined experiment	Experiment is presented in the editor		
3	Makes changes in the experiment workflow	The experiment is updated		
4	Save the experiment	The experiment is stored in the database and specific files are produced to be adopted by the remaining RAWFIE components		• RAWFIE Web Portal RAWFIE Textual Editor

Table 21: Verification test of the Editor switching

Test ID: **EAT05**	Conducted by:	Date:	Test Category: **Verification Tests (front end tier – middle tier)**
Hardware Configuration	-		
Software Configuration	•		
Test Name:	***Switch between the Editors***		
Preconditions	• User entered in the RAWFIE Portal		
Related Requirements	PT-EXA-T-001, PT-EXA-T-002, PT-EXA-T-003, PT-EXA-T-004, PT-EXA-T-005, PT-EXA-T-008, PT-EXA-T-009, PT-EXA-T-010, PT-EXA-T-011, PT-EXA-T-012, PT-EXA-T-013, PT-EXA-T-015		
Tools Used	• RAWFIE Web Portal • RAWFIE Textual Editor • RAWFIE Visual Editor		

Step	Action	Expected Result	Status	Remarks
1	Access to the editors through the RAWFIE Web Portal	Redirection to the editor interface		
2	Create an experiment	Experiment is presented in the editor interface		
3	Switch to the alternative editor and make changes	The experiment is updated		
4	Save the experiment	The experiment is stored in the database		

		and specific files are produced to be adopted by the remaining RAWFIE components		

Table 22: Verification test of the experiment Launchings

Test ID**: EAT06**	Conducted by**:**	Date**:**	Test Category**: Verification Tests (front end tier – middle tier)**
Hardware Configuration	-		
Software Configuration	•		
Test Name:	***Launch experiments***		
Preconditions	• User entered in the RAWFIE Portal		
Related Requirements	PT-EXA-T-001, PT-EXA-T-002, PT-EXA-T-003, PT-EXA-T-004, PT-EXA-T-005, PT-EXA-T-008, PT-EXA-T-009, PT-EXA-T-010, PT-EXA-T-011, PT-EXA-T-012, PT-EXA-T-013, PT-EXA-T-015		
Tools Used	• RAWFIE Web Portal • RAWFIE Textual - Visual Editors • RAWFIE Launching Tool		

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Access to the authoring tool through the RAWFIE Web Portal	Redirection to the editor interface		
2	Select an experiment	A drop-down list of the available experiments is appeared and the experimenter has the opportunity to select one		
3	Start the experiment execution	The launching service is informed with the experiment ID and the execution starts		

Table 23: Verification test of the experiment Launchings

Test ID**: EAT07**	Conducted by**:**	Date**:**	Test Category**: Verification Tests (front end tier – middle tier)**
Hardware Configuration	-		
Software Configuration	•		
Test Name:	***Launch (scheduled) experiments***		
Preconditions	• User entered in the RAWFIE Portal		
Related Requirements	PT-EXA-T-001, PT-EXA-T-002, PT-EXA-T-003, PT-EXA-T-004, PT-EXA-T-005, PT-EXA-T-008, PT-EXA-T-009, PT-EXA-T-010, PT-EXA-T-011, PT-EXA-T-012, PT-EXA-T-013, PT-EXA-T-015		
Tools Used	• RAWFIE Web Portal • RAWFIE Textual - Visual Editors • RAWFIE Launching Tool		

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Access to the authoring tool through the	Redirection to the		

	RAWFIE Web Portal	editor interface		
2	Select the scheduled launching tool	A drop-down list of the available experiments is appeared and the experimenter has the opportunity to select one		
3	Define the experiment execution	The launching service is informed with the experiment ID and the execution is planned		

5.1.1.6 Experiment Monitoring Tool

Table 24: Verification test of the Visualization of experiment status

Test ID**: EMT01**	Conducted by**:**	Date**:**	Test Category**: Verification Tests (front end tier)**
Hardware Configuration	-		
Software Configuration	-		
Test Name:	***Visualisation of experiment status***		
Preconditions	• Experiments running knowledge about the experiments state needed on user side (to check results)		
Related Requirements	PT-EXM-T-002, PT-EXM-T-002		
Tools Used	•		

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	user opens Experiment Monitoring Tool in the Web Portal	Experiment Monitoring Tool displays a view with all experiments of the current user (ordered by date descending). The list also contains a sort summary of the experiments state		
2	user selects an experiment	Experiment Monitoring Tool displays all experiment details (date / timespan; related testbed; list of used UxVs; execution state ; link to the used EDL)		
3	User clicks to start the visualisation	The Visualisation Tool is opened for the experiment		

Table 25: Verification test of the canceling of experiments

Test ID: **EMT02**		Conducted by:	Date:	Test Category: **Verification Tests (front end tier)**
Hardware Configuration		-		
Software Configuration		-		
Test Name:		***Cancel of experiment***		
Preconditions		• Experiments running		
Related Requirements		PT-EXP-C-001, PT-LAU-S-010, PT-LAU-S-012, TB-MAN-005		
Tools Used		•		
Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	user opens Experiment Monitoring Tool in the Web Portal	Experiment Monitoring Tool displays a view with all experiments of the current user		
2	user selects an experiment	Experiment Monitoring Tool displays all experiment details and the option to cancel it		
3	User clicks the cancel button	Cancellation request is sent. User is informed about the ongoing cancellation		
4	User watches further the experiment status	Experiment status is set to "cancelled" when the cancellation is complete		

5.1.1.7 System Monitoring Tool

Table 26: Verification test of the Visualisation of system and UxV health status

Test ID: **SMT01**		Conducted by:	Date:	Test Category: **Verification Tests (front end tier)**
Hardware Configuration				
Software Configuration				
Test Name:		***Visualisation of system and UxV health status***		
Preconditions		• connection to the System Monitoring Service • administrative knowledge about the system state needed on user side (to check results)		
Related Requirements		PT-SYM-T-001, PT-SYM-T-002, PT-SYM-T-004, PT-SYM-T-005		
Tools Used		•		
Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	user opens System Monitoring Tool in the Web Portal	the System Monitoring Tool displays a view with severity indication and textual information of middleware components, testbeds components, UxVs components		
2	User sets some sorting and filter options to see the services he is interested in.	Monitoring Tool filters and sorts the data accordingly		
3	User watches the web site for a while	Displayed data is updated automatically		

(See also tests for System Monitoring Service)

5.1.1.8 Visualisation Tool

Table 18: Verification test of the User request handling

Test ID: **VIS01**	Conducted by:	Date:	Test Category: **Verification Tests (front end)**
Hardware Configuration			
Software Configuration			
Test Name:	***User request handling***		
Preconditions	• Requires visualization tool to be functioning & accessible. • Requires visualization engine to be functioning & accessible.		
Related Requirements	PT-VIS-E-001, PT-VIS-E-003, PT-EXP-C-002, PT-EXP-C-003, PT-EXP-C-004, PT-EXP-C-005, PT-EXP-C-006, PT-EXP-C-007, PT-EXP-C-008, PT-EXP-C-009, PT-VIS-T-001, PT-VIS-T-002, PT-VIS-T-003, PT-VIS-T-004, PT-VIS-T-005, PT-VIS-T-006, PT-VIS-T-007		
Tools Used	•		

Step	Action	Expected Result	Status	Remarks
1	User starts one of the experiments from the experiment list	The visualization tool forwards it to the visualization engine		
2	the visualisation engine starts the visualisation of the experiment	The map is loaded and the experiment is visualized on the user screen		

Table 19: Verification test of the Geospatial data handling

Test ID: **VIS02**	Conducted by:	Date:	Test Category: **Verification Tests (front end)**
Hardware Configuration			
Software Configuration			
Test Name:	***Geospatial data handling***		
Preconditions	• Requires visualization tool to be functioning & accessible. • Requires visualization engine to be functioning & accessible. • Requires message bus to be functioning & accessible.		
Related Requirements	PT-VIS-E-001, PT-VIS-E-003, PT-EXP-C-002, PT-EXP-C-003, PT-EXP-C-004, PT-EXP-C-005, PT-EXP-C-006, PT-EXP-C-007, PT-EXP-C-008, PT-EXP-C-009, PT-VIS-T-001, PT-VIS-T-002, PT-VIS-T-003, PT-VIS-T-004, PT-VIS-T-005, PT-VIS-T-006, PT-VIS-T-007		
Tools Used	•		

Step	Action	Expected Result	Status	Remarks
1	The user starts an already finished experiment	Request is forwarded to the VE		
2	The VE sends the data for the experiment in the correct format to the VT	VT presents the data for the experiment in layers to the user		

Table 20: Verification test of the Geospatial data modification

Test ID: **VIS03**	Conducted by:	Date:	Test Category: **Verification Tests (front end)**
Hardware Configuration			
Software Configuration			
Test Name:	***Geospatial data modification***		
Preconditions	• Requires visualization tool to be functioning & accessible. • Requires visualization engine to be functioning & accessible. • Requires message bus to be functioning & accessible.		
Related Requirements	PT-VIS-E-001, PT-VIS-E-003, PT-EXP-C-002, PT-EXP-C-003, PT-EXP-C-004, PT-EXP-C-005, PT-EXP-C-006, PT-EXP-C-007, PT-EXP-C-008, PT-EXP-C-009, PT-VIS-T-001, PT-VIS-T-002, PT-VIS-T-003, PT-VIS-T-004, PT-VIS-T-005, PT-VIS-T-006, PT-VIS-T-007		
Tools Used	• Browser		

Step	Action	Expected Result	Status	Remarks
1	User starts an already running experiment	Data is visualized properly to the user		
2	User turns off a layer with data	VT hides the data from this layer from the user		
3	User turns on a layer with data from the experiment	VT requests this data from the VE, receives it and shows it to the user in the proper layer		

Table 21: Verification test of the Experiment Controller communication

Test ID: **VIS04**	Conducted by:	Date:	Test Category: **Verification Tests (front end)**
Hardware Configuration			
Software Configuration			
Test Name:	***Experiment Controller communication***		
Preconditions	• Requires experiment controller to be functioning & accessible. • Requires visualization engine to be functioning & accessible.		
Related Requirements	PT-VIS-E-001, PT-VIS-E-003, PT-EXP-C-002, PT-EXP-C-003, PT-EXP-C-004, PT-EXP-C-005, PT-EXP-C-006, PT-EXP-C-007, PT-EXP-C-008, PT-EXP-C-009, PT-VIS-T-001, PT-VIS-T-002, PT-VIS-T-007		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	The user starts an experiment	The message is forwarded to the visualisation engine		
2	Receive a message that the experiment has started from the Experiment Controller	The visualization tool starts the experiment and loads the map		
3	Receive a message that the experiment has stopped from the Experiment Controller	The VT stops the experiment and the user gets a notification about that event		

Table 22: Verification test of the Visualization Tool Interaction

Test ID: **VIS05**	Conducted by:	Date:	Test Category: **Verification Tests (front end)**
Hardware Configuration			
Software Configuration			
Test Name:	***Visualization Tool Interaction***		
Preconditions	• Requires visualization tool to be functioning & accessible. • Requires visualization engine to be functioning & accessible.		
Related Requirements	PT-VIS-E-001, PT-VIS-E-003, PT-VIS-T-001, PT-VIS-T-002, PT-VIS-T-003, PT-VIS-T-004, PT-VIS-T-005, PT-VIS-T-006, PT-VIS-T-007		
Tools Used	•		

Step	Action	Expected Result	Status	Remarks
1	Enable different features of the visualization tool (e.g. show/hide speed web widget)	The user sees the updated plot (show speed web widget)		
2	Disable a feature (e.g. speed web widget)	The widget is removed from the screen		

Table 23: Verification test of the Indoor maps

Test ID: **VIS06**	Conducted by:	Date:	Test Category: **Verification Tests (front end)**
Hardware Configuration			
Software Configuration			
Test Name:	***Indoor maps interaction***		
Preconditions	• Requires visualization tool to be functioning & accessible. • Requires visualization engine to be functioning & accessible. • Requires Experiment controller to be functioning & accessible. • Requires an indoor map to be loaded in the GeoServer		
Related Requirements	PT-VIS-E-001, PT-VIS-E-003, PT-VIS-T-001, PT-VIS-T-002, PT-VIS-T-003, PT-VIS-T-004, PT-VIS-T-005, PT-VIS-T-006, PT-VIS-T-007		
Tools Used	•		

Step	Action	Expected Result	Status	Remarks
1	Start an experiment with indoor maps	An experiment is loaded, the indoor map is loaded from the GeoServer and is shown on the screen		
2	A UxV moves	The data from the VE is received and plotted on the screen		

5.1.1.9 Data Analysis Tool

Table 22: Verification test of starting a data analysis task on the DAE via the DAT

Test ID: **DAT01**	Conducted by:	Date:	Test Category: **Verification Tests (front end)**
Hardware Configuration			
Software Configuration			
Test Name:	***Start a data analysis task on the DAE via the DAT***		
Preconditions	• Requires the message bus to be functioning and accessible • Requires the schema registry to be functioning and accessible • Requires the Zeppelin notebook interface of the DAT to be functioning and accessible • Requires result repository to be functioning and accessible		
Related Requirements	PT-DAA-T-001, PT-DAA-T-003, PT-DAA-T-005		
Tools Used	•		

Step	Action	Expected Result	Status	Remarks
1	Authorized user logs into the web portal and clicks on the schema registry tab of the Data Analysis Tool GUI embedded into the web portal	Login successful, successfully reaches the schema registry GUI tab of the Data Analysis Tool GUI embedded into the web portal		
2	User selects the topics and fields corresponding to streaming data currently present on the message bus to perform an analysis task on, then clicks on the "create Zeppelin notebook" button once the desired elements have been selected.	A Zeppelin notebook has been successfully created, and is already populated with the topics and fields selected by the user.		
3	User designs an analysis task in the notebook relying on Spark and starts it within the notebook.	The job has been successfully started. The process should be visible through the spark master UI of the Data Analysis Tool. Additionally, if the streaming results are published to the time series database (result repository), the results should be visible on the Grafana dashboard (part the Data Analysis Tool).		

