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	D6.3: RAWFIE Operational Platform Testing and Integration Report		
Confidentiality	PU	Deliverable type[1]	R
Deliverable File	D6.3	Date	30.04.2016
Approval Status[2]	WP Leader, 1st Reviewer, 2nd Reviewer	Version	1.0
Contact Person	Philippe 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
Kakia Panagidi	UoA	kakiap@di.uoa.gr
Kostas Kolomvatsos	UoA	kostasks@di.uoa.gr
Konstantinos Kolomvatsos	UoA	kostasks@di.uoa.gr
Vasil Kumanov	Aberon	Vasil.kumanov@aberon.bg
Marcel Heckel	Fraunhofer	marcel.heckel@ivi.fraunhofer.de
Kiriakos Georgouleas	HAI	Georgouleas.Kiriakos@haicorp.com
Nikolaos Priggouris	HAI	PRIGGOURIS.Nikolaos@haicorp.com
Jason Ramapuran	HES-SO	jason-emmanuel.ramapuram@hesge.ch
Philippe Dallemagne	CSEM	Philippe.dallemagne@csem.ch
Damien Piguet	CSEM	Damien.Piguet@csem.ch
Giovanni Tusa	IES	g.tusa@iessolutions.eu
Miquel Cantero	ROBOTNIK	mcantero@robotnik.es
Ricardo Martins	MST	rasm@oceanscan-mst.com

REVIEWERS TABLE

Name	Company	E-Mail
Marcel Heckel	Fraunhofer	Marcel.Heckel@ivi.fraunhofer.de
Kakia Panagidi	UOA	kakiap@di.uoa.gr

DISTRIBUTION

Name / Role	Company	Level of confidentiality[3]	Type of deliverable
Consortium		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
0.1	01.03.2017	First draft	All
0.2	27.03.2017	Version for contribution integration	All
0.3	17.04.2017	Updated sections 3.4 and 3.5	Sections 3.4 and 3.5
0.4	28.04.2017	Version for internal review	All
1.0	30.04.2016	Final version	All

Abstract:

The objective of this deliverable is to report about the integration and testing of the RAWFIE system. It presents the status of the interface tests and the verification tests as well as of the integration results. It mentions the technicalities required for the consolidation of the RAWFIE components in a unified platform. The results obtained during the experimentations and the specific tests are analysed to identify and characterise the improvements and fixes to be brought to the prototype implementation during the third development iteration. The integration roadmap lists the enhancements of the RAWFIE operational platform based on the outcomes of the testing procedures and the deployments on the operational sites. The document is the second release over the three phases/cycles defined in the RAWFIE project.

This deliverable is based on the results of the tasks T6.1 and T6.2, on the work done in WP5, and on the verification tests planning presented in D4.6.

Keywords: Integration, **i**nterface tests, verification tests, roadmap

Table of Contents-

List of Figures

List of Tables

The following table gives the abbreviations used across the RAWFIE projects in the documents and deliverables.

Table 1: Common abbreviations

Abbreviation	Meaning
3D	three-dimensional space
ACL	Access Control List
AGL	Above Ground Level
AHRS	Attitude and Heading Reference System
AJAX	Asynchronous JavaScript and XML
AM	Aggregate Manager (of SFA)
AP	Access Point
API	Application Programming Interface
API	Application programming interface
AT	Aerial Testbed
AUV	Autonomous underwater vehicle
B-VLOS	Beyond Visual Line Of Sight
CA	Certification Authority
CAA	Civil Aviation Authority
CAO	Cognitive Adaptive Optimization
CBNR	Chemical Biological Nuclear Radiological
CEP	Circular Error Probability
CPU	Central Processing Unit
CSR	Certificate Signing Request
DETEC	Department of the Environment, Transport, Energy and Communication
DGCA	Directorate General of Civil Aviation
DoA	Description of Actions
EASA	European Aviation Safety Agency
EC	Experiment Controller
ECC	Error Correction Code
ECV	EDL Compiler & Validator
EDL	Experiment Description Language
EDL	Experiment Description Language
EER	Experiment and EDL Repository
EU	European Union
E-VLOS	Extended Visual Line Of Sight
EVS	Experiment Validation Service
FIRE	Future Internet Research & Experimentation
FOCA	Federal Office of Civil Aviation
FPS	Frames Per Second
FPV	First Person View
GAA	German Aviation Act
GIS	Geographic Information System
GNSS	Global Navigation Satellite System
GPIO	General Purpose Input/Output
GPS	Global Positioning System
GUI	Graphical user interface
HD	High Definition
HTTP	Hypertext Transfer Protocol
HW	Hardware

IAA	Irish Aviation Authority
IaaS	Infrastructure as a Service
IDE	Integrated Development Environment
IDE	integrated development environment
IFR	Instrument Flight Rules
IP	Internet Protocol
ISO	International Standards Organization
JDBC	Java Database Connectivity
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
KPI	Key Performance Indicator
LBL	Long Baseline
LDAP	Lightweight Directory Access Protocol
LS	Launching Service
MEMS	MicroElectroMechanical System
MM	Monitoring Manager
MSO	Multi Swarm Optimization
MT	Maritime Testbed
MOM	Message Oriented Middleware
MVC	Model View Controller
NAT	Network Address Translation
NC	Network Controller
NF	Non Functional
ODBC	Open Database Connectivity
OEDL	OMF EDL
OMF	cOntrol and Management Framework
OMF	Orbit Management Framework
OML	ORBIT Measurement Library
OS	Operating System
OTA	Over The Air
P2P	Point to Point
PSO	Particle Swarm Optimization
PTZ	Pan Tilt Zoom
RC	Resource Controller
RC	Resource Controller
RE	Requirement Engineering
REST	Representational state transfer
RIA	Research and Innovation Action
ROS	Robot Operating System
ROV	Remotely Operated Vehicle
RPA	Remotely Piloted Aircraft
RPAS	Remotely Piloted Aircraft System
RPS	Remotely Piloted Station
RSpec	SFA Resource Specification
SaaS	Software as a Service
SAML	Security Assertion Markup Language
SFA	Slice-based Federation Architecture
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Simple Query Language
SSO	Single-Sign-On
SVN	Apache Subversion
TM	Testbed Manager

TMS	Testbed Manager Suite
TP	Testbed Proxy
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
UI	User Interface
UML	Unified Modelling Language
USV	Unmanned Surface Vehicle
UUV	Unmanned Underwater Vehicle
UxV	Unmanned aerial/ground/surface/underwater Vehicle
VE	Visualization Engine
VT	Vehicular Testbed
VT	Visualization Tool
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Map Service
WPS	Web Processing Service
WSDL	Web Services Description Language
XMPP	Extensible Messaging and Presence Protocol

Table 2 gives the notations commonly used across the present document.

Table 2: Notations

Notation	Description
D*X.Y*	Deliverable *X.Y* from the DoW
MS*X*	Milestone *X* from the DoW
WP*X*	Work package *X* from the DoW
OC*X*	Open Call *X*
A*X.Y*	Activity number *Y* in Phase *X*
DL*X.Y*	Deadline number *Y* in Phase *X*
M*X*	Project month number *X*

A glossary is located at the end of this document in Annex, p. 126.

Part I: Executive Summary

The objective of this deliverable is to report about the results obtained during the tests of the interfaces of the RAWFIE components and of their integration into a unified and operational system. It presents the status of the interface tests and the verification tests as well as of the integration results, including the technicalities required for the consolidation of the RAWFIE system and the identified enhancements of the RAWFIE platform based on the aforementioned results. The integration roadmap mentions the target milestones and the enhancements of the RAWFIE operational platform based on the outcomes of the testing procedures and the deployments on the operational sites. The document is the second release over the three phases/cycles defined in the RAWFIE project.

The document is organised into 4 parts. The second part (Part II) is the main section, which is structured into two Chapters. Chapter 1 presents the scope of the document, some definitions and abbreviations together with the relation to other RAWFIE deliverables. Chapter 2 describes the various aspects of the integration and testing of the RAWFIE system. It describes the approach and methodology used for describing, performing and reporting the tests and integration verification. It is followed by the integration with external entities (mainly SFA), the integration setup and the results of the tests of the interface and the verification tests performed on the RAWFIE components and system. To make sure that the current RAWFIE system meets the basic performance requirements, a section presents the measured performance of the kafka message bus in a typical setup. A conclusion is drawn in Part III, combined with a roadmap based on the results obtained from the previous steps and the subsequent modifications and improvements.

Part II: Main Section

1 Introduction

1.1 Scope of D6.3

The scope of this document is to present the results of the tests of the operational platform and the status of the component integration.

1.2 Definitions

This document makes use of a number of specific terms, which the RAWFIE team understands as defined below:

- **Verification** of a system is the task of determining that the system is built according to its specifications (functionalities according to requirements and design specifications);
- **Validation** is the process of determining that the system actually fulfils the purpose for which it was intended (according to the specification);
- **Evaluation** reflects the acceptance of the system by the end users and its performance in the field, which eventually translates into usefulness (always according to user needs and / or performances in the field against realistic scenarios).

1.3 Relation to other deliverables

The work performed in WP6 relies on the outcome of WP3 and WP4, as well as on WP5 activities, which performed the development and integration of components, according to the roadmap described in D2.2.

D6.3 is an update of D6.1. From a programmatic point of view, it provides a feedback to WP5 (based on the results of the integration tests to be taken into account in D5.3 and D5.4) for revisiting and improving the implementation of components and their interaction in the global architecture. WP3 exploits these results as well to identify any required revision or extension of the defined requirements. Finally, WP4 may review and revise the architecture in subsequent iterations in light of the WP6 outcome.

D6.3 refers to D4.7 for many aspects, including the architectural concepts, the data model, the components interactions, etc. The testing of the components interfaces and their integration is based on the architecture and design deliverables of WP4, and specifically on the verification scenarios and planning presented in deliverable D4.6, with some modifications that will be highlighted in the rest of the document.

In spite of its coarse granularity, D2.2 forms the basis for checking the completeness of D6.3 coverage. D2.2 specifies the different rounds of development and the objectives in terms of function, environment, etc. which directly defines the boundaries of the prototype integration or related tasks (see sections 3.3 to 3.10). D6.3 reports on the integration steps and the verification of components once combined with the rest of the RAWFIE system, before the submission of this system to the validation process.

D6.3 refers explicitly to the Verification scenarios defined in D4.3 (section 5.1) for the component testing at a high level, which gives emphasis to the integration process and therefore on the interfaces, dependencies and interactions between components. D6.3 reflects the actual emphasis of the integration process on the interfaces, dependencies and interactions between components. D6.3 deals with, and presents, the interface testing results and the high-level testing results, according to verification templates found in D4.6.

2 Integration & Testing

2.1 Approach

The objective of the Integration & Testing activities, whose results are presented in D6.3, is to produce the second version of the end-to-end operational prototype of the RAWFIE platform. Following the time-plan defined for Phase 2 of the Integration & Testing roadmap (D2.2), the results reported in this deliverable reflect the integration and testing work carried out by project's partners during the 2nd technical iteration.

Since the approach does not substantially differ from what described in deliverable D6.1 (Integration & Testing during the 1st iteration), the reader is also invited to refer to Section 2 of the same deliverable for further details.

As a result of the 1st Integration & Testing iteration, some suggestions for modifications and improvements to RAWFIE components and interfaces were derived. These suggestions, together with the outcomes of the implementation activities from WP5, and the second version of the requirements from D3.2, have triggered modifications and improvements in the design of components' functionalities and interfaces, being used as inputs for the second version of the RAWFIE architecture (D4.4) and components' specification (D4.5). In turn, the new version of the components' design, was used for defining new interface tests and verification scenarios, or for updating the existing ones in D4.6. D4.6 is therefore the main reference document for the integration and verification tests reported in D6.3.

2.2 Methodology

Integration testing includes activities where the different software components of the system are combined and tested as a group, to verify both the communication interfaces and end-to-end workflows and functionalities. The reader is invited to refer also to D6.1, Section 2, where further details of the methodology are explained. Here we highlight that, for the purposes of integration testing, the following tests categories are considered in the integration and verification plan (D4.6) and, as a consequence, in the present deliverable:

- **Testing of components interfaces:** this kind of tests are performed for all implemented components that provide a software interface to other components (via a REST or SOAP / RPC API) or are capable to send/receive data from Message Bus. As an example of the communication interfaces that need to be verified during system components' integration, following **Figure 1** and **Figure 2**, taken from the D4.5,

provide an overview of the several interactions (through different communication technologies) between Frontend Tier components and Middle Tier components, and between Middle Tier components and other system components, respectively.