Table 22: Verification test of retrieving data from the message bus

Test ID: **DAT02**	Conducted by:	Date:	Test Category: **Verification Tests (front end)**
Hardware Configuration			
Software Configuration			
Test Name:	***Retrieve data from the message bus***		
Preconditions	• Requires the message bus to be functioning and accessible • Requires the schema registry to be functioning and accessible • Requires result repository to be functioning and accessible		
Related Requirements	PT-DAA-T-003		
Tools Used	•		

Step	Action	Expected Result	Status	Remarks
1	Authorized user logs into the web portal and clicks on the schema registry tab of the Data Analysis Tool GUI embedded into the web portal	Login successful, successfully reaches the schema registry GUI tab of the Data Analysis Tool GUI embedded into the web portal		
2	User selects the topics and fields corresponding to streaming data currently present on the message bus to perform an analysis task on, then clicks on the "create Zeppelin notebook" button once the desired elements have been selected.	A Zeppelin notebook has been successfully created, and is already populated with the topics and fields selected by the user.		
3	User designs an streaming analysis task in the notebook to be performed on data from the message bus and starts it within the notebook.	The data is successfully retrieved and the analysis task therefore can process it and display the results on the Grafana dashboard.		

Table 22: Verification test of ending a running job

Test ID: **DAT03**	Conducted by:	Date:	Test Category: **Verification Tests (front end)**
Hardware Configuration			
Software Configuration			
Test Name:	***End a running job***		
Preconditions	• Requires the message bus to be functioning and accessible • Requires the schema registry to be functioning and accessible • Requires the Zeppelin notebook interface of the DAT to be functioning and accessible • Requires result repository to be functioning and accessible		
Related Requirements	PT-DAA-T-004, PT-DAA-T-003, PT-DAA-T-005		
Tools Used	•		

Step	Action	Expected Result	Status	Remarks
1	Authorized user logs into the web portal and clicks on the schema registry tab of the Data Analysis Tool GUI embedded into the web portal	Login successful, successfully reaches the schema registry GUI tab of the Data Analysis Tool GUI embedded into the web portal		
2	User selects the topics and fields corresponding to streaming data currently present on the message bus to perform an analysis task on, then clicks on the "create Zeppelin notebook" button once the desired elements have been selected.	A Zeppelin notebook has been successfully created, and is already populated with the topics and fields selected by the user.		
3	User designs an streaming analysis task in the notebook to be performed on data from the message bus and starts it within the notebook.	The data is successfully retrieved and the analysis task therefore can process it and display the results on the Grafana dashboard.		
4	User stops the running job within the Zeppelin notebook	The job has been successfully stopped (results stopped being sent to the dashboard)		

Table 22: Verification test of accessing past results

Test ID: **DAT04**	Conducted by:	Date:	Test Category: **Verification Tests (front end)**
Hardware Configuration			
Software Configuration			
Test Name:	***Access past results***		
Preconditions	• Requires the message bus to be functioning and accessible • Requires the schema registry to be functioning and accessible • Requires the Zeppelin notebook interface of the DAT to be functioning and accessible • Requires result repository to be functioning and accessible		
Related Requirements	PT-DAA-T-002, PT-DAA-T-005		
Tools Used	•		

Step	Action	Expected Result	Status	Remarks
1	Authorized user logs into the web portal and clicks on the results repository tab of the Data Analysis Tool GUI embedded into the web portal	Login successful, successfully reaches results repository GUI (Grafana dashboard) tab of the Data Analysis Tool GUI embedded into the web portal		
2	User uses the Grafana dashboard interface to display results of previous time steps.	The dashboard allows such browsing and displays the past results of the associated experiment (associated to a metric) correctly		
3	User accesses the data persistently stored on Grafana's underlying time series database vias CLI.	The data is correctly accessed.		

5.1.2 Middle Tier

This section presents the test of the Middle tier services and communication components.

5.1.2.1 Testbed Directory Service

In the following tables (Table 27 to Table 30), an updated version of the verification tests for checking Testbed Directory Service features is reported. This version of the verification tests presents some additions and modifications from the previous version of D4.6, and although the same tables have been already presented in the report of components' verification tests of D6.3, here they are reported again, since they reflect the latest design and development of the component itself.

Table 27: Verification test of the resources information retrieval and resources search

Test ID: **TD01**	Conducted by:	Date:	Test Category: **Verification Tests (Middle Tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Retrieve resources information and search for specific resources***		
Preconditions	Access to the PostgreSQL server must be granted for the Testbed Directory Service When preparing the test, the test executor should know either the ID of the resource he is looking for, or other parameters according to the criteria he/she is using for selecting specific resources		
Related Requirements	PT-DIR-S-003, PT-DIR-S-004, PT-DIR-S-006		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1.a	The input JSON request is prepared, specifying a testbed identifier (for the ***/request/getResources()*** REST interface) or a resource identifier (for the ***/request/searchResource()*** REST interface), or nothing in case the ***/request/getAllResources()*** REST interface is used	No error occurred. The Testbed Directory Service gives back a JSON response message, containing details about a specific resource, the resources belonging to the specified testbed, or all resources in case the *getAllResources()* interface is used		
2.a	The ***/request/getAllResources()*** **(without parameters) or** ***request/searchResource()*** or ***request/getResources()*** (providing the prepared JSON request in input) REST interfaces can be called from the SOAP UI Client Tool.			
1.b	The ***/request/resource/identifier/{id}*** REST interface is called (from the browser or using a tool like SOAP UI), specifying the id of a specific resource	No error occurred. The Testbed Directory Service gives back a JSON response message, containing detailed information about the resource (or the list of resources) matching the search criteria		
2.b	The ***/request/resource/name/{name}*** REST interface is called (from the browser or using a tool like SOAP UI), specifying the name of a specific resource			
3.b	The ***/request/resources?param1=value1&param2=value2&param3=value3&param4=value4*** REST interface is called (from the browser or using a tool like SOAP UI), with one or more query parameters according to the selected search criteria, that is, a combination of one or more of the following 4 possible search parameters: • *resource_status* • *resource_status_message* • *resource_type* • *health*			
4.b	The ***/request/resources/testbedid/{id}*** REST interface is called (from the browser or using a tool like SOAP UI), specifying the id of the Testbed we would like to get resources from			

Table 28: Verification tests for adding or removing a testbed facility

Test ID: **TD02**	Conducted by:	Date:	Test Category: **Verification Tests (Middle Tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Add / delete a testbed facility to RAWFIE***		
Preconditions	Access to the PostgreSQL server must be granted for the Testbed Directory Service When preparing the test for the testbed registration case, the test executor should know the information about the testbed to be inserted. In case of a testbed deletion, the testbed id must be known in advance		
Related Requirements	PT-DIR-S-005		
Tools Used	SOAP UI		

Step	Action	Expected Result	Status	Remarks
1.a	The input JSON request is prepared, with the information about the new testbed to be added	No error occurred. And the information about the new testbed is from now on available in the Master Data Repository, as it can be verified by using the ***getAllTestbeds()*** or other REST interfaces for Testbeds searches (see **TD04**)		
2.a	***The /request/createTestbed()*** REST interface is called from the SOAP UI Client Tool, specifying the testbed information in the input JSON request			
1.b	The input JSON message request is prepared, with the unique id of the testbed facility to be deleted	No error occurred. And the information about the deleted testbed (and related resources) is not available anymore in the Master Data Repository, as it can be verified by using the ***getAllTestbeds()*** or other REST interfaces (see **TD04** in the following)		
2.b	The ***/request/deleteTestbed()*** REST interface is called from the SOAP UI Client Tool, specifying the information about the testbed to be deleted in the provided input JSON request			

Table 29: Verification test of the registration or removal of a new UxV node into a testbed facility

Test ID: **TD03**	Conducted by:	Date:	Test Category: **Verification Tests (Middle Tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Register / delete an UxV node into a testbed facility***		
Preconditions	Access to the PostgreSQL server must be granted for the Testbed Directory Service. When preparing the test, the test executor should know either the ID of the testbed		
Related Requirements	PT-DIR-S-007		
Tools Used	SOAP UI		

Step	Action	Expected Result	Status	Remarks
1.a	The input JSON message request is prepared, with all information about the new resource to be added (and the unique id of the testbed facility it belongs to)	No error occurred. And the information about the new resource (UxV node) is from now on available in the Master Data Repository, as it can be verified by using the ***getAllResources()*** or other REST API for Resources searches (see previous tests **TD01**)		
2.a	The ***/request/createResource()*** REST interface is called from the SOAP UI Client Tool, specifying the information about the resource to be added in the provided input JSON request			
1.b	The input JSON message request is prepared, with the unique id of the resource to be deleted and of the testbed facility it belongs to	No error occurred. And the resource (UxV node) is not available anymore in the Master Data Repository, as it can be verified by using the ***getAllResources()*** or other REST API (see previous tests **TD01**)		
2.b	The ***/request/deleteResource()*** REST interface is called from the SOAP UI Client Tool, specifying the information about the resource to be deleted in the provided input JSON request			

Table 30: Verification test of the testbeds information retrieval and testbeds search

Test ID: **TD04**	Conducted by:	Date:	Test Category: **Verification Tests (Middle Tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Retrieve testbed information and search for specific testbeds***		
Preconditions	Access to the PostgreSQL server must be granted for the Testbed Directory Service When preparing the test, the test executor should know the ID of the testbed he is looking for, or the value of other parameters in case he/she is looking for resources based on different search criteria		
Related Requirements	PT-DIR-S-001, PT-DIR-S-002, PT-DIR-S-006		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1.a	The /***request/getAllTestbeds()*** REST interface is called from the SOAP UI Client Tool, without any specific testbed information (null JSON input request)	No error occurred. The Testbed Directory Service gives back a JSON response message, containing details about all registered testbeds and all resources belonging to each of them		
1.b	The input JSON request is prepared, specifying a testbed identifier (for the ***request/searchTestbed()*** REST interface)	No error occurred. The Testbed Directory Service gives back a JSON response message, containing details about the requested testbed		
2.b	The ***/request/searchTestbed()*** REST interface is called from the SOAP UI Client Tool, using the abovementioned JSON as input message request			
1.c	The /***request/testbed/identifier/{id}*** REST interface is called from the Browser, specifying the id of a specific testbed	No error occurred. The Testbed Directory Service gives back a JSON response message, containing		
2.c	The /***request/testbed/name/{name}*** REST interface is called, specifying the name of a specific testbed			

3.c	The ***/request/testbeds?param1=value1&param2=value2&param3=value3*** REST interface is called, with one or more query parameters according to the selected search criteria, that is, a combination of one or more of the following 3 possible search parameters: • *health* • *testbedstatusmessage* • *srid*	details about the available testbeds conforming to the search criteria		
4.c	The ***/request/testbed/uav*** REST interface is called, looking for all testbeds supporting UAV resources			
5.c	The ***/request/testbed/ugv*** REST interface is called, looking for all testbeds supporting UGV resources			
6.c	The ***/request/testbed/usv*** REST interface is called, looking for all testbeds supporting USV resources			
7.c	The ***/request/testbed/auv*** REST interface is called, looking for all testbeds supporting AUV resources			

5.1.2.2 EDL Compiler and Validator

Table 31: Verification test of the in-Textual Editor Experiments definition

Test ID**: EAT01**	Conducted by**:**	Date**:**	Test Category**: Verification Tests (front end tier – middle tier)**
Hardware Configuration	-		
Software Configuration			
Test Name:	***Define Experiments in the Textual Editor***		
Preconditions	• User entered in the RAWFIE Portal		
Related Requirements	PT-EXA-T-001, PT-EXA-T-002, PT-EXA-T-003, PT-EXA-T-004, PT-EXA-T-005, PT-EXA-T-008, PT-EXA-T-009, PT-EXA-T-010, PT-EXA-T-011, PT-EXA-T-012, PT-EXA-T-013, PT-EXA-T-015		
Tools Used	• RAWFIE Web Portal • RAWFIE Textual Editor		

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Access to the Textual Editor through the RAWFIE Web Portal	Redirection to the Textual Editor interface		
2	Write an experiment	Experiment is presented in the editor		
3	Utilize code completion, content assist and compilation	The editor responds with specific drop down lists, messages, etc.		
4	Define erroneous commands in the experiment workflow	The editor responds with error messages and indication for correcting the error		
5	Save the experiment	The experiment is stored in the database and specific files are produced to be adopted by the remaining RAWFIE components		

Table 32: Verification test of the Textual Editor Experiments Update

Test ID: **EAT02**	Conducted by:	Date:	Test Category: **Verification Tests (front end tier – middle tier)**
Hardware Configuration	-		
Software Configuration			
Test Name:	***Update Experiments in the Textual Editor***		
Preconditions	• User entered in the RAWFIE Portal		
Related Requirements	PT-EXA-T-001, PT-EXA-T-002, PT-EXA-T-003, PT-EXA-T-004, PT-EXA-T-005, PT-EXA-T-008, PT-EXA-T-009, PT-EXA-T-010, PT-EXA-T-011, PT-EXA-T-012, PT-EXA-T-013, PT-EXA-T-015		
Tools Used	• RAWFIE Web Portal • RAWFIE Textual Editor		

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Access to the Textual Editor through the RAWFIE Web Portal	Redirection to the Textual Editor interface		
2	Open an already defined experiment	Experiment is presented in the editor		
3	Makes changes in the experiment workflow	The experiment is updated		
4	Save the experiment	The experiment is stored in the database and specific files are produced to be adopted by the remaining RAWFIE components		

Table 33: Verification test of the in-Visual Editor Experiments Define

Test ID: **EAT03**	Conducted by:	Date:	Test Category: **Verification Tests (front end tier – middle tier)**
Hardware Configuration	-		
Software Configuration	•		
Test Name:	***Define Experiments in the Visual Editor***		
Preconditions	• User entered in the RAWFIE Portal		
Related Requirements	PT-EXA-T-001, PT-EXA-T-002, PT-EXA-T-003, PT-EXA-T-004, PT-EXA-T-005, PT-EXA-T-008, PT-EXA-T-009, PT-EXA-T-010, PT-EXA-T-011, PT-EXA-T-012, PT-EXA-T-013, PT-EXA-T-015		
Tools Used	• RAWFIE Web Portal • RAWFIE Visual Editor		

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Access to the Visual Editor through the RAWFIE Web Portal	Redirection to the Visual Editor interface		
2	Access the available toolbar	Specific windows are presented		
3	Create an experiment by utilizing the available tools	The experimenter can define waypoints and experiment information by clicking and designing in the visual editor		
4	Define erroneous commands	The authoring tool responds with error messages and indication for correcting the error		
5	Save the experiment	The experiment is stored in the database and specific files are produced to be adopted by the remaining RAWFIE components		