- **Execution/Testing of verification scenarios**: This involves the execution of all the verification scenarios defined in D4.6, Section 6.1, and can comprise tests whose aim is mainly to verify individual components' functionality – although in most cases they have as prerequisite the existence of other components – as well as end to end scenarios, where several system components are involved

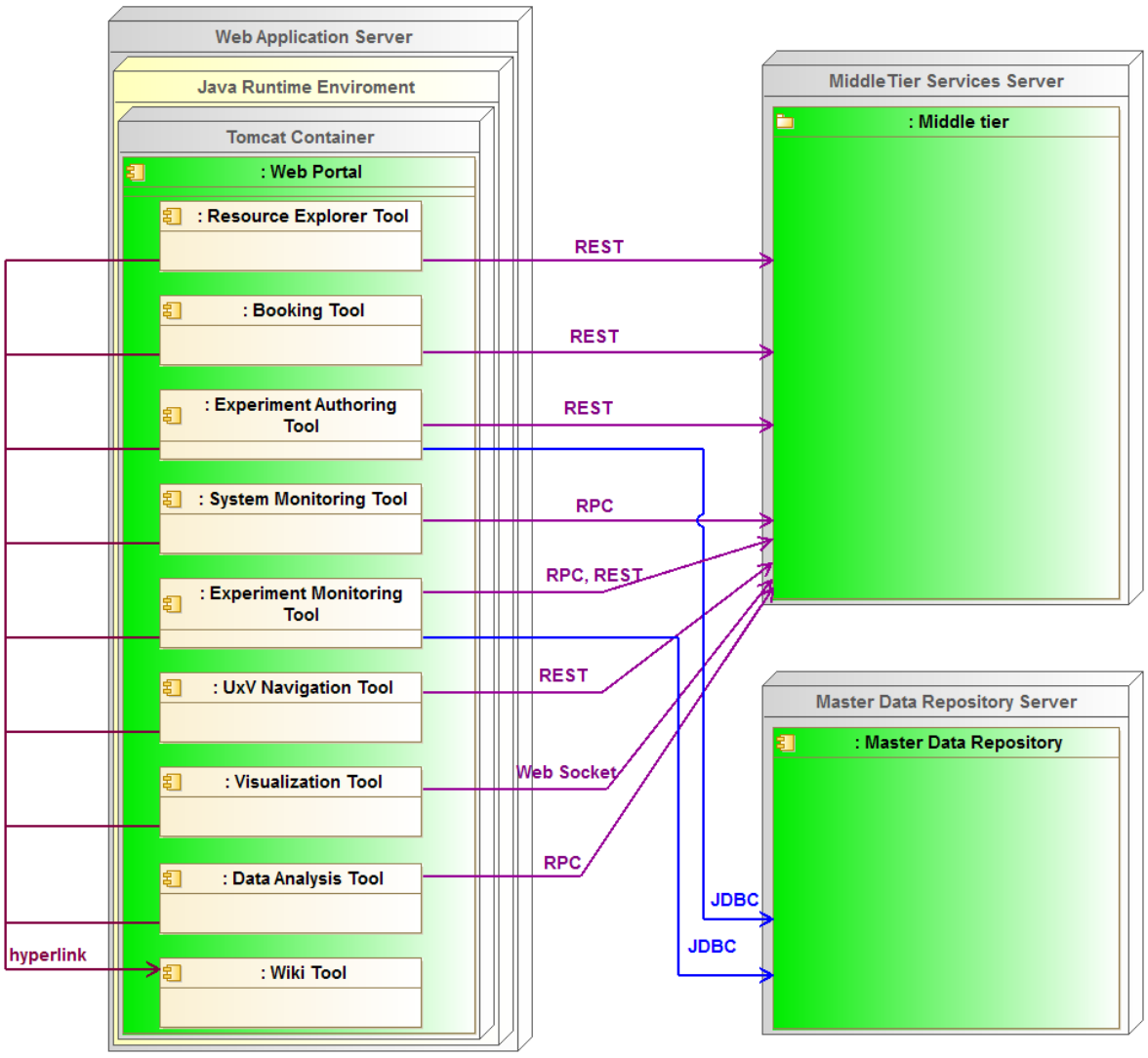


Figure 1: Overview of software interfaces provided by Middle Tier Services and the Master Database, and used by Frontend Tier modules