Table 34: Verification test of the in-Visual Editor Experiments Update

Test ID: **EAT04**	Conducted by:	Date:	Test Category: **Verification Tests (front end tier – middle tier)**
Hardware Configuration	-		
Software Configuration			
Test Name:	***Update Experiments in the Visual Editor***		
Preconditions	• User entered in the RAWFIE Portal		
Related Requirements	PT-EXA-T-001, PT-EXA-T-002, PT-EXA-T-003, PT-EXA-T-004, PT-EXA-T-005, PT-EXA-T-008, PT-EXA-T-009, PT-EXA-T-010, PT-EXA-T-011, PT-EXA-T-012, PT-EXA-T-013, PT-EXA-T-015		
Tools Used	• RAWFIE Web Portal • RAWFIE Visual Editor		

Step	Action	Expected Result	Status	Remarks
1	Access to the Visual Editor through the RAWFIE Web Portal	Redirection to the Visual Editor interface		
2	Open an already defined experiment	Experiment is presented in the editor		
3	Makes changes in the experiment workflow	The experiment is updated		
4	Save the experiment	The experiment is stored in the database and specific files are produced to be adopted by the remaining RAWFIE components		• RAWFIE Web Portal RAWFIE Textual Editor

Table 35: Verification test of the Editor switching

Test ID: **EAT05**	Conducted by:	Date:	Test Category: **Verification Tests (front end tier – middle tier)**
Hardware Configuration	-		
Software Configuration	•		
Test Name:	***Switch between the Editors***		
Preconditions	• User entered in the RAWFIE Portal		
Related Requirements	PT-EXA-T-001, PT-EXA-T-002, PT-EXA-T-003, PT-EXA-T-004, PT-EXA-T-005, PT-EXA-T-008, PT-EXA-T-009, PT-EXA-T-010, PT-EXA-T-011, PT-EXA-T-012, PT-EXA-T-013, PT-EXA-T-015		
Tools Used	• RAWFIE Web Portal • RAWFIE Textual Editor • RAWFIE Visual Editor		

Step	Action	Expected Result	Status	Remarks
1	Access to the editors through the RAWFIE Web Portal	Redirection to the editor interface		
2	Create an experiment	Experiment is presented in the editor interface		
3	Switch to the alternative editor and make changes	The experiment is updated		
4	Save the experiment	The experiment is stored in the database and specific files are		

		produced to be adopted by the remaining RAWFIE components		

Table 36: Verification test of the experiment Launchings

Test ID**: EAT06**	Conducted by**:**	Date**:**	Test Category**: Verification Tests (front end tier – middle tier)**
Hardware Configuration	-		
Software Configuration	•		
Test Name:	***Launch experiments***		
Preconditions	• User entered in the RAWFIE Portal		
Related Requirements	PT-EXA-T-001, PT-EXA-T-002, PT-EXA-T-003, PT-EXA-T-004, PT-EXA-T-005, PT-EXA-T-008, PT-EXA-T-009, PT-EXA-T-010, PT-EXA-T-011, PT-EXA-T-012, PT-EXA-T-013, PT-EXA-T-015		
Tools Used	• RAWFIE Web Portal • RAWFIE Textual - Visual Editors • RAWFIE Launching Tool		

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Access to the authoring tool through the RAWFIE Web Portal	Redirection to the editor interface		
2	Select an experiment	A drop-down list of the available experiments is appeared and the experimenter has the opportunity to select one		
3	Start the experiment execution	The launching service is informed with the experiment ID and the execution starts		

Table 37: Verification test of the experiment Launchings

Test ID**: EAT07**	Conducted by**:**	Date**:**	Test Category**: Verification Tests (front end tier – middle tier)**
Hardware Configuration	-		
Software Configuration	•		
Test Name:	***Launch (scheduled) experiments***		
Preconditions	• User entered in the RAWFIE Portal		
Related Requirements	PT-EXA-T-001, PT-EXA-T-002, PT-EXA-T-003, PT-EXA-T-004, PT-EXA-T-005, PT-EXA-T-008, PT-EXA-T-009, PT-EXA-T-010, PT-EXA-T-011, PT-EXA-T-012, PT-EXA-T-013, PT-EXA-T-015		
Tools Used	• RAWFIE Web Portal • RAWFIE Textual - Visual Editors • RAWFIE Launching Tool		

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Access to the authoring tool through the RAWFIE Web Portal	Redirection to the editor interface		

2	Select the scheduled launching tool	A drop-down list of the available experiments is appeared and the experimenter has the opportunity to select one		
3	Define the experiment execution	The launching service is informed with the experiment ID and the execution is planned		

5.1.2.3 Users & Rights Service

Table 38: Verification test of the Users & Rights Service login checking

Test ID**: URS01**	Conducted by**:**	Date**:**	Test Category**: Verification Tests (middle tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Login checking***		
Preconditions	• Valid user name and password known		
Related Requirements	PT-USR-S-001		
Tools Used	•		

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	invalid user name and password sent to the Users & Rights Service	Users & Rights Service returns failure		
2	valid user name and password sent to the Users & Rights Service	Users & Rights Service returns OK		

Table 39: Verification test of the user rights checks

Test ID**: URS02**	Conducted by**:**	Date**:**	Test Category**: Verification Tests (middle tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Roles/rights checking***		
Preconditions	• Valid user rights known		
Related Requirements	PT-USR-S-002		
Tools Used	•		

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	user ID and available required rights sent to the Users & Rights Service	Users & Rights Service return true		
2	user ID and not available required rights sent to the Users & Rights Service	Users & Rights Service return false		

Table 40: Verification test for adding and editing user data

Test ID: **URS03**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Adding and editing user data***		
Preconditions	• New user does not exist		
Related Requirements	PT-USR-S-002		
Tools Used	•		

Step	Action	Expected Result	Status	Remarks
1	New user data (personal data and roles) sent to the Users & Rights Service	Users & Rights Service creates the new user and returns true		
2	Request user data of new user	Users & Rights Service return the data. It should be equal to the data of step 1		
3	Edited user data (personal data and roles) sent to the Users & Rights Service	Users & Rights Service saves the user data and returns true		
4	Request user data of the user	Users & Rights Service return the data. It should be equal to the data of step 3		

5.1.2.4 Booking Service

The Booking Service is tightly coupled with the Booking Tool component. Therefore, the verification tests described for the Booking Tool should also be considered during Booking Service functionality verification activities. Verification tests of the component focus around testing and ensuring the correctness of each provided method.

The Booking Service requirements not addressed by the tests specified below are

- PT-BOO-S-003 (concerns experiment level booking on a subset of resources of the user level booking and is outside the scope of the booking service – performed by the Authoring Tool prior to manual or scheduled launching)
- PT-BOO-S-012 (ensured by the way booking process is implemented in steps needing always testbed approval before being accepted)

All Test Procedures BS01, BS02, BS03, BS04, BS05, BS06, BS07, BS08 remain unchanged compared to what was defined in the previous version of the deliverable (D4.6).

Table 41: Verification test of Booking Service add reservation functionality

Test ID: **BS01**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	-		
Software Configuration	-		
Test Name:	***Booking Service add reservation functionality***		
Preconditions	• Master DB is prepopulated with reservations of different status and timeslots (involved tables are: Reservation, Resource Reservation) • User initiating the call is a valid experimenter		
Related Requirements	PT-BOO-S-001 (user level booking) PT-BOO-S-002 PT-BOO-S-004 PT-BOO-S-005 PT-BOO-S-007 PT-BOO-S-012		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Call addReservation() providing a datetime interval that has passed	response should be returned with a proper failure message		
2	Call addReservation() providing a datetime interval in the future (NO conflict in requested resources with existing reservation at the same time)	Appropriate MasterDB tables are updated (new reservation in status=PENDING)		
		If email sending is enabled then email is send to both the creator and the testbed operator of the reserved resources		
		The returned response contains the newly created reservationId and the reservation status		
3	Call addReservation() providing a datetime interval in the future conflict in requested resources with existing reservation at the same time)	response should be returned with a proper failure message		

Table 42: Verification test of Booking Service edit reservation functionality

Test ID: **BS02**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	-		
Software Configuration	-		
Test Name:	***Booking Service add reservation functionality***		
Preconditions	• Master DB is prepopulated with reservations of different status and timeslots (involved tables are: Reservation, Resource_Reservation) • User initiating the call is a valid experimenter		
Related Requirements	PT-BOO-S-002 PT-BOO-S-005 PT-BOO-S-007 PT-BOO-S-013		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Call editReservation() providing appropriate ReservationData which should include the reservationId (the call should include credentials about the user initiating it)	If provided user credentials do not match with the ones of the reservation owner then a proper failure message is returned		
		If existing reservation status!= PENDING then no update should be possible and a proper failure message is returned		
		If time related changes refer to an interval in the past then a proper failure message is returned		
	(If status= PENDING & user credential match)	If overlaps with existing reservation are introduced and resources conflicts are detected then a proper failure message is returned		
	(If status= PENDING & user credential match)	If no resources conflicts are detected the changes are accepted and the corresponding DB tables updated		
2	Repeat step 1 with different kind of changes related to timeslots and resource selection	Ensure that expected results are respected as described in step 1		

Table 43: Verification test of Booking Service approve reservation functionality

Test ID: **BS03**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	-		
Software Configuration	-		
Test Name:	***Booking Service approve reservation functionality***		
Preconditions	• Master DB is prepopulated with reservations of different status and timeslots (involved tables are: Reservation, Resource_Reservation)		
Related Requirements	PT-BOO-S-002 PT-BOO-S-005 PT-BOO-S-007 PT-BOO-S-013 PT-NF-002		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Call approveReservation() (the call should include credentials about the user initiating it)	If provided credentials do not match with an authorized platform user then a proper failure message is returned		
		If provided credentials do not refer to an authorized platform user with role=TESTBED_OP then a proper failure message is returned		
		If reservationId refers to a reservation with status !=PENDING then a proper failure message is returned		
		If reservationId refers to a past reservation then then a proper failure message is returned		
		If conflicts are detected with any other APPROVED reservation then then a proper failure message is returned		
2	(If status= PENDING & caller=TESTBED_OP & no conflicts detected	Status change is accepted and corresponding DB tables updated		
		An email is send to the owner of the reservation		
		A ReservationStatusMsg is send to Message bus		

Table 44: Verification test of Booking Service reject reservation functionality

Test ID: **BS04**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	-		
Software Configuration	-		
Test Name:	***Booking Service reject reservation functionality***		
Preconditions	• Master DB is prepopulated with reservations of different status and timeslots (involved tables are: Reservation, Resource_Reservation)		
Related Requirements	PT-BOO-S-002 PT-BOO-S-005 PT-BOO-S-007 PT-BOO-S-013 PT-NF-002		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Call approveReservation() (the call should include credentials about the user initiating it)	If provided credentials do not match with an authorized platform user then a proper failure message is returned		
		If provided credentials do not refer to an authorized platform user with role=TESTBED_OP then a proper failure message is returned		
		If reservationId refers to a reservation with status !=PENDING or APPROVED then a proper failure message is returned		
		If reservationId refers to a past reservation then then a proper failure message is returned		
2	(If status= PENDING & caller=TESTBED_OP	Status change is accepted and corresponding DB tables updated		
		An email is send to the owner of the reservation		
		A ReservationStatusMsg is send to Message bus		

Table 45: Verification test of Booking Service delete reservation functionality

Test ID: **BS05**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	-		
Software Configuration	-		
Test Name:	***Booking Service delete reservation functionality***		
Preconditions	• Master DB is prepopulated with reservations of different status and timeslots (involved tables are: Reservation, Resource_Reservation)		
Related Requirements	PT-BOO-S-002 PT-BOO-S-005 PT-BOO-S-007 PT-NF-002		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Call deleteReservation() (the call should include credentials about the user initiating it)	If provided credentials do not match with an authorized platform user then a proper failure message is returned		
		If reservationId refers to a past reservation then a proper failure message is returned		
		If reservationId refers to a reservation with resources involved in a currently running experiment a proper failure message is returned		
		If none of the above then status change to CANCELLED		

Table 46: Verification test of Booking Service retrieve reservation(s) functionality

Test ID: **BS06**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	-		
Software Configuration	-		
Test Name:	***Booking Service retrieve reservation(s) functionality***		
Preconditions	• Master DB is prepopulated with reservations of different status and timeslots (involved tables are: Reservation, Resource_Reservation)		
Related Requirements	PT-BOO-S-002 PT-BOO-S-008		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Call getReservation() providing a reservationId	Inspect response and ensure data is inline with the information stored in the MasterDB		
2	Call getReservations() providing appropriate search criteria (time, user etc.)	Inspect response and ensure data is in line with the information stored in the MasterDB		

Table 47: Verification test of Booking Service check for conflicts functionality

Test ID: **BS07**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	-		
Software Configuration	-		
Test Name:	***Booking Service check for conflicts functionality***		
Preconditions	• Master DB is prepopulated with reservations of different status and timeslots (involved tables are: Reservation, Resource_Reservation)		
Related Requirements	PT-BOO-S-002 PT-BOO-S-006 PT-BOO-S-012		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Call checkForConflictingReservations() providing proper reservation data info	Returns true or false depending on whether resource conflicts are detected for time overlapping with pre-existing in the MasterDB reservations		
2	Call getReservations() providing appropriate search criteria (time, user etc.)	Inspect response and ensure data is in line with the information stored in the MasterDB		

Table 48: Verification test of Booking Service simultaneous reservations support

Test ID: **BS08**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	-		
Software Configuration	-		
Test Name:	***Booking Service simultaneous reservations support***		
Preconditions	• Master DB is prepopulated with reservations of different status and timeslots (involved tables are: Reservation, Resource_Reservation)		
Related Requirements	PT-BOO-S-002 PT-BOO-S-010		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Multiple calls of Booking Service addReservation() method (execute BS01 multiple times simultaneously from different clients)	Ensure that all requests are processed and multiple reservations are created in the MasterDB		

5.1.2.5 Launching Service

All Test Procedures LS01, LS02, LS03, LS04 remain unchanged compared to what was defined in the previous version of the deliverable (D4.6).