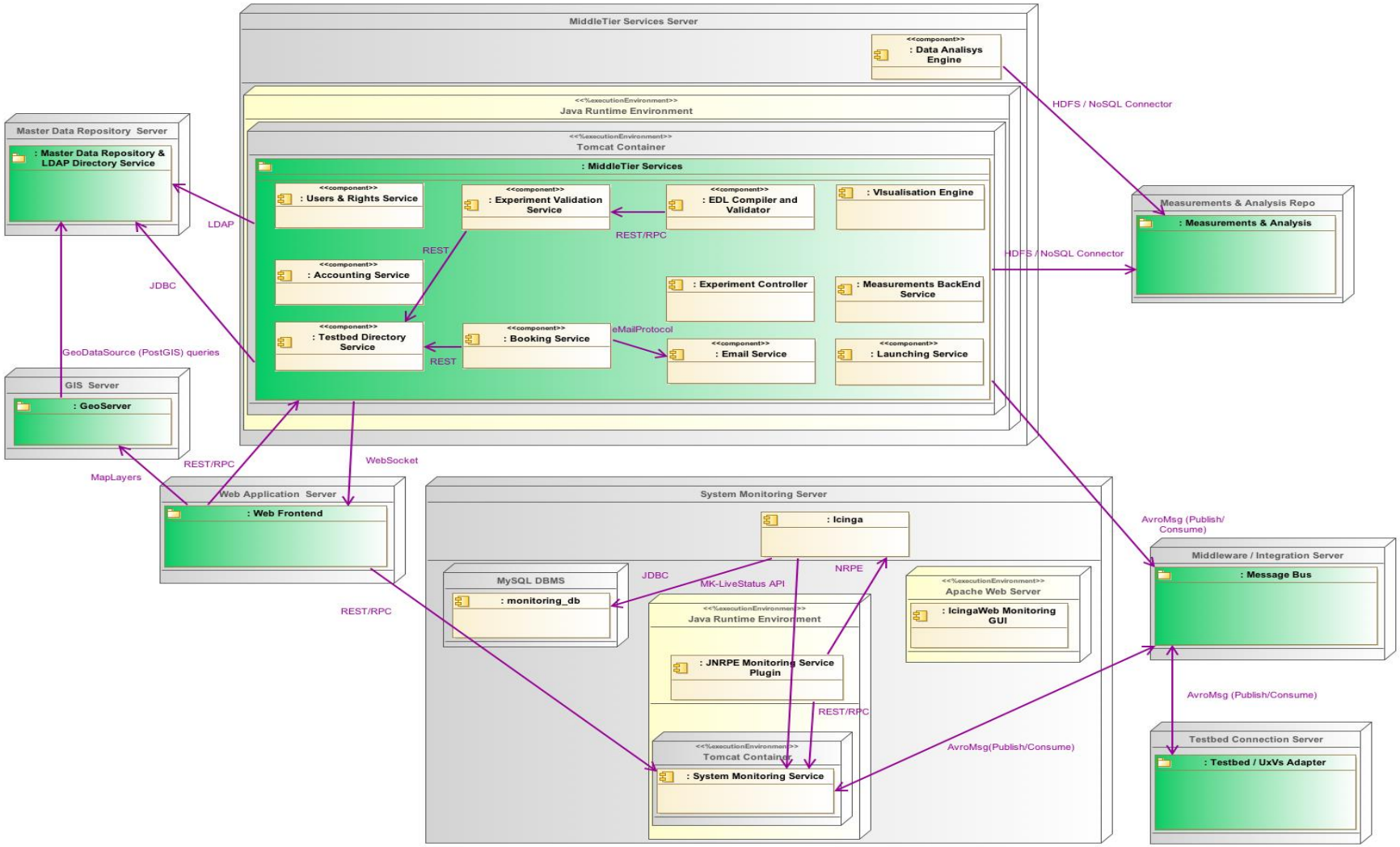


Figure 2: Overview of software interfaces between Middle Tier components, and between Middle Tier components and other system components

2.2.1 Tests reporting format

Results of the verification tests are reported using two different reporting templates, for interfaces testing and for the verification scenarios, respectively. These templates are described in Section 2.2.1 of deliverable D6.1.

2.3 Integration of external components

The integration of new tools and services for the extension of the experimentation capabilities, is easily realised thanks to the open architecture of RAWFIE, based on a mix of SOA principles (therefore the availability of RPC and REST API), and the decoupling of components through the asynchronous communication via the Message Bus.

2.3.1 Interoperability with external SFA clients through the SFA Aggregation Manager

From the technical standpoint, interoperability with external SFA clients is realised through the implementation of a modified version of the SFA Aggregation Manager (AM) at Testbed level, and its integration with existing RAWFIE components. The modified SFA Aggregation Manager is provided in the context of the SAM proposal, who joined the project after the 1st Open Call. It is therefore part of the SAM software module, which will be deployed on each connected Testbed in order to handle, among the others, the reservation process of the respective resources. Please also refer to D4.4, D4.5, and D4.7 for more details about the components and functionalities of SAM software module.

The following are the main integration scenarios that will be realised the SFA principles:

- **Adding/Editing/Deleting of resources**. This action will always be performed through the Testbed Manager admin UI. In this scenario the RAWFIE Testbed Manager component will act as the gateway to the SFA Aggregation Manager, since it will forward the modification requests to both the SFA Aggregation Manager using the provided REST API (for updating the local Triple Store DB), and to the Testbed Directory Service through its REST API, for updating the same information in the centralised Master Data Repository of RAWFIE
- **Listing / searching of resources**. This action can be performed through the RAWFIE platform as well as through external SFA enabled clients / GUI (e.g. MySlice). In the former case, the RAWFIE Resource Explorer Tool and, in turn, the Testbed Directory Service components will be used to search and visualise all or specific UxV resources in the given Testbed. In the latter case, external SFA clients will directly call the SFA Aggregation Manager through the provided REST API. The SFA AM will in turn perform semantic queries to the local Triple Store DB.
- **Booking requests.** This action can be performed through the RAWFIE platform as well as through external SFA enabled clients / GUI (e.g. MySlice). In the former case, the RAWFIE Booking Tool will forward the booking request, through the Booking

Service, to the SFA Aggregation Manager using the provided REST API and to the RAWFIE Master Data Repository, so that all repositories will be synchronised. In the latter case, external SFA clients will directly call the SFA Aggregation Manager through the provided REST API, and the SFA AM will in turn perform the booking of resources in the local Triple Store DB. The Booking Service will also periodically synchronise itself with the SFA Aggregation Manager, in order to ensure consistency between the reservations made using the SFA interface (and therefore the content of the Triple Store DB), and the ones made using the RAWFIE Booking Tool (Master Data Repository).

2.3.2 Feedback from professional stakeholders

The RAWFIE system will be deployed in several sites, in which professional stakeholders will take an active part. The RAWFIE technical team expects to get some feedback on various topics during the deployments and exploitation of the RAWFIE system. This includes the ease of deployment, the ease and efficiency in the documentation of all these aspects, the edition of EDL scripts, the execution of experiments, the support given by the team, the exploitation of the data collected during the experiment, including the experience gained by the various stakeholders, etc. RAWFIE hereafter plans to implement a methodology, in which RAWFIE should give a possibility for the professional stakeholders to give a feedback about the use and impact of RAWFIE.

The RAWFIE technical team, in charge of the design, the prototyping and the validation of the RAWFIE system and components will survey their deployments of the operational use through several channels. Since the initial deployments will be done under the guidance of RAWFIE experts (typically some members of the technical teams), the feedback will be obtained directly, during debriefing meetings and from written reports. The outcome of the initial deployments will help in defining the scope and processes of the professional stakeholders, in particular the support teams.