Table 49: Verification test of the Launching Service manualStart (short term launching)

Test ID: **LS01**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Experiment short term launching***		
Preconditions	• Requires the Message Bus and the experiment controller to be accessible. • The master data repository should contain reservations for the user and for a defined experiment (involved tables are Experiment Experiment_Execution., Reservation, Reservation_item)		
Related Requirements	PT-LAU-S-001 PT-LAU-S-003 PT-LAU-S-004 PT-LAU-S-005 PT-LAU-S-007 PT-LAU-S-008 PT-LAU-S-009 (by design) PT-LAU-S-012 PT-LAU-S-013 (by design)		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	User call manualStart() providing an experiment Id	if experimentId is not present in the MasterDB then a proper failure message is returned		
		If supplied user credentials do not match an authorized user then a proper failure message is returned		
		If supplied user credentials match an authorized user but refer to booked resources of another user then a proper failure message is returned		
2	(case experimentId exists)	if an executionId already exists and refers to a running experiment (status=Ongoing) then a proper failure message is returned		
3	(case no executionId exists or exists for an status!=Ongoing)	Launching service generates an ExperimentStartRequest to the Message Bus (targeting the Experiment Controller).		
		Master DB tables are properly updated (tables Experiment_Execution, Reservation_item)		
		LaunchingServiceActionResp json message is returned containing the generated executionId and the status of the experiment		

Table 50: Verification test of the Launching Service schedule (long term launching)

Test ID: **LS02**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Experiment long term launching***		
Preconditions	• Requires the Message Bus and the experiment controller to be accessible. • The master data repository should contain reservations for the user and for a defined experiment (involved tables are Experiment Experiment_Execution., Reservation, Reservation_item) • The platform launching scheduler must be running		
Related Requirements	PT-LAU-S-002 PT-LAU-S-003 PT-LAU-S-004 PT-LAU-S-005 PT-LAU-S-007 PT-LAU-S-008 PT-LAU-S-009 (by design) PT-LAU-S-011, PT-LAU-S-012 PT-LAU-S-013 (by design) PT-LAU-S-014		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	User call schedule() providing experimentId, startDate, endDate	if experimentId is not present in the MasterDB then a proper failure message is returned		
		If supplied user credentials do not match an authorized user then a proper failure message is returned		
		If supplied user credentials match an authorized user but refer to booked resources of another user then a proper failure message is returned		
		If startDate or, endDate refer to past time then a proper failure message is returned		
		If startDate or endDate are not contained within the timeslot defined for the associated reservation then a proper failure message is returned		
		if an executionId already exists and refers to a running experiment (status=Ongoing) then a proper failure message is returned		
2	Scheduling part (case all preconditions are met)	Launching Scheduler is called and a job is added to be launched at the specified startDate		
		The user (owner) of the experiment and the testbed operator are informed by an appropriate notification (email)		
		Master DB tables are properly updated (tables Experiment_Execution, Reservation_item). The status of the		

		experiment should be BOOKED		
		LaunchingServiceActionResp json message is returned containing the generated executionId and the status of the experiment		
3	Execution part (check Launching Service activity when startDate arrives)	Master DB tables are properly updated (tables Experiment_Execution, Reservation_item) The status of the experiment changes to ONGOING		
		Launching service generates an ExperimentStartRequest to the Message Bus (targeting the Experiment Controller).		
		Scheduled job (for the executionId) is removed from scheduler		

Table 51: Verification test of the Launching Service cancellation request

Test ID: **LS03**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Experiment cancellation request***		
Preconditions	• Requires the Message Bus and the experiment controller to be accessible. • The master data repository should contain reservations for the user and for a defined experiment (involved tables are Experiment Experiment_Execution., Reservation, Reservation_item) • An experiment should be schedule for a future time		
Related Requirements	PT-LAU-S-009 (by design) PT-LAU-S-010 PT-LAU-S-012 PT-LAU-S-013 (by design)		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	User call cancellation() providing an executionId	if executionId is not present in the MasterDB then a proper failure message is returned		
		If supplied user credentials do not match an authorized user then a proper failure message is returned		
		If supplied user credentials match an authorized user but refer to an experiment of another experimenter then a proper failure message is returned (Exception to this rule if credentials refer to a testbed operator or administrator)		
2	(case executionId exists)	If the experiment is already running (status= ONGOING) then cancellation is not possible and a proper failure message is returned		
		If no schedule job is found in Launching scheduler then a proper failure message is returned		
3	(executionId exists and the execution is still in the scheduler)	Job is removed from the scheduler		
		Master DB tables are properly updated (tables Experiment_Execution, Reservation_item) The status of the experiment changes to CANCELLED		
		LaunchingServiceActionResp json message is returned containing with the executionId,		

		status= CANCELLED and empty message field		
		The user (owner) of the experiment and the testbed operator are informed by an appropriate notification (email)		

Table 52: Verification test of Launching Service simultaneous launching capability

Test ID: **LS04**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	-		
Software Configuration	-		
Test Name:	***Launching Service simultaneous launching capability***		
Preconditions	• Master DB is prepopulated with reservations of different status and timeslots (involved tables are: Reservation, Resource_Reservation)		
Related Requirements	PT-LAU-S-006, PT-LAU-S-011		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Multiple calls of Launching Service schedule() method (execute LS01 multiple times simultaneously from different clients)	Ensure that all requests are processed multiple experiments executions exist in the Job Scheduler		

5.1.2.6 Visualisation Engine

Table 55: Visualisation engine user request handling

Test ID: **VE01**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Connection Test***		
Preconditions	• Requires visualization tool and visualization engine to function and be accessible		
Related Requirements	VIS01		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Visualization engine receive through websocket request from visualization tool	The visualization engine handles the request		
2	Visualization engine sends through websocket the response	Visualization tool receives response		

Table 56: Visualisation engine user request handling

Test ID: **VE02**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***User Request Test***		
Preconditions	• Requires visualization tool and visualization engine to function and be accessible		
Related Requirements	VIS01, VIS02		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Visualization engine receive through websocket request from visualization tool	The visualization engine handles the request		
2	Visualization engine sends through websocket the response	Visualization tool receives response		

Table 57: Visualization engine geospatial data modification

Test ID: **VE03**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Geo Data Test***		
Preconditions	• Requires visualization tool and visualization engine to function and be accessible		
Related Requirements	VIS01,VIS02		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	visualisation engine receives through the message bus data from the visualisation tool	The visualization engine handles the request		
2	Visualization engine updates data in database	Data is properly stored in the database for future retrieval		

Table 58: Visualization engine indoor map handling

Test ID: **VE04**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Indoor map test***		
Preconditions	• Requires visualization tool and visualization engine to function and be accessible and an indoor map to be loaded in the GeoServer and experiment using indoor map		
Related Requirements	VIS01, VIS02		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	visualisation engine receives a request from the visualisation tool to start an experiment that needs indoor map	the visualisation engine loads needed data from the db		
2	Visualization engine receives data from an UxV	Visualisation engine updates this data and forwards it to the VE		

5.1.2.7 Data Analysis Engine

Table 58: Verification test of accepting analysis tasks defined through the Data Analysis Tool

Test ID: **DAE01**	Conducted by:	Date:	Test Category: **Verification Tests (front end)**
Hardware Configuration			
Software Configuration			
Test Name:	***Accept analysis tasks defined through the Data Analysis Tool***		
Preconditions	• Requires the Zeppelin notebook interface of the DAT to be functioning and accessible • Requires result repository to be functioning and accessible		
Related Requirements	PT-DAA-S-001, PT-DAA-S-002		
Tools Used	•		

Step	Action	Expected Result	Status	Remarks
1	Authorized user logs into the web portal and clicks on the Zeppelin notebook GUI tab of the Data Analysis Tool GUI embedded into the web portal	Login successful, successfully reaches the Zeppelin notebook GUI tab of the Data Analysis Tool GUI embedded into the web portal		
2	User designs a spectrum of data analysis tasks in the notebook relying on various interpreters (e.g. Spark, Python, etc.). For a given task, the user starts it in its respective notebook. A tasks can be defined using predefined built-in algorithms or via procedures that the user would have designed from scratch within the interface.	The task has been successfully started (statement for a given task). The results (again, for a given task) are visible through the Grafana dashboard.		

Table 58: Verification test of scales properly to the addition of workers

Test ID: **DAE02**	Conducted by:	Date:	Test Category: **Verification Tests (front end)**
Hardware Configuration			
Software Configuration			
Test Name:	***Scales properly to the addition of workers***		
Preconditions	• Requires the Zeppelin notebook interface of the DAT to be functioning and accessible • Requires result repository to be functioning and accessible		
Related Requirements	PT-DAA-S-004, PT-DAA-T-004		
Tools Used	•		

Step	Action	Expected Result	Status	Remarks
1	Administrator designs and starts an analysis task via the Data Analysis Tool Zeppelin notebook GUI (see DAE01) under a given cluster configuration.	The task has been successfully started, results are visible on the Grafana dashboard (for streaming tasks).		
2	Administrator stops running task.	The task has been successfully stopped.		
3	Administrator increases the number of workers in the Spark cluster and launches the same task.	The task has been successfully started, results are visible on the Grafana dashboard (for streaming tasks). The results are similar to the results of the previous task.		

5.1.2.8 System Monitoring Service

Table 53: Verification test of the System Monitoring

Test ID: **SYMS01**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***System Monitoring***		
Preconditions	•		
Related Requirements	PT-SYM-S-001, PT-SYM-S-002		
Tools Used			

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Service polls the computes of the middle tier for their status	Computes return their health status to the service		
2	Service listen to status messages on the message bus	Testbed component sent automatically status information on the message bus. Messages received by the service		
3	System Monitory Tool request status information	Service collects the information and returns it		

Table 54: Verification test of sending notification on system monitoring error

Test ID: **SYMS02**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***System Monitoring Problem Notifications***		
Preconditions	• Notification receivers are configured • Status information is collected • connection System Monitoring Service and Tool • administrative knowledge about the system state needed on user side (to check results) • administrative access to a server to shutdown a server		
Related Requirements	PT-SYM-T-001, PT-SYM-S-003		
Tools Used	•		

Step	Action	Expected Result	Status	Remarks
1	User shuts down a server of RAWFIE	An error notification (e.g. email) should be sent by the System Monitoring Service to the administrators		
	user opens System Monitoring Tool in the Web Portal	the System Monitoring Tool request the data from the Service and displays the server in critical state		
	User restarts the server of RAWFIE	A recovery notification (e.g. email) should be sent by the System Monitoring Service to the administrators		
	user opens System Monitoring Tool in the Web Portal	the System Monitoring Tool request the data from the Service and displays the server in OK state		

Table 55: Verification test of sending notification on planned downtime

Test ID: **SYMS03**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***System Monitoring Problem Notifications***		
Preconditions	• Notification receivers are configured		
Related Requirements	PT-SYM-S-005		
Tools Used	•		

Step	Action	Expected Result	Status	Remarks
1	User marks a service with downtime start	A notification (e.g. email) should be sent by the System Monitoring Service to the administrators.		
1	User marks a service with downtime end	A notification (e.g. email) should be sent by the System Monitoring Service to the administrators.		

5.1.2.9 Accounting Service

Table 56: Verification test of the accounting data collection

Test ID: **ACCS01**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Accounting data collection***		
Preconditions	• Accounting data is empty for the used user • Experimenter 1 and experimenter 2 have different active cost models		
Related Requirements	PT-ACC-S-001, PT-ACC-S-002, PT-ACC-S-003, PT-ACC-S-004, PT-ACC-S-005, PT-ACC-S-006		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Experiment of experimenter 1 is completed. Notifications sent on the message bus.	Accounting received the event and computes the charge for the experiments based on the active cost model of experimenter 1		
2	Experiment of experimenter 2 is completed. Notifications sent on the message bus.	Accounting received the event and computes the charge for the experiments based on the active cost model of experimenter 2 (should be different to experimenter 1)		
3	Billing period ends	Bill is created and sent to the both experimenters		

Table 57: Verification test of the account charging

Test ID: **ACCS01**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Account charging***		
Preconditions	• User has an external payment system account		
Related Requirements			
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	User opens the user profile page in the Web Portal and klicks on account charging. .	Redirect to payment system selection and the to the external payment system		
2	User executes the payment	Payment is added to the account balance		

5.1.2.10 Experiment Controller

The Experiment Controller component requirement not addressed by the tests specified below is

- PT-EXP-C-001 “Cancellation of running experiments should be possible”.

Justification:
The cancellation of an ongoing experiment is possible through direct communication between Experiment Monitoring Tool (see 5.1.1.6 paragraph) and the Resource Controller.

Test procedures EC01 and EC02 are updated versions of the ones defined in D4.6 with extra expected results added. An additional test procedure EC03 has been performed to verify the ability of the Experiment Controller to support (in parallel) experiments running in multiple testbeds.

Table 58 Verification test of experiment forwarding

Test ID: **EC01**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	-		
Software Configuration	-		
Test Name:	***Forward experiment details to the corresponding testbed***		
Preconditions	• Requires the Message Bus to be accessible • Requires the corresponding instance Resource Controller to be up and running • Requires the entries on the corresponding tables in the Master DB to be appropriately filled in.		
Related Requirements	PT-EXP-C-002		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Send an ExperimentLaunchRequest type of message	Experiment Controller properly consumes the message.		
		Interaction with the Master DB to retrieve all the required information. During this procedure, the following fields are properly retrieved: • EDL script • Vehicles canonical names • Partitions IDs of all the involved vehicles • Coordinate system		
		An ExperimentStartRequest type of message is dispatched to the Kafka message bus.		

Table 59 Verification test of handling status updates of a running experiment

Test ID: **EC02**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	-		
Software Configuration	-		
Test Name:	***Status updates of a running experiment***		
Preconditions	• Requires the Message Bus to be accessible • Requires the corresponding instance Resource Controller to be up and running • Requires the entries on the corresponding tables in the Master DB to be appropriately filled in.		
Related Requirements	PT-EXP-C-006 PT-EXP-C-007 PT-EXP-C-008 PT-EXP-C-009		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Send an ExperimentStatusMsgtype type of message regarding a running experiment	Experiment Controller properly consumes the message and updates the following tables inside Master DB: • experimentlog • experiment_execution • experiment		

Table 60 Verification test of supporting experiments execution in multiple testbeds

Test ID: **EC03**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	-		
Software Configuration	-		
Test Name:	***Support execution of experiments in multiple testbeds – Parallel execution***		
Preconditions	• Requires the Message Bus to be accessible • Requires the corresponding instance Resource Controller to be up and running • Requires the entries on the corresponding tables in the Master DB to be appropriately filled in. • Requires that multiple testbeds are connected to the RAWFIE platform		
1Related Requirements	PT-EXP-C-003 PT-EXP-C-004		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Send an ExperimentLaunchRequest type of message for testbed A	Experiment Controller properly consumes the message and dispatch an ExperimentStartRequest type of message.		
		An instance of the Resource Controller, launched for testbed A, successfully receives the requested experiment.		
2	Send an ExperimentLaunchRequest type of message for testbed B	While the first experiment is executed, Experiment Controller properly consumes the new message and dispatch an ExperimentStartRequest type of message.		
		An instance of the Resource Controller, launched for testbed B, successfully receives the requested experiment.		
		During the execution of all the experiments, Experiment Controller receives distinct status messages of each experiment and properly updates the corresponding fields inside the Master DB.		