Beyond their own experience gathered during the initial deployments and operations of the RAWFIE prototypes, one of the most important channels for the validation is the feedback from the professional stakeholders that are engaged in the lifecycle of the RAWFIE system. Other stakeholders (e.g. testbed owners) will perform most of the next deployments, possibly with the remote or on-site help of commercial support teams (see WP2 for the details about the commercial exploitation of RAWFIE). The tools will still consists in debriefing meetings organised during and after the deployments, combined with electronic questionnaire distributed at specific points in time, corresponding to important points in the system lifecycle, such as specification, installation, deployment tests, education, experiment development and execution, result exploitation, etc. They can be complemented by the use of feedback forms and statistics about the usage and the performance, directly integrated into the RAWFIE system.

2.3.3 Integration of RAWFIE "newcomers"

Many companies submitted proposals to Open calls, aiming at integrating their resources (considered until then as external components) into the RAWFIE environment. RAWFIE addresses the possibility of connecting new Testbeds to the core RAWFIE platform, and adding new resources (UxVs) to already connected Testbeds. From the methodology standpoint, newcomers joining the consortium through the open calls, are provided with all the needed information before and during the submission of the proposals. After joining, they are supported by RAWFIE partners during the integration process, through the organisation of meetings and training events, and in the technical activities described below. The first outcome of such integration process is mentioned in section 2.3.3.

2.4 Integration environment

This section describes the environment (depicted in Figure 3) used for the integration of the RAWFIE components and sub-systems and the subsequent testing. This may include the information, communication and computing infrastructure (servers, networks, etc.), the configuration (component settings, credentials, etc.) and data repositories, the testbeds used for testing and all other external services.

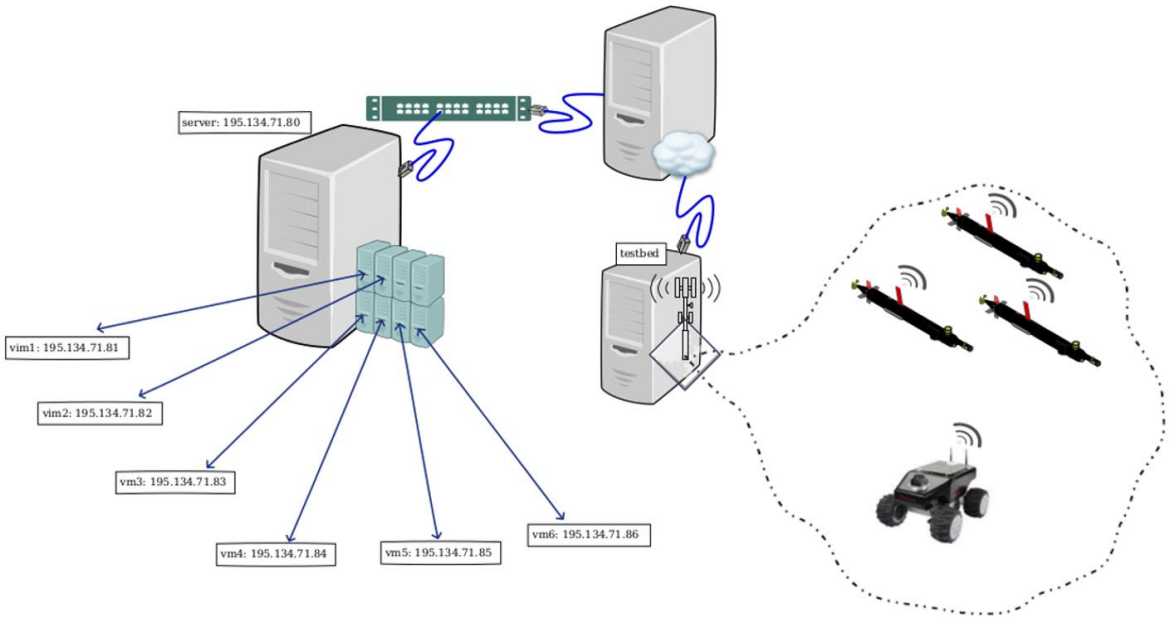


Figure 3: 1st RAWFIE environment integration

2.4.1 ICT infrastructure

For integration purposes clones for the development infrastructure described in D5.3 are created. An identical environment, illustrated in Figure 4, was created for facilitating continuous integration and resolving of errors. The messages from the online platform (production environment) are mirrored to the development environment in order to all services can be tested with real data. The mirrored environment is used for updates in coding and upgrading the services without affecting the rest of the infrastructure and when a service is stable enough is moved to the online platform.

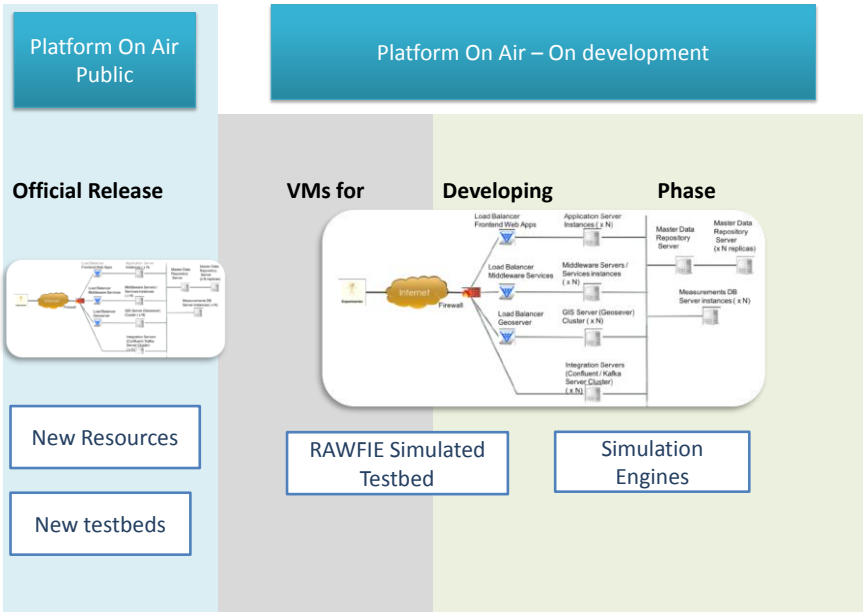


Figure 4: RAWFIE clones for the development infrastructure

According to the DoA, the first Milestone related to the development cycles was defined in M18 on which the 1st release of the platform was released. In order to outline a structured development process while maximizing the productivity and reducing possible bugs (that could be exposed to the experimenters), the RAWFIE consortium agreed in the creation of two identical environments: production and development. The production environment is the online platform that external users and experimenters can reach the RAWFIE functionalities via Internet. The development environment consists of the same devices being used for updates in coding and upgrading the services without affecting the rest of the infrastructure. An application or a service is moved to the production environment when it is stable enough.

2.4.2 Data repositories

The data model defined in D4.7 can be broken down into four major components:

1. Persistent Storage of Message BUS / Measurements DB: This will be done by Kafka Connect duplicating all messages on the BUS to HBASE (which is in turn backed by Hadoop).
2. Analysis Results DB: This database will contain the results for the data analysis tasks and is currently backed by a time series database called Whisper.
3. Master Data DB: This will house traditional SQL type data and is currently implemented by PostgreSQL.
4. Users & Rights Repository: uses a LDAP repository, as LDAP is a de facto standard for user management. It stores all user related data (name, organisation, address, password) and group memberships (roles based access control). The selected implementation is OpenDJ.

2.4.3 Tools & techniques for integration

RAWFIE uses a number of collaboration tools providing an integration friendly environment for development and deployment, such as Git, Docker and Redmine (see Figure 1). Hadoop and Hbase can also be considered as the connectors between the messages and the data storage of experimenters, which provides an efficient decoupling that is convenient for integration.

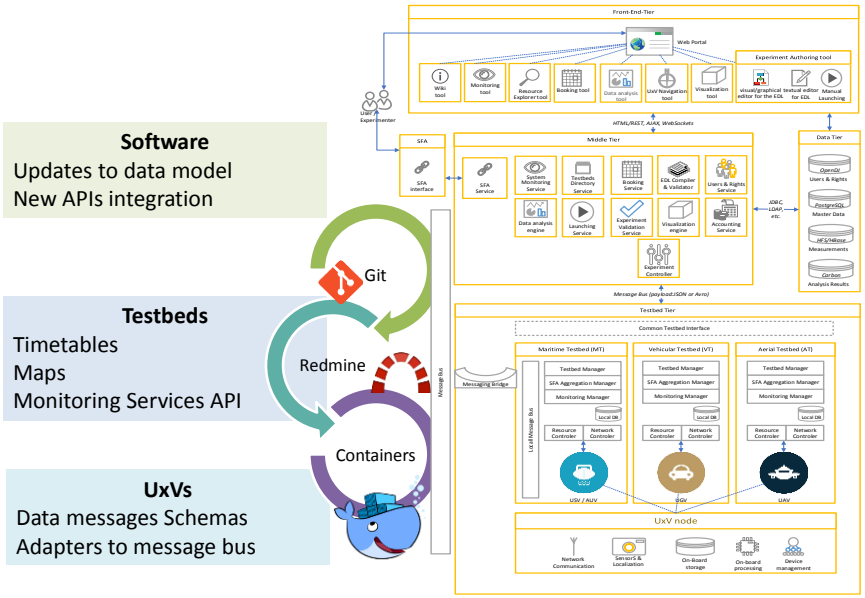


Figure 5: Tools for integration

Several tools are being used in order to facilitate continuous reporting and the integration of the software tools in a common environment. A Git platform was installed with Gitlab environment for all partners can work concurrently by using branching. All software is uploaded in project forms and then partners create branches for their specific features.

Another features that is used for the integration is the creation of machine image boxes in order to provide to testbed operators “black boxes” with the RAWFIE required services pre-installed and pre-configures. RAWFIE components are installed in Vagrant image boxes, which are used for quick deployment of the RAWFIE system by the developers and testers.

2.4.4 Message Bus

The message bus is an essential integration tool. RAWFIE uses the Kafka message bus for interconnecting the components, for data exchange, ordering and persistency, for reliability and robustness, etc.

The Kafka mirroring feature is used for creating the replica of an existing cluster, for example, for the replication of an active data centre into a passive data centre. Kafka provides a mirror maker tool for mirroring the source cluster into target cluster. This feature is used to allow for the replication of an exploitation environment to a site dedicated to development, test or maintenance.

The following diagram depicts the mirroring tool placement in architectural form:

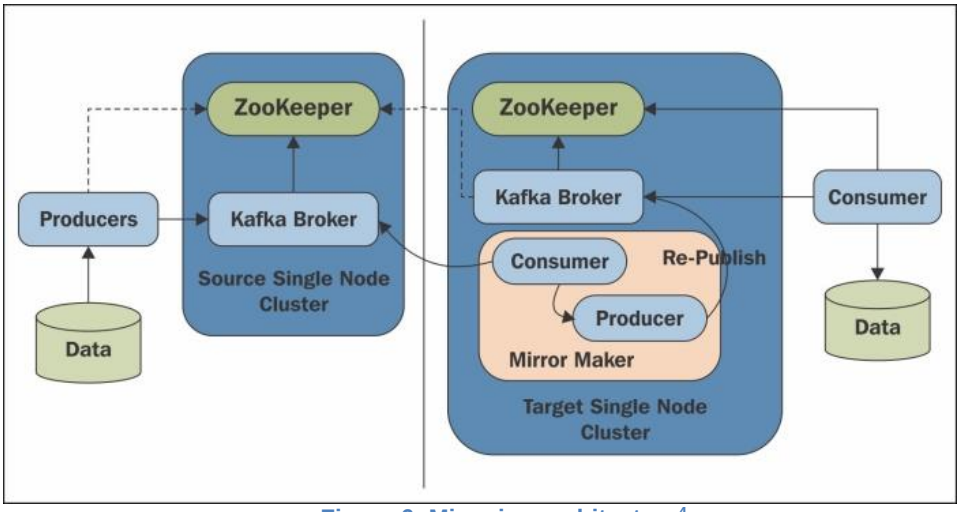


Figure 6: Mirroring architecture[4]

In contrast of replication processes, mirroring provides duplication of data across the testbeds. The advantages of mirroring are multiple like single connections down, clients connection/session times longer (depending on the location of the testbeds), legislation (some data can be collected in a country while some other data should not).