5.1.3 Testbed Tier

This section presents the test of the Testbeds and Resources control components.

5.1.3.1 Monitoring Manager

Monitoring Manager is tightly coupled with Testbed Manager coexisting in the same application running at testbed level enabling the user to have a close look at computing and UxV resources utilization.

The Monitoring Manager component requirement not addressed by the tests specified below is

- TB-MOM-005: Testing of this requirement presumes that other services with well-defined interfaces like Weather conditions service are available to make verification procedures feasible.

Test procedure MM01 is an updated version of that defined in D4.6 with extra steps added. Test procedure MM02 is almost identical to Test Manager's procedure TM03 in D4.6 which has been moved to Monitoring Manager section for better cohesion of monitoring activities.

Table 61: Verification test of UxV health status

Test ID: **MM01**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Check UxV health status***		

Preconditions	• Requires the Message Bus to be accessible
	• Requires the network controller to be accessible.
	• Requires the System Monitoring Service to be accessible
	• Initial UxV status configuration:
	  ○ Fuel usage WARNING < 50%, CRITICAL < 15%
	  ○ CPU usage WARNING > 50%, CRITICAL > 85%
	  ○ Storage usage WARNING > 50%, CRITICAL > 85%		

Related Requirements	TB-MOM-001, TB-MOM-003, TB-MOM004, PT-SYM-S-002, UXV-NOD-001, TB-UVG-001		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Monitoring Manager receives periodically messages from UxVs related to resources utilization (FuelUsage, CpuUsage, Storage Usage) from the message bus.	Monitoring Manager properly consumes the messages and displays the result in Monitoring Manager's User Interface		
2	Monitoring Manager calculates an overall UxV status upon predefined criteria for the above received messages	UxV status is displayed in Monitoring Manager's User Interface		
3	Monitoring Manager periodically transmits a message describing the UxV Status to the Message Bus	System Monitoring Service receives and displays the current status for each UxV		

Table 62: Verification test of testbed health status

Test ID: **MM02**	Conducted by:	Date:	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration Details			
Software Configuration Details			
Test Name:	***Check Testbed health status***		
Preconditions	• Requires middle tier to be accessible (System Monitoring Service) • Initial Testbed health status configuration: o CPU usage WARNING > 50%, CRITICAL >85% o Memory usage WARNING > 50%, CRITICAL >85% o Disk usage WARNING > 50%, CRITICAL >85% o Frequency of sending messages 30 sec		
Related Requirements	TB-MOM-002, TB-MOM-003, TB-MOM004, PT-SYM-S-002		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Monitoring Manager started	1. Monitoring manager successfully initialized 2. Monitoring Manager checks periodically CPU load, memory and disk usage		
2	Monitoring manager processing (status assessment)	3. A TestbedHealthStatus message is created containing an overall assessment (OK, WARNING, CRITICAL) for the usage metrics monitored 4. The message is sent to the Message bus		
3	Check System Monitoring Service UI display at Middle Tier	Display of Testbed health status. Initial status OK		
4	Artificially increase CPU or Memory usage	Status message sent to the message bus		i.e. by opening or running additional resource intensive applications in the machine where Testbed Manager is installed
5	Recheck System monitoring Service UI display at Middle Tier	Display of Testbed health status. Status changes to WARNING or CRITICAL		
6	Decrease CPU or Memory usage and recheck System monitoring Service UI display at Middle Tier	Display of Testbed health status. Status changes back to OK		Close extra running applications

5.1.3.2 Network Controller

Table 63: Verification test of network interface switching due to connectivity problems

Test ID: **NC01**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Switch network interface due to connectivity problem***		
Preconditions	• Requires the Testbed Manager to be accessible		
Related Requirements	TB-NEC-001, TB-NEC-003, TB-NEC-004		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	The Network Controller 'checks' the connectivity of the resources through the Resource Controller.	The Resource Controller informs the Network Controller for malfunctions in the network connectivity of the resources.		
2	The Network Controller receives the incoming messages from the Resource Controller.	The appropriate network interface is selected.		

Table 64: Verification test of network interface management

Test ID: **NC02**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Manage the communication interfaces***		
Preconditions	• Requires the Testbed Manager to be accessible		
Related Requirements	TB-NEC-001, TB-NEC-002, TB-NEC-003, TB-NEC-004		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	The Network Controller 'lists' the available communication interfaces (as resources) through the Resource Controller (or RAWFIE database).	The Resource Controller (or RAWFIE database) informs the Network Controller about the available Network interfaces.		
2	The Network Controller 'checks' the connectivity of the resources through the Resource Controller.	The Network Controller gets a status for every available Network interface.		
3	The Network Controller 'assigns' the Network interface to any requesting Testbed entity (typically a UxV).	The Network Controller informs the Testbed entity about the identification of the assigned Network interface (and updates the database).		
4	The Network Controller receives the incoming messages from the Resource Controller.	The appropriate network interface is selected.		

Table 65: Verification test of the performance of the communication interfaces

Test ID: **NC02**	Conducted by:	Date:	Test Category: **Verification Tests (middle tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Performance of the communication interfaces***		
Preconditions	• Requires an external entity for monitoring the Kafka message bus • Requires the Testbed Manager and Network controller to be accessible		
Related Requirements	TB-NEC-005		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	An external entity is configured to monitor the activity of the Network interfaces in particular for the temporal characteristics of the traffic.	The external entity is configured to monitor the Kafka message bus.		
2	An external entity monitors the activity of the Network interfaces in particular the temporal characteristics of the traffic.	The external entity notifies the Network controller whenever any time specification is not met or in absence of expected traffic.		

5.1.3.3 Resource Controller (plus Navigation Service sub-component)

Table 66 Verification test of starting/canceling an experiment

Test ID**: RC01**	Conducted by**:**	Date**:**	Test Category**: Verification Tests (testbed tier)**
Hardware Configuration	-		
Software Configuration	-		
Test Name:	***Start/Cancel an experiment***		
Preconditions	• Requires the Message Bus to be accessible. • Requires Experiment Controller to be up and running.		
Related Requirements	TB-REC-001 TB-REC-002 TB-REC-006		
Tools Used			

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Send an ExperimentStartRequest type of message	Resource Controller properly consumes the message (filtering out all the messages that do not belong to the specific testbed) and initiates the command and control loop.		
		An experiment status update is dispatched.		
2	Send an ExperimentCancelRequest type of message	Resource Controller properly consumes the message (filtering out all the messages that do not belong to the specific testbed) and dispatches abort commands to all the operational UxVs.		
		An experiment status update is dispatched.		

Table 67 Verification test of the command the control loop

Test ID: **RC02**	Conducted by:	Date:	Test Category: **Verification Tests (testbed tier)**
Hardware Configuration	-		
Software Configuration	-		
Test Name:	***Check functionality of command and control loop.***		
Preconditions	• Requires the Message Bus to be accessible. • Requires Experiment Controller to be up and running. • Requires all the involving UxVs to be operational.		
Related Requirements	TB-REC-003, TB-REC-004,TB-REC-005, UXV-NOD-001, UXV-SEN-004,		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Resource Controller sends a set of waypoints to all the involved UxVs	Each one of the involved UxV receives and proceeds to the commanded waypoint.		
2	UxV continuously sends actual location	Resource Controller receives actual position and checks if the UxV has reached the previously transmitted waypoint (within a pre-specified radius of tolerance).		
		Resource Controller sends the new set of waypoints, when all the operational UxVs have reached their current waypoints.		

5.1.3.4 UxV Proximity component

Table 68: Verification test of Proximity component Backup communication

Test ID**: UxP01**	Conducted by**:**	Date**:**	Test Category**: Verification Tests (UxV tier)**
Hardware Configuration	UxV with Proximity component		
Software Configuration			
Test Name:	***Backup communication***		
Preconditions	• UxV are equipped with the Proximity component		
Related Requirements	PT-GEN-001, PT-P-001, PT-P-003, PT-A-001, PT-A-003, PT-A-004, PT-A-005, PT-A-006, PT-A-007, ,PT-A-009, ,PT-A-014, PT-A-016, PT-B-001, PT-L-002, PT-E-002, PT-E-003, TB-G-004, TB-G-006, TB-I-001, TB-G-013, TB-D-001		
Tools Used			

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	The UxVs are booked, the experiment is programmed and started.			
2	The UxVs lose the connection with the primary RAWFIE communication system	The Proximity communication system takes over		
3	The UxVs act autonomously, following the loaded mission instructions, logging all motion parameters, exchanging information across the swarm	The UxV use the Proximity communication system.		
4	The UxVs come back and the logged information is analysed	The communication statistics exhibits low packet error rate and low latency		

Table 69: Verification test of UxV retrieval using the communication system of the Proximity component

Test ID: **UxP02**	Conducted by:	Date:	Test Category: **Verification Tests (UxV tier)**
Hardware Configuration	UxV with Proximity component		
Software Configuration			
Test Name:	***UxV retrieval***		
Preconditions	• UxV are equipped with the Proximity component		
Related Requirements	PT-GEN-001, PT-P-001, PT-P-003, PT-A-001, PT-A-003, PT-A-004, PT-A-005, PT-A-006, PT-A-007, ,PT-A-009, ,PT-A-014, PT-A-016, PT-B-001, PT-L-002, PT-E-002, PT-E-003, TB-G-004, TB-G-006, TB-I-001, TB-G-013, TB-D-001		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	The UxVs are booked, the experiment is programmed and started.			
2	The UxVs perform their mission and one of them exhausts its main power source			
3	The other UxVs uses the Proximity component communication systems to communicate and locate the stopped UxV	The connection is established with the stopped UxV and the collected information allows for locating it		
4	The other UxVs transmit the location and status of the stopped UxV to the RAWFIE resource manager			

Table 70: Verification test of Swarm motion using the Proximity component

Test ID: **UxP03**	Conducted by:	Date:	Test Category: **Verification Tests (UxV tier)**
Hardware Configuration	UxV with Proximity component		
Software Configuration			
Test Name:	***Swarm motion***		
Preconditions	• UxV are equipped with the Proximity component. • Acceptable margin for the relative location of UxV is defined depending on the type of UxV and the scenario dynamics.		
Related Requirements	PT-GEN-001, PT-P-001, PT-P-003, PT-A-001, PT-A-003, PT-A-004, PT-A-005, PT-A-006, PT-A-007, ,PT-A-009, ,PT-A-014, PT-A-016, PT-B-001, PT-L-002, PT-E-002, PT-E-003, TB-G-004, TB-G-006, TB-I-001, TB-G-013, TB-D-001		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	The UxVs are booked, the experiment is programmed and started.			
2	The UxVs perform their mission moving in a coordinated fashion			
3	The UxVs log all position			
4	The UxVs come back and the logged information is analysed	The UxV relative locations were within the acceptable margin		

5.1.3.5 Testbed Manager

Test procedures related to verifying Testbed Manager correct behaviour and adherence to requirements defined in D3.3 are provided in this section.

Test procedures TM01 and TM04 have been updated with extra steps added.

TM03 of D4.6 has been moved to Monitoring Manager section as MM02. TM02 of D4.6 has been eliminated based on the assumption that the actions specified in this test will be handled by proper Message Bus configuration.

TM02, TM03 and TM05 presented here are new.

Table 71: Verification test of experiment handling from testbed manager

Test ID: **TM01**	Conducted by:	Date:	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration Details			
Software Configuration Details			
Test Name:	***Testbed Manager Experiment Handling***		
Preconditions	• Requires middle tier to be accessible (Experiment Controller Service) • Requires the resource controller to be accessible • Requires local PostgreSQL Server accessible		
Related Requirements	TB-MAN-005 TB-MAN-004 TB-MAN-001 TB-MAN-007 TB-MAN-010		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Start Testbed Manager	Testbed manager successfully initialized Successful connection to the local (testbed site) database server		
2	Testbed Manager receives an ExperimentStartRequest message from Message Bus	A new experiment is registered in the local database. Testbed Manager rejects experiments not intended for this testbed		
3	Testbed Manager receives ExperimentStatusMsg messages from Message Bus	ExperimentStatusMsg messages are periodically transmitted from Resource Controller providing the current status of the experiment. Upon reception of a final state message the experiment is registered either as completed, failed or cancelled in the experiments history log in the local database		
4	Testbed Manager sends an ExperimentCancelRequest message to the Message Bus	Resource controller receives the message and initiates all necessary actions to safely stop all UxV resources. The experiment is registered as cancelled in the experiments history log in the local database		
5	User selects to see the experiments executed in the testbed	Information about the experiments executed in the testbed is retrieved from the local database (experiments log) and shown in the relevant window		

Table 72: Verification test of creating, updating and deleting a resource in the master database

Test ID: **TM02**	Conducted by:	Date:	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration Details			
Software Configuration Details			
Test Name:	***Register, update and delete a resource in master RAWFIE database***		
Preconditions	• Requires Testbed Directory Service		
Related Requirements	TB-MAN-002 TB-MAN-007 PT-GEN-R-004 PT-DIR-S-007		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	User starts Testbed Manager application in testbed site	Testbed manager successfully initialized Successful connection to the local (testbed site) database server		
2	The user creates a new UxV resource by editing the appropriate user interface window	A new resource is created in the master database using a REST call defined in Testbed Directory Service's API. The new resource is displayed in Resource Explorer Tool		
3	The user updates and existing UxV resource by editing the appropriate user interface window	The resource is updated in the master database using a REST call defined in Testbed Directory Service's API. The updated resource is displayed in Resource Explorer Tool		
4	The user deletes an existing UxV resource	The resource is deleted from the master database using a REST call defined in Testbed Directory Service's API. The resource now is not present in Resource Explorer Tool		

Table 73: Verification test of Aggregate Manager create, update and delete operations

Test ID: **TM03**	Conducted by:	Date:	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration Details			
Software Configuration Details			
Test Name:	***Register, update and delete a resource in SFA Aggregate Manager triple store database***		
Preconditions	• Requires Aggregate Manager REST API		
Related Requirements	TB-GEN-R-001 TB-AGG-005 TB-MAN-002 TB-MAN-007		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	User starts Testbed Manager application in testbed site	Testbed manager successfully initialized Successful connection to the local (testbed site) database server		
2	The user creates a new UxV resource by editing the appropriate user interface window	A new resource is created in the triple-store database using a POST REST call defined in Aggregate Manager's API. The new resource is accessible from MySlice API		
3	The user updates and existing UxV resource by editing the appropriate user interface window	The resource is updated in the triple store database using a PUT REST call defined in Aggregate Manager's API. The updated resource is accessible from MySlice API		
4	The user deletes an existing UxV resource	The resource is deleted from triple-store database using a DELETE REST call defined in Aggregate Manager's API. The resource now is not present in MySlice API		

Table 74: Verification test of services running at testbed

Test ID: **TM04**	Conducted by:	Date:	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration Details			
Software Configuration Details			
Test Name:	***Check the status of all services running at testbed level***		
Preconditions	• Requires middle tier to be accessible (Experiment Controller Service) • Requires the resource controller to be accessible • Requires local PostgreSQL Server accessible		
Related Requirements	TB-MAN-009 TB-MAN-007		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	User starts Testbed Manager application in testbed site	Testbed manager successfully initialized Successful connection to the local (testbed site) database server		
2	Testbed manager receives periodical status messages from Resource Controller and Network Manager in the Message Bus			
3	User is able to see the availability of the components that run at testbed level by selecting the appropriate user interface window	Show current status of components running at testbed level		

Table 75: Verification test of testbed statistics display

Test ID: **TM05**	Conducted by:	Date:	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration Details			
Software Configuration Details			
Test Name:	***Display testbed statistics***		
Preconditions	• Requires the Message Bus to be accessible • Requires middle tier to be accessible (Experiment Controller Service) • Requires local PostgreSQL Server accessible		
Related Requirements	TB-MAN-009 TB-MAN-007		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	User starts Testbed Manager application in testbed site	Testbed manager successfully initialized Successful connection to the local (testbed site) database server		
2	The user selects to see statistical information related to testbed usage by selecting the appropriate user interface window	Statistical information about testbed alive time, number of experiments completed/failed/cancelled and information about time utilization and participation in experiments per resource is displayed		
3	A new experiment is executed in the testbed	See TM01 above		
4	The user selects to see statistical information related to testbed usage by selecting the appropriate user interface window	Statistical information has been updated		

5.1.3.6 UxV Node

Table 76: Verification test of UxV Return to base

Test ID: **UxV01**	Conducted by:	Date:	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Return to base***		
Preconditions	- Requires the RAWFIE system to be operational (e.g. Resource controller reachable) - Requires the mission to be defined and running. - Requires the UxV to be ready to operating (e.g. en route). - Requires the UxV to be reachable by any communication mean.		
Related Requirements	PT-EXA-T-008, PT-NAV-T-001, PT-NAV-T-002, PT-NAV-T-003, PT-VIS-T-001, TB-REC-001, TB-REC-004, UXV-NET-009, UXV-SEN-003, UXV-SEN-005, UXV-PRC-001, UXV-MGT-002 ,UXV-PRC-003,UXV-PRC-005, UXV-MGT-006, UXV-NOD-001,UXV-SEN-004, TB-UVG-001		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Establish the communication with the UxV	Communication established		
2	Establish a secure control session	Secured control session established		
3	Send the return to base command	Return to base command received		
4	If the UxV is not autonomous, instruct it with the necessary waypoint or guidance information, possibly until the end of the test	Further optional instructions for returning home received, Confirmation of the UxV at home		
5	Close the secure control session.	The UxV is home after a safe return. Connection closed		

Table 77: Verification test of the ability of the UxV to follow a route

Test ID: **UxV02**	Conducted by:	Date:	Test Category: **Verification Tests (testbed tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Follow a route***		
Preconditions	- Requires the RAWFIE system to be operational (e.g. Resource controller reachable) - Requires the mission to be defined and running. - Requires the UxV to be ready to operating (e.g. en route). - Requires the UxV to be reachable by any communication mean.		
Related Requirements	PT-EXA-T-008, PT-NAV-T-001, PT-NAV-T-002, PT-NAV-T-004, PT-VIS-T-001, TB-REC-001, TB-REC-004, UXV-NET-009, UXV-SEN-003, UXV-SEN-004, UXV-SEN-005, UXV-PRC-001, UXV-NOD-001, TB-UVG-001		
Tools Used			

Step	**Action**	**Expected Result**	**Status**	**Step**
1	Resource controller computes mission and send waypoint	Robot proceeds to the specified point,		
2	Robot continuously sends actual location	RC receives position and check if WP have been reached		
3	RC sends next point	Robot receives and proceed to next point		

Table 78: Verification test of Acquire sensor samples

Test ID: **UxV03**	Conducted by:	Date:	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Acquire sensor samples***		
Preconditions	- Requires the RAWFIE system to be operational - Requires the mission to be defined and running. - Requires the UxV to be ready to operating (e.g. en route). - Requires the UxV to be reachable by any communication mean.		
Related Requirements	PT-NF-001, UXV-SEN-004, UXV-SEN-005, UXV-STO-001, UXV-STO-002, UXV-NET-006, UXV-NET-007, PT-VIS-T-003, TB-MAN-004, UXV-STO-001, UXV-STO-002, UXV-STO-003, UXV-STO-004, UXV-SEN-001, UXV-SEN-002, UXV-SEN-003, UXV-SEN-005, UXV-MGT-001, UXV-NOD-001, UXV-MGT-006- TB-UVG-001		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Establish the communication with the UxV	Communication established		
2	Establish a secure control session (if not done already)	Secured control session established		
3	Send the acquisition commands	Commands received and executed		
4	Store sensor samples and, if possible, transmit them via the data communication system	Samples stored and, if possible, transmitted		
5	If opened specifically for the matter of the test, close the secure control session.	Sensor samples have acquired correctly and are stored in the UxV memory or in the experiment database. Connection closed		

Table 79: Verification test of Fidelity to commands

Test ID: **UxV04**	Conducted by:	Date:	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Fidelity to commands***		
Preconditions	- Requires the RAWFIE system to be operational - Requires the mission to be defined and running. - Requires the UxV to be ready to operating (e.g. en route). - Requires the UxV to be reachable by any communication mean.		
Related Requirements	UXV-NET-006, UXV-NET-007, PT-NF-001, TB-MOM-003, TB-MAN-004, UXV-STO-001, UXV-STO-002, UXV-STO-003, UXV-STO-004,, TB-UVG-001, UXV-NOD-001, UXV-PRC-003, UXV-PRC-005		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Establish the communication with the UxV	Communication established		
2	Establish a secure control session (if not done already)	Secured control session established		
3	Send repeatedly pre-defined sets of commands, covering the full range of possible UxV actions,	Commands received and executed		
4	Check the conformance of the undertaken actions and corrections (if necessary) to the commands,	Undertaken actions in conformance to the commands		
5	Record all fine-grained status of the UxV over the duration of the test, to be able to reconstruct the behavior of the UxV,	Status recorded		
6	If opened specifically for the matter of the test, close the secure control session.	Sensor samples have acquired correctly and are stored in the UxV memory or in the experiment database. Connection closed		

Table 80: Verification test of Continuous communication

Test ID: **UxV05**	Conducted by:	Date:	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Continuous communication***		
Preconditions	- Requires the RAWFIE system to be operational - Requires the mission to be defined and running. - Requires the UxV to be ready to operating. - Requires the UxV to be reachable by any communication mean.		
Related Requirements	UXV-NET-006, UXV-NET-007, TB-MOM-003, UXV-STO-004, UXV-MGT-006, TB-UVG-001		
Tools Used			

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Establish the communication with the UxV	Communication established		
2	Exchange a predefined set of commands and data.	Commands and data correctly exchanged		
3	Close the communication session.	Communication closed		

Table 81: Verification test of Secure communication

Test ID: **UxV06**	Conducted by:	Date:	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Secure communication***		
Preconditions	- Requires the RAWFIE system to be operational - Requires the UxV to be ready to operating. - Requires the UxV to be reachable by any communication mean.		
Related Requirements	UXV-NET-006, UXV-NET-007, PT-NF-001, TB-MOM-003, UXV-STO-004, UXV-MGT-006,TB-UVG-001		
Tools Used			

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Establish the communication with the UxV	Communication established		
2	Establish a secure control session (if not done already)	Secured control session established		
3	Check communication parameters	Communication parameters and status are correct and matching		
4	Exchange a pre-defined set of commands and data,	Commands and data correctly exchanged		
5	Close the secure control session.	Connection closed		

Table 82: Verification test of Real-time communication

Test ID: **UxV07**	Conducted by:	Date:	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Real-time communication***		
Preconditions	- Requires the RAWFIE system to be operational - Requires the mission to be defined and running. - Requires the UxV to be ready to operating (e.g. en route). - Requires the UxV to be reachable by any communication mean.		
Related Requirements	UXV-NET-006, UXV-NET-007, PT-NF-001, TB-MOM-003, UXV-STO-004, TB-UVG-001, UXV-NOD-001, UXV-PRC-003, UXV-PRC-005, UXV-MGT-006		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Establish the communication with the UxV	Communication established		
2	Establish a secure control session (if not done already)	Secured control session established		
3	Send safe commands and measure the temporal characteristics of the communication (e.g. response time, synchronisation of reception across a swarm of UxV (coordinated group of UxV), etc.).	Real-time constraints applicable to the exchanged commands are met or mismatches are detected		
4	Close the secure control session.	Connection closed		

Table 83: Verification test of Resume communication and data transfer

Test ID: **UxV08**	Conducted by:	Date:	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Resume communication and data transfer***		
Preconditions	• Requires the RAWFIE system to be operational • Requires the mission to be defined and running. • Requires the UxV to be ready to operating. • Requires the UxV to be reachable (at least sporadically) by any communication mean.		
Related Requirements	UXV-NET-006, UXV-NET-007, TB-MOM-003, TB-MAN-004, UXV-STO-001, UXV-STO-002, UXV-STO-003, UXV-STO-004, TB-UVG-001, UXV-MGT-003, UXV-MGT-006		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Establish the communication with the UxV	Communication established		
2	Start a transaction.	Transaction started		
3	Interrupt the communication at the low-level (e.g. disconnect the antenna)	Communication is interrupted, the transaction is not complete.		
4	Re-establish the communication low level means	The transaction resumes and completes		
5	Close the communication session.	Connection closed		

Table 84: Verification test of UxV Device Management

Test ID: **UxV9**	Conducted by:	Date:	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***UxV Device Management***		
Preconditions	• Requires the RAWFIE system to be operational • Requires the mission to be defined and running. • Requires the UxV to be ready to operating (e.g. en route). • Requires the UxV to be reachable by any communication mean.		
Related Requirements	UXV-NET-006, UXV-NET-007, PT-NF-001, TB-MOM-003, TB-MAN-004, UXV-STO-001, UXV-STO-002,UXV-STO-003, UXV-STO-004, UXV-MGT-006		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Establish the communication with the UxV	Communication established		
2	Establish a secure control session (if not done already)	Secured control session established		
3	Send device management commands	Command received and applied		
4	Check and log the status of the device	Device has responded to the commands according to the specification		
5	Close the secure control session.	The UxV is home after a safe return. Connection closed		

Table 85: Verification test of the UxV connection

Test ID: **UxV10**	Conducted by:	Date:	Test Category**: Verification Tests (testbed tier)**
Hardware Configuration			
Software Configuration			
Test Name:	**UxV Connection Test**		
Preconditions	UxV-Node launched, Message bus working		
Related Requirement	UXV-NET-006, UXV-NET-007, TB-MOM-003, UXV-STO-004		
Tools Used			

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Kafka Subscriber is called from another machine	Topic is shown with UxV information being published		
2	Kafka Publisher is called with a valid waypoint	Robot proceeds to the specified point		

Table 86: Verification test of Sensor Data Acquisition 1

Test ID: **UxV11**	Conducted by:	Date:	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Sensor Data Acquisition 1***		
Preconditions	- UxV is in operation state and the parent UxV node has been launched - Network Communication is also fully functional		
Related Requirements	UXV-NET-006, UXV-NET-007, PT-NF-001, TB-MOM-003, TB-MAN-004, UXV-STO-001, UXV-STO-002,UXV-STO-003, UXV-STO-004, UXV-SEN-004, UXV-MGT-001, UXV-MGT-006		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Establish the communication with the UxV	Communication established		
2	Establish a secure control session (if not done already)	Secured control session established		
3	Acquire sensor data	Data acquired (every sensor works as specified)		
4	Send acquired data	Data received		
5	Close the secure control session.	The UxV is home after a safe return. Connection closed		

Table 87: Verification test of Sensor Data Acquisition 2

Test ID: **UxV12**	Conducted by:	Date:	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Sensor Data Acquisition 2***		
Preconditions	- UxV is in operation state and the parent UxV node has been launched - Network Communication is also fully functional		
Related Requirements	UXV-NET-006, UXV-NET-007, PT-NF-001, TB-MOM-003, TB-MAN-004, UXV-STO-001, UXV-STO-002,UXV-STO-003, UXV-STO-004, UXV-SEN-004, UXV-MGT-001, UXV-MGT-006		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Establish the communication with the UxV	Communication established		
2	Establish a secure control session (if not done already)	Secured control session established		
3	Instruct the robot to move to a known location	Robot at the specific location		
4	Acquire current location data	Location data acquired (location sensor works as specified)		
5	Send acquired location data	Data received		
6	Close the secure control session.	The UxV is home after a safe return. Connection closed		

Table 88: Verification test of Data Storage

Test ID: **UxV13**	Conducted by:	Date:	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Data Storage***		
Preconditions	- UxV is in operation state and the parent UxV node has been launched. - Sensor node is functional		
Related Requirements	UXV-NET-006, UXV-NET-007, TB-MAN-004, UXV-STO-001, UXV-STO-002,UXV-STO-003, UXV-STO-004, TB-MAN-004, UXV-STO-001, UXV-STO-002, UXV-STO-003, UXV-STO-004, UXV-STO-005, UXV-MGT-006		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Establish the communication with the UxV	Communication established		
2	Establish a secure control session (if not done already)	Secured control session established		
3	A request for storing certain data is done	Command received and data is stored locally		
4	After a mission given, data storage in the system is checked.	Data was correctly stored and kept.		
5	Close the secure control session.	The UxV is home after a safe return. Connection closed		

Table 89: Verification test of Waypoints Processed

Test ID: **UxV14**	Conducted by:	Date:	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Waypoints Processed***		
Preconditions	- UxV is in operation state and the UxV parent node has been launched. - Sensor node is functional, network communication is functional		
Related Requirements	UXV-NET-006, UXV-NET-007, TB-MAN-004, UXV-STO-001, UXV-STO-002,UXV-STO-003, UXV-STO-004, UXV-SEN-004, UXV-MGT-006		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Establish the communication with the UxV	Communication established		
2	Establish a secure control session (if not done already)	Secured control session established		
3	Waypoints are sent to the UxV	UxV receives and processes the waypoints		
4	The calculated route is applied to the UxV	The actual trajectory matches the route calculated by the navigation.		
5	Iterate step 4 until assessment is complete	UxV stops, informs and recalculate its route to next waypoint if an unexpected obstacle is found.		
6	Close the secure control session.	The UxV is home after a safe return. Connection closed		

5.2 Integrated system testing

As well as testing each individual component, the system will also be tested as a whole unit to validate its overall behaviour. Testing will be covered in the following areas:

The integrated testing procedure will be detailed during the first development iteration. The testing procedure will be based on the successful chain of verification scenarios described in Section 2 of this document.