2.4.5 Integration of new UxVs

D4.4 and D4.5 provide technical guidelines for new UxVs integration in the platform. As specified in D4.5, UxV providers need to implement an "UxV Node" software module. This module is the software adaptor for RAWFIE, which will make the integrated UxV able to send measurements data, and to receive information and commands in standard format, mainly as JSON messages based on AVRO schemas. The RAWFIE "UxV Node" module also implements Apache Kafka Publishers and Consumers software, for the communication with other RAWFIE components.

2.4.6 Integration of new Testbeds

Besides providing the needed equipment for network connectivity, Testbeds owners need to deploy on premises the following RAWFIE software components:

- At least two local **Apache Kafka message bus servers**, for redundancy and high availability: these nodes realise the communication of the UxVs in the given Testbed, with other RAWFIE components
- **Testbed Manager**: provides the software interface to store UxVs related information to the Local DB, to the Master Data Repository through the Testbed Directory Service and to the Triple Store DB through the SFA Aggregate Manager (see D4.4, D4.5, D4.7 for detailed information on the design and interactions of these components)
- **Triple Store DB and SFA Aggregate Manager**: the SFA AM provides, through a REST API, advertising functionalities based on semantic searches on the local Triple

[4] https://www.packtpub.com/mapt/book/big_data_and_business_intelligence/9781782167938/4/ch04lvl1sec20/cluster-mirroring-in-kafka

Store. The same REST API is used for editing or adding new resources, to store local resources (UxVs) information in the Triple Store DB

- **Resource Controller** (optional): provides resources controlling capabilities according to custom algorithms developed within the RAWFIE project
- **Monitoring Manager**: provides Testbed side connection to the System Monitoring services and the related Frontend tools.

These elements are distributed using Vagrant virtual machines, as described in section 2.4.5.

Several Vagrant[5] virtual machine image boxes provide testebed operators with an environment bundled with all the RAWFIE components and the required software for these components to function properly. These images include all the testbed services, such as the Testbed Manager, the Resource Controller, the message bus broker, etc.

The distribution of these boxes to our testbed operators has two main benefits. First, we save time from building from scratch every time the required software environment to perform tests. Secondly, the distribution of ready-to-go images ensures that there will be no problems to our testers, due to software incompatibilities. In addition, with every upcoming upgrade to the RAWFIE components everything will continue to work properly.

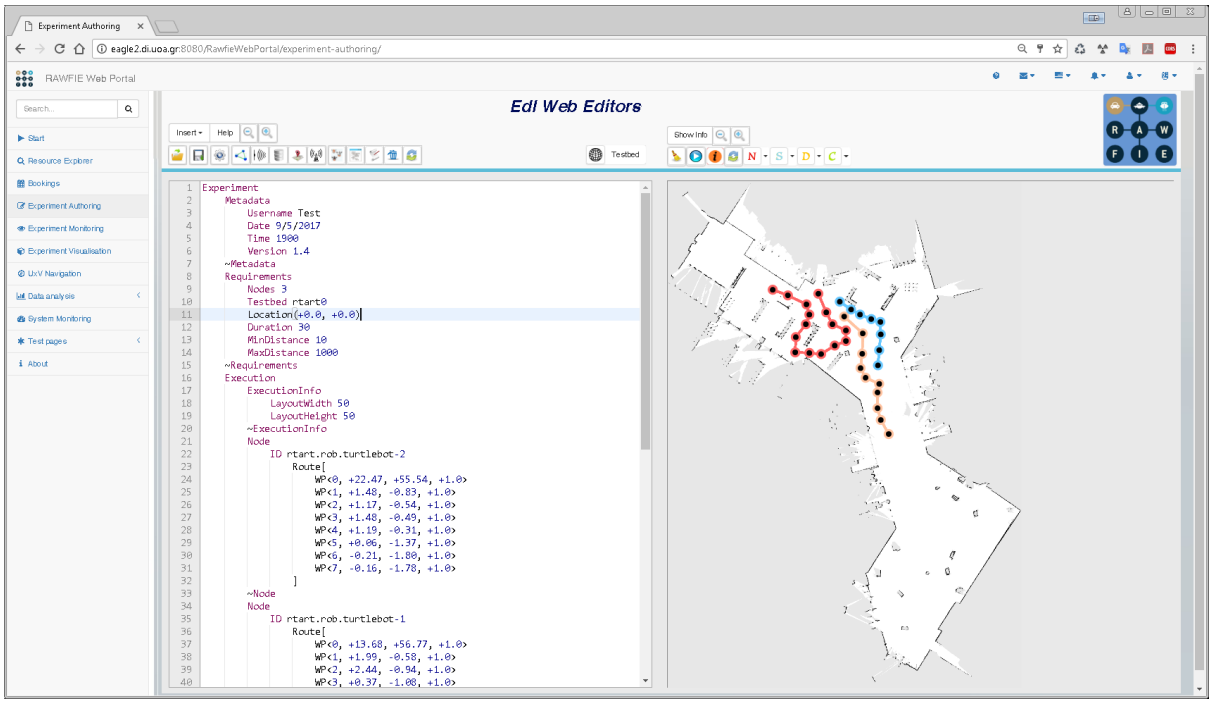
The process to integrate devices and testbeds in RAWFIE platform is straightforward:

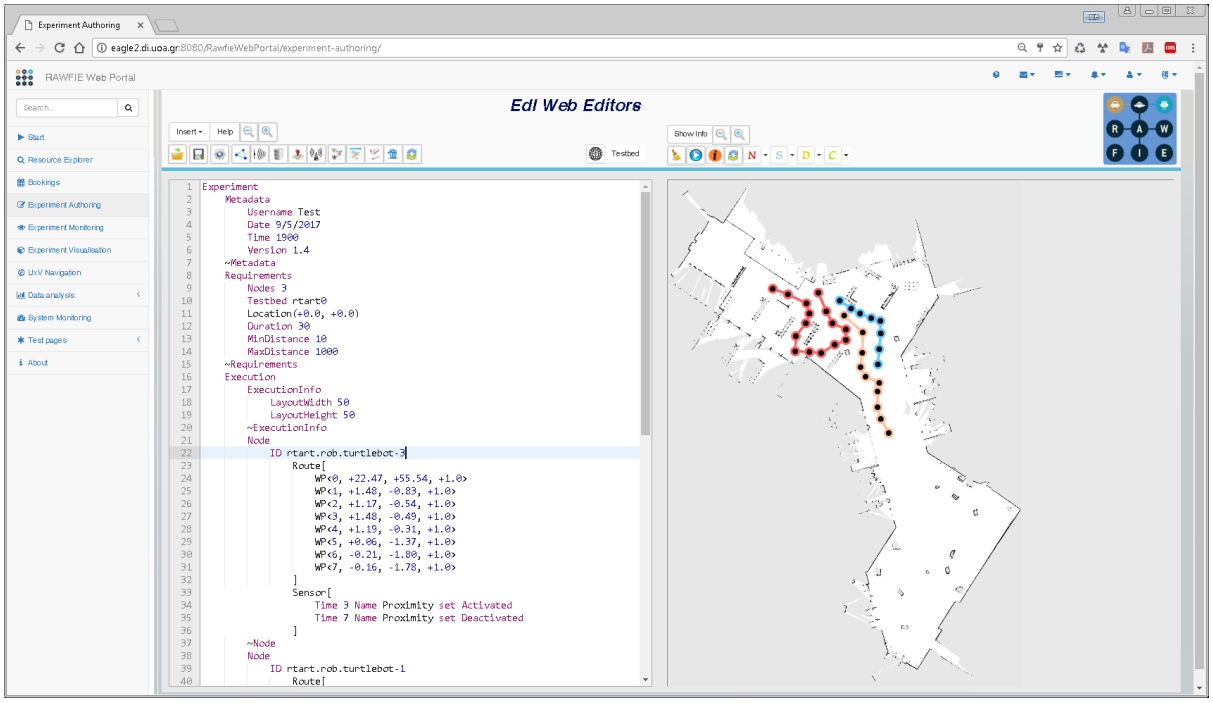
1. Testbeds provide information registered in RAWFIE database like location, name of the testbed, polygon of area or indoor map (if the testbeds is indoor)
2. RAWFIE provides to testbed operator a VM for being installed in a local server
3. VPN certificates created for the testbed and VPN connection
4. Testbed operator registers via Testbed Manager the devices in database
5. Trainings for the devices delivered in testbed
6. Testbed is up and running

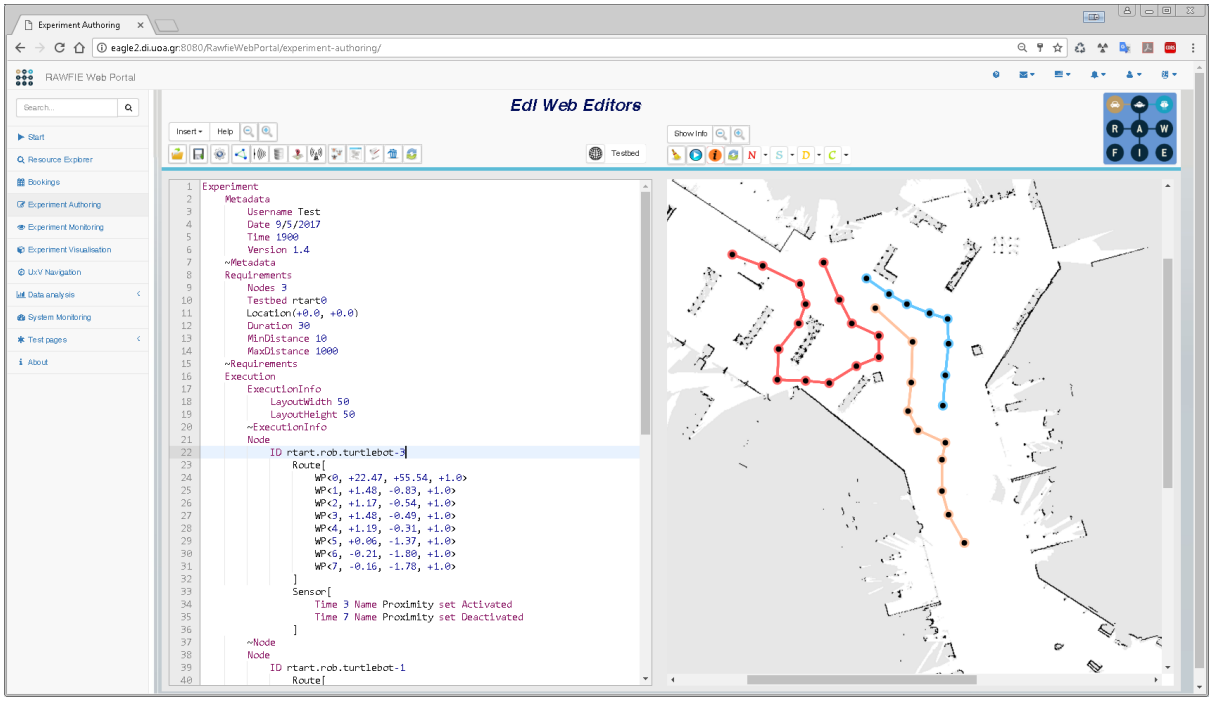
Although the delivery of the devices to testbeds coming from 1st Open Call is ongoing, some testbeds have started the integration process to the RAWFIE platform. The first testbed ready for the integration was an indoor testbed providing experiments for UGVs in several rooms. Starting from the kick off meeting in Athens for the Open Calls 1 people from the University of Zaragoza provided an infrastructure of monitoring of the possible area of experiments. The Wi-Fi coverage was established and tested to all the areas. The next thing was the installation of a local RAWFIE server. The credentials for the VPN network was sent to the testbed and a Virtual image of machine embedding of the required aforementioned services was sent to the testbed. The indoor maps were created by a lidar-embedded sensor on the devices and sent for their integration to RAWFIE geoserver in order to be used by the Experiment Authoring Tool and the Visualization tool. The devices were compatible with the Message bus by implementing a kafka consumer and producer and work in the VPN network. The integration

[5] https://www.vagrantup.com/

was fulfilled with the training of the manufacturer to the testbed owners for the devices functionalities.







2.5 Results of the Integration Test

This section provides an overview of the interfaces developed between the various SW modules developed within RAWFIE. It includes front-end components as well as modules implemented at middle tier, testbed and UxV level. The table below provides additional information about the type of interface that exists between two components. The level of implementation/testing is depicted with appropriate colouring