Such scenarios will correspond to sequences and combinations of the components tests.

6 Validation scenarios

This chapter describes the validation scenarios. Some have been defined by the selected users of the RAWFIE system. Other simpler and more dedicated scenarios can focus on the evaluation of specific characteristics or behaviours of the RAWFIE components, testbeds,

federation, etc. They are defined on the basis of requirements described in D3.1, D3.2 and D3.3. Other scenarios may be defined on the basis of user defined use cases.

6.1 User defined scenarios

In the first version of requirements' deliverable (D3.1) a set of user scenarios were defined with the purpose to serve as a starting point for identifying the needs and assisting the elicitation of high level system wide requirements, for the potential experiments that should be supported by the platform. D3.2 added two further scenarios. From the evaluation of the 1st Open Call also several new scenarios where derived.

These user defined scenarios can be considered as a starting point for the definition of appropriate activities and steps that can be used for the overall RAWFIE platform validation. Despite their differences in nature and purpose, when considering them from the RAWFIE platform perspective, a set of common general steps can be identified for all of them. These general steps are summarized below:

1. The experimenter logs in to the Web Portal
2. The experimenter looks for a UxV testbed where the UxVs could be or are equipped with the technology T (e.g. infra-red cameras, ZigBee transmitters, radar, etc.) and the testbed provides an environment E.
3. The experimenter books resources in a testbed for the desirable timeframe.
4. The experimenter writes the experiment steps with EDL. Depending on the experimenter an algorithm *A* may be declared in EDL using the provided API.
5. The experiment is scheduled for execution at the given timeframe (actual resources are associated with it during this step)
6. UxV gets equipped with technology T by the support personal, if necessary.
7. The experiment is launched
8. UxVs execute the given script correctly. Declared algorithm A is carried out.
9. Measurements M are sent to message bus, evaluated by the algorithms and stored in the database.
10. Experimenter observes the experiment via the appropriate platform services (Experimenter monitoring Tool, Visualization Tool) and can intervene to the execution if need be..
11. The experiment completes.
12. The experimenter evaluates the results/measurements and possibly assesses the behavior of the applied algorithm *A* and technology *T* through the appropriate platform services (experiment log, Data Analysis Tool, etc.)

It becomes evident that the differences will be in the used technologies (T), recorded measurements (M), the testbed environment (E) and the possible algorithms (A) that need to be reflected in EDL.

The involved subsystems will be most of the time:

- Resource Explorer Tool
- Testbeds Directory Service
- Booking Tool
- Booking Service
- Experiment Authoring Tool
- EDL Compiler & Validator
- Experiment Validation Service
- Launching Service
- Experiment Controller
- Resource Controller
- Experiment Monitoring Tool
- Visualization Tool
- Visualization Engine
- Data Analysis Tool
- Data Analysis Engine
- Network Controller
- Proximity Component

Based on the above steps a Common User Defined Validation Scenario template has been compiled which is provided below

Table 90: Common User Defined Validation Scenario

Scenario ID**: UD-GEN**	Conducted by**:**	Date**:**
Title	Common User Defined Validation Scenario template	
Comments	This is a generic template. The steps prescribed should be adopted by all User Defined Validation Scenarios. For specific technologies T, environment of execution E, applicable algorithms E, sensor measurements M, please refer to the UD-xx scenarios analysed in D4.6	
Main stakeholder	Experimenter	
Secondary stakeholder		
Involved Sub-systems	Resource Explorer Tool Testbeds Directory Service Booking Tool Booking Service Experiment Authoring Tool EDL Compiler & Validator Experiment Validation Service Launching Service Experiment Monitoring Tool Visualization Tool Visualization Engine Data Analysis Tool Data Analysis Engine	
Validated requirement		

Step	**Description**	**Status**	**Remarks**
1	Experimenter logs in to the Web Portal		
2	Experimenter browses testbed and UxV resources, via the Resource Explorer Tool, looking for UxVs equipped with a specific technology T (e.g. infra-red cameras, ZigBee transmitters, radar, etc.) and a testbed environment E		• For T and E refer to specific UD scenarios defined
3	Experimenters navigates to the Booking Tool and books resources in a testbed for the desirable timeframe		• Booking request should be created in pending state
4	Experimenter navigates to Experiment Authoring and writes the experiment steps with EDL. Depending on the experimenter an algorithm A may be declared in EDL using the provided API		
5	After completing authoring, the experimenter schedules the experiment for execution at a feature timeframe (actual resources are associated with it during this step)		• execution timeframe should be a subset of the initial booking request timeframe • certain validation tests should apply
6	UxV gets equipped with technology T by the support personal, if necessary		• Action in responsibility of the testbed facility personnel
7	Experiment launching take place		• Booking request should have been confirmed
8	UxVs execute the given script correctly. Declared algorithm A or set of actions are carried out.		• Communication should go through ResourceController amd MessageBus • For A refer to specific UD scenarios defined
9	Measurements M are sent from UxVs to message bus, evaluated by the algorithms and stored in the database.		• Data Analysis Tool may be involved • For M refer to specific UD scenarios defined
10	Experimenter observes the experiment (i.e. route, sensor reading) via the appropriate platform services (Experimenter monitoring Tool, Visualization Tool)		
11	The experiment completes		
12	The experimenter evaluates the results/measurements and possibly assesses the behaviour of the applied algorithm A and technology T through the appropriate platform services (experiment log, Data Analysis Tool, etc.)		

Metric	**Success criteria**	**Status**	**Remarks**
PLATFORM / PERF / 1 /	Downtime < 2%	Success	

STABLE SYSTEM			
PLATFORM / PERF / 2 / ERRORS	Errors to experiments rate < 5 %	Success	
PLATFORM / PERF / 5 / LATENCY/ RESULTS UPDATE TIME	Update time < 5 sec	Success	
PLATFORM / PERF / 6 / LATENCY/ BOOKING TIME	Booking Time < 30 seconds	Success	
PLATFORM / USE / 7 / NOTIFICATION	Questionnaire rates “notification” with an average > 3.5 (1 to 5)	4	
PLATFORM / USE / 8 / ROLES	Questionnaire rates “roles” with an average > 3.5 (1 to 5)		
PLATFORM / USE / 9 / VISUALISATION / BALANCE	Questionnaire rates “balance” with an average > 3.5 (1 to 5)		
PLATFORM / USE / 10 / VISUALISATION / SIMPLICITY	Questionnaire rates “simplicity” with an average > 3.5 (1 to 5)	5	
PLATFORM / USE / 11 / VISUALISATION / CONSISTENCY	Questionnaire rates “consistency” with an average > 3.5 (1 to 5)	4	
PLATFORM / USE / 12 / VISUALISATION / UTILITY	Questionnaire rates “utility” with an average > 3.5 (1 to 5)	4	
PLATFORM / USE / 13 / GUIDANCE	Questionnaire rates “guidance” with an average > 3.5 (1 to 5)	4	
PLATFORM / USE / 14 / FILTERING	Questionnaire rates “filtering” with an average > 3.5 (1 to 5)	4	
PLATFORM / USE / 15 / EXPERIMENTS STATISTICS			
TESTBED / DATA / 1 / INFORMATION	Daily updates. Always available during testbed working hours.	Not measured	
TESTBED / FUNC / 3 / AVAILABILITY	Downtime for maintenance, as well as other planned unavailability which may prevent the execution of the experiments should be communicated in advance, at least 2 days before.		
TESTBED / USE / 4 / CONSISTENCY	Questionnaire rates “consistency” with an average > 3.5 (1 to 5)	5	
UxV / FUNC / 1 / COHERENCE	Questionnaire rates “coherence” with an average > 3.5 (1 to 5)	5	
UxV / FUNC/ 2 / MISSION ACHIEVEMENT	Questionnaire rates “mission achievement” with an average > 3.5 (1 to 5)	5	

6.2 RAWFIE Platform Admin scenarios

6.2.1 Administrator manages the user rights

Table 91: "Administrator manages the user rights" Validation Scenario

Scenario ID: **PA-01**	Conducted by:	Date:
Title	Administrator manages the user rights	
Comments		
Main stakeholder	RAWFIE Admin	
Secondary stakeholder	Experimenters	
Involved Sub-systems	Web Portal Users & Rights Service	
Validated requirement	PT-GEN-R-002, PT-WEB-P-001, PT-WIK-002	

Step	**Description**	**Status**	**Remarks**
1	User tries to edit the Wiki, which fails due to missing rights.		
2	Administrator opens the user management of the Web Portal		
3	Administrator searches for a given user		
4	Administrator changes the rights of the given user		
5	User tries to edit the Wiki again and succeeds.		

Metric	**Success criteria**	**Status**	**Remarks**
PLATFORM / USE / 9 / VISUALISATION / BALANCE	Questionnaire rates "balance" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 10 / VISUALISATION / SIMPLICITY	Questionnaire rates "simplicity" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 12 / VISUALISATION / UTILITY	Questionnaire rates "utility" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 13 / GUIDANCE	Questionnaire rates "guidance" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 14 / FILTERING	Questionnaire rates "filtering" with an average > 3.5 (1 to 5)		

6.2.2 Administrators adds a new user

Table 92: "Administrators adds a new user" validation scenario

Scenario ID: **PA-02**	Conducted by:	Date:
Title	Administrators adds a new user	
Comments		
Main stakeholder	RAWFIE Admin	
Secondary stakeholder	Experimenters	
Involved Sub-systems	Web Portal Users & Rights Service	
Validated requirement	PT-GEN-R-002, PT-WEB-P-001, PT-USR-S-001, PT-USR-S-002	

Step	**Description**	**Status**	**Remarks**
1	New user tries to login (which fails as the account does not exist)		
2	Administrator opens the user management of the Web Portal		
3	Administrator clicks on "new user"		
4	Administrator inserts the user data and submits the data		
5	Users & Rights Service save the user		
6	Information is sent to the new user via email		
7	New user logs-in into the Web Portal		

Metric	**Success criteria**	**Status**	**Remarks**
PLATFORM / USE / 7 / NOTIFICATION	Questionnaire rates "notification" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 9 / VISUALISATION / BALANCE	Questionnaire rates "balance" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 10 / VISUALISATION / SIMPLICITY	Questionnaire rates "simplicity" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 12 / VISUALISATION / UTILITY	Questionnaire rates "utility" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 13 / GUIDANCE	Questionnaire rates "guidance" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 14 / FILTERING	Questionnaire rates "filtering" with an average > 3.5 (1 to 5)		

6.2.3 System monitoring and error notifications

Table 93: "System monitoring and error notifications" validation scenario

Scenario ID: **PA-03**	Conducted by:	Date:
Title	System monitoring and error notifications	
Comments		
Main stakeholder	RAWFIE Admin	
Secondary stakeholder		
Involved Sub-systems	Web Portal System Monitoring Tool System Monitoring Service (Launching Service)	
Validated requirement	PT-WEB-P-001	

Step	**Description**	**Status**	**Remarks**
1	Launching Service crashes (e.g. stopped manually)	n.a.	
2	System Monitoring Service checks system state and detects that Launching Service is not running		
3	System Monitoring Service sends a notification email to the administrator		
4	Administrator opens the System Monitoring Tool		
5	Administrator checks system state		
6	Administrator restarts Launching Service via some SSH client		
7	Administrator checks system state (now Launching Service is running again)		

Metric	**Success criteria**	**Status**	**Remarks**
PLATFORM / USE / 7 / NOTIFICATION	Questionnaire rates "notification" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 9 / VISUALISATION / BALANCE	Questionnaire rates "balance" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 10 / VISUALISATION / SIMPLICITY	Questionnaire rates "simplicity" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 12 / VISUALISATION / UTILITY	Questionnaire rates "utility" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 13 / GUIDANCE	Questionnaire rates "guidance" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 14 / FILTERING	Questionnaire rates "filtering" with an average > 3.5 (1 to 5)		

6.2.4 System stability

Table 94: "System stability" validation scenario

Scenario ID: **PA-04**	Conducted by:	Date:
Title	System stability	
Comments		
Main stakeholder	RAWFIE Admin	
Secondary stakeholder		
Involved Sub-systems	System Monitoring Tool (all monitored components / services)	
Validated requirement	PT-WEB-P-001	

Step	**Description**	**Status**	**Remarks**
1	RAWFIE system runs several weeks with several executed experiments	n.a.	
2	System Monitoring Service collects status information all the time	n.a.	
3	Administrator opens the System Monitoring Tool		
4	Administrator checks statistics about uptime and error counts		

Metric	**Success criteria**	**Status**	**Remarks**
PLATFORM / PERF / 1 / STABLE SYSTEM	Downtime < 2%		
PLATFORM / PERF / 2 / ERRORS	Errors to experiments rate < 5 %		
PLATFORM / PERF / 4 / RECOVERY TIME	Recovery in 1 hour after error occurs (during business time)		
PLATFORM / USE / 10 / VISUALISATION / SIMPLICITY	Questionnaire rates "simplicity" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 12 / VISUALISATION / UTILITY	Questionnaire rates "utility" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 13 / GUIDANCE	Questionnaire rates "guidance" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 14 / FILTERING	Questionnaire rates "filtering" with an average > 3.5 (1 to 5)		

6.3 Testbed operator scenarios

Table 95: "Connect a new testbed" validation scenario

Scenario ID: **TO-03**	Conducted by:	Date:
Title	Connect a new testbed	
Comment		
Main stakeholder	Testbed Operator	
Secondary stakeholder	RAWFIE Admin	
Involved Sub-systems	Web Portal Testbed Manager Testbed Directory Service Resource Explorer Tool	
Validated requirement	PT-DIR-S-005, PT-REE-T-001, TB-GEN-R002, TB-GEN-R-003, TB-GEN-R-004, TB-GEN-R-005, TB-GEN-R-006, TB-GEN-R-007, TB-GEN-R008, TB-GEN-R-009, TB-GEN-R-010, TB-GEN-R-011, TB-GEN-R012, TB-GEN-R013, TB-MAN-001, TB-MAN-007	

Step	Description	Status	Remarks
1	The Testbed Operator agrees with the RAWFIE platform Admin to connect its Testbed		
2	Testbed Operator ensures the testbed fulfill the needed requirements to be connected to the RAWFIE platform (Networking facilities, and so on)		
3	Testbed Operator fills the new Testbed information via Testbed Manager and inserts the testbed in the Master Data Repository using Testbed Directory Service		
4	Testbed Operator explores all testbeds and their details from Resource Explorer Tool		
5	Testbed Operator configures the Testbed components to be able to communicate with the rest of the RAWFIE platform		

Metric	Success criteria	Status	Remarks
PLATFORM / USE / 7 / NOTIFICATION	Questionnaire rates "notification" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 8 / ROLES	Questionnaire rates "roles" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 10 / VISUALISATION / SIMPLICITY	Questionnaire rates "simplicity" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 12 / VISUALISATION / UTILITY	Questionnaire rates "utility" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 13 / GUIDANCE	Questionnaire rates "guidance" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 14 / FILTERING	Questionnaire rates "filtering" with an average > 3.5 (1 to 5)		
TESTBED / DATA / 1 / INFORMATION	The information managed by the testbed		

	components is available		
PLATFORM / FUNC / 17 / EXTENSIBILITY	Connection of the new testbed did require the input of new data related only to the new testbed and its resources.		

6.4 UxV Manufacturers scenarios

6.4.1 Install new UxVs in a testbed

Table 96: "Install new UxVs in a testbed" validation scenario

Scenario ID: **UM-01**	Conducted by:	Date:
Title	Install new UxVs in a testbed	
Comment		
Main stakeholder	UxV Manufacturers	
Secondary stakeholder	Testbed Operator	
Involved Sub-systems	Web Portal Resource Explorer	
Validated requirement	PT-P-003, TB-G-004, UXV-MGT-006	

Step	**Description**	**Status**	**Remarks**
1	UxV Manufacturer ask the Testbed Operator if new UxVs could be installed in the testbed		
2	Testbed Operator agrees		
3	UxV Manufacturer sends the new UxVs to the testbed site		
4	UxV Manufacturer give the information about the UxVs to the Testbed Operator		
5	Testbed Operator update the resource description for its testbed via the Resource Explorer		
6	UxV Manufacturer ensures the UxV Node is able to send / receive information to/from the RAWFIE components through the foreseen software interfaces		
7	UxV Manufacturer and Testbed Operator configure the Testbed and RAWFIE platform components to control the new UxVs		

Metric	**Success criteria**	**Status**	**Remarks**
PLATFORM / FUNC / 17 / EXTENSIBILITY	Connection of the new UxV did require the input of new data related only to the new UxV.		
PLATFORM / USE / 7 / NOTIFICATION	Questionnaire rates "notification" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 8 / ROLES	Questionnaire rates "roles" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 10 / VISUALISATION / SIMPLICITY	Questionnaire rates "simplicity" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 12 / VISUALISATION / UTILITY	Questionnaire rates "utility" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 13 / GUIDANCE	Questionnaire rates "guidance" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 14 / FILTERING	Questionnaire rates "filtering"		

	with an average > 3.5 (1 to 5)		

6.4.2 Autonomous coordination of multiple UxVs

Table 97: "Autonomous coordination of multiple UxVs" validation scenario

Scenario ID: **UM-02**	Conducted by:	Date:
Title	Autonomous coordination of multiple UxVs	
Comment	This scenario deals with the autonomous coordination of multiple UxVs for providing the RAWFIE experiment with some robustness with respect to the loss of communication or performance issue in the connection between the UxV swarms and the RAWFIE system. This is particularly relevant for ensuring the UxV coordination when they are operating in large remote areas or over the sea.	
Main stakeholder	Testbed Manager, Experimenters	
Secondary stakeholder	UxV Manufacturers,	
Involved Sub-systems	Local RAWFIE entities Proximity component	
Validated requirement	TB-UVG-001, UXV-MGT-002, UXV-NET-002, UXV-NET-003, UXV-NET-004, UXV-NET-005, UXV-NET-006, UXV-NET-007, UXV-NET-008, UXV-NET-009, UXV-PRC-001, UXV-PRC-002, UXV-PRC-004, UXV-MGT-002, UXV-MGT-004, UXV-MGT-005, UXV-NOD-001	

Step	**Description**	**Status**	**Remarks**
1	The UxV manufacturer(s) deploys several UxVs that will operate in swarm in the experiment. The experiment consists in collecting and analysing the data exchanges that occurred during the experiment on the Proximity component network interface, for the sake of the coordination of the UxV motion.		
2	UxV Manufacturer sends the new UxVs to the testbed site. UxV Manufacturer gives the information about the UxVs to the Testbed Operator.		
3	Testbed Operator update the resource description for its testbed via the Resource Explorer, while the route followed by the UxV and relative UxV locations are specified in the experiment EDL script. UxV Manufacturer and Testbed Operator configure the testbed to control the new UxVs.		
4	The experiment is started and the experimental conditions, the exchanged data and the behavior of the UxV are logged with a time information.		
5	The UxV manufacturer collects the logged data and evaluates the relationship between the experimental conditions, the exchanged data and the behaviour of the UxV • View experiment log • Examine measurements • Percentage of the covered area • Nodes lifetime • Nodes energy consumption • Final positions		
6	The experimenter details the deviations of the UxV route and their relative trajectories from the expected behaviour.		
7			

Metric	**Success criteria**	**Status**	**Remarks**
PLATFORM / USE / 12 / VISUALISATION / UTILITY	Questionnaire rates "filtering" with an average > 3.5 (1 to 5)		

PLATFORM / USE / 13 / GUIDANCE	Questionnaire rates “filtering” with an average > 3.5 (1 to 5)		
PLATFORM / USE / 14 / FILTERING	Questionnaire rates “filtering” with an average > 3.5 (1 to 5)		
PLATFORM / USE / 15 / EXPERIMENTS STATISTICS	Questionnaire rates “filtering” with an average > 3.5 (1 to 5)		
UxV / FUNC / 1 / COHERENCE	Questionnaire rates “filtering” with an average > 3.5 (1 to 5)		
UxV / FUNC/ 2 / MISSION ACHIEVEMENT	Questionnaire rates “filtering” with an average > 3.5 (1 to 5)		
INTERCONNECTIVITY / PERF / 1 / AGGREGATED THROUGHPUT	Questionnaire rates “filtering” with an average > 3.5 (1 to 5)		

Table 98: "Test payload movement" validation scenario

Test ID: **UxV15**	Conducted by:	Date:	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration			
Software Configuration			
Test Name:	***Test payload movement (indoor or outdoor)***		
Preconditions	Requires the RAWFIE system to be operational (e.g. Resource controller reachable) Requires the mission to be defined and running. Requires the UxV to be ready to operating (e.g. en route). Requires the UxV to be reachable by any communication mean.		
Related Requirements	PT-EXA-T-008, PT-NAV-T-001, PT-NAV-T-002, PT-NAV-T-003, PT-VIS-T-001, TB-REC-001, TB-REC-004, UXV-NET-009, UXV-SEN-003, UXV-SEN-005, UXV-PRC-001, UXV-MGT-002 ,UXV-PRC-003,UXV-PRC-005, UXV-MGT-006, UXV-NOD-001,UXV-SEN-004, TB-UVG-001, UXV-NOD-003		
Tools Used			

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Establish the communication with the UxV	Communication established		
2	Establish a secure control session	Secured control session established		
3	Load on the robot (max payload). Send a reachable waypoint	Command received		
4	If the UxV is not autonomous, instruct it with the necessary waypoint or guidance information, possibly until the end of the test	Further optional instructions for returning home received, Confirmation of the UxV at home		
5	Close the secure control session.	The UxV is at the waypoint, the load is been transported. Connection closed		

Metric	**Success criteria**	**Status**	**Remarks**
PLATFORM / USE / 12 / VISUALISATION / UTILITY	Questionnaire rates "filtering" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 13 / GUIDANCE	Questionnaire rates "filtering" with an average > 3.5 (1 to 5)		
PLATFORM / USE / 14 / FILTERING	Questionnaire rates "filtering" with an average > 3.5 (1 to 5)		
UxV / FUNC / 1 / COHERENCE	Questionnaire rates "filtering" with an average > 3.5 (1 to 5)		

UxV / FUNC/ 2 / MISSION ACHIEVEMENT	Questionnaire rates "filtering" with an average > 3.5 (1 to 5)		
INTERCONNECTIVITY / PERF / 1 / AGGREGATED THROUGHPUT	Questionnaire rates "filtering" with an average > 3.5 (1 to 5)		

7 ANNEX 1. Validation scenario template

The templates for the validation scenarios and their evaluation metrics are described in the tables hereafter.

Table 99: Validation scenario: Scenario 1

Scenario ID:	Conducted by:		Date:	
Title				
Comment				
Validated requirement				
Technology	**Details**		**Status**	**Remarks**
Measurements	**Details**		**Status**	**Remarks**
Environment	**Details**		**Status**	**Remarks**
Algorithm	**Details**		**Status**	**Remarks**
Special script steps	**Details**		**Status**	**Remarks**
Metric		**Success criteria**	**Status**	**Remarks**

Scenario ID:	Conducted by:	Date:	
Title			
Comment			
Validated requirement			
Technology	**Details**	**Status**	**Remarks**
Measurements	**Details**	**Status**	**Remarks**
Environment	**Details**	**Status**	**Remarks**
Algorithm	**Details**	**Status**	**Remarks**
Special script steps	**Details**	**Status**	**Remarks**
Metric	**Success criteria**	**Status**	**Remarks**

Or

Scenario ID**: UD-01**	Conducted by**:**	Date**:**
Title		
Comment		
Main stakeholder		
Secondary stakeholder		
Involved Sub-systems		
Validated requirement		

Step	**Description**	**Status**	**Remarks**
1			
2			
3			
4			
5			
6			
7			

Metric	**Success criteria**	**Status**	**Remarks**

The validation scenario addressing a specific feature of function of the RAWFIE testbed are described in the tables hereafter.

Table 100: specific validation scenario: xxxx

Scenario ID:	Conducted by:		Date:	
Title				
Comment				
Validated requirement				
Technology	**Details**		**Status**	**Remarks**
Measurements	**Details**		**Status**	**Remarks**
Environment	**Details**		**Status**	**Remarks**
Algorithm	**Details**		**Status**	**Remarks**
Special script steps	**Details**		**Status**	**Remarks**
Metric		**Success criteria**	**Status**	**Remarks**

Table 101: Metrics and success criteria

Component, feature, function	Short description	Metrics	success criteria	comment

8 ANNEX 2. Component testing - how to read the verification scenarios

Even if at conceptual level in most of the case, the members of the consortium have identified all main system components, both hardware and software, and for each of them the template that will be described in the following will be adopted, in order to describe each required testing scenario.

We will assume that each component can consist of zero, one or more sub-components. If there are no sub-components the testing scenario is related to the component itself, whereas in the other cases it will be related to each sub-component.

Moreover, each component (or sub-component) can have one or more verification scenarios.

Two cases can be distinguished:

1. Only one test for the component (or sub-component). In this case the following table template is used:

Table 102: test for the component (or sub-component)

Test ID**:**	Conducted by**:**	Date**:**	Test Category**: Verification Tests (which tier?)**
Hardware Configuration			
Software Configuration			
Test Name:	***Name of the test***		
Preconditions	•		
Related Requirements	Requirement IDs from D3.2		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1				
2				
3				
4				

2. More tests for the component (or sub-component). In this case the following template is used:

A) first template describing the component and identifying the tests that will be performed on that component, by attributing a TEST ID at each test;

Table 103: tests that will be performed on a given component

Component (parent component if applicable)	Name of the component
Component Behaviour	If useful, please refer to the corresponding section in D4.1.
Tests	List of the test IDs All tests in this section… As in the following the test…

B) a second template describing the specific test and the expected results.

Table 104: specific test of a given component and the expected results

Test	Name of the test
Test ID	ID of the test
Component (parent component if applicable)	Name of the component/s involved in the verification test
Pre-requisites	Working condition for the component in order to be able to execute the test
Test description	Condition that should be verified; Sequence of steps to perform the test
Expected results	The expected results from the execution of the steps described.

9 ANNEX: Unreferenced Requirements

This table provides an overview of the unreferenced requirements of D3.3.

Table 105 – Unreferenced Requirements

No	ID	Component	Category	Title	Type	Priority	Reason
27	PT-ACC-S-007	Accounting Service	PLATFORM	The accounting service should be able to handle the addition of new services that may be incorporated in the RAWFIE platform during time.	FUNC	**MEDIUM**	Implementation specific for the given case. Not testable.

10 References

[FFF D6.1] Detailed specifications for first cycle ready, Fed4FIRE D6.1 deliverable, 2013. http://www.fed4fire.eu/fileadmin/documents/public_deliverables/D6-1_Fed4FIRE_Detailed_specifications_for_first_cycle_ready.pdf

[RAWFIE D3.2] Specifications and Analysis of RAWFIE Components Requirements, RAWFIE D3.2 deliverable, 2016.

[RAWFIE D3.3] Specifications and Analysis of RAWFIE Components Requirements, RAWFIE D3.3 deliverable, 2017.

[RAWFIE D4.5] Design and Specifications of RAWFIE Components, RAWFIE D4.5 deliverable, 2016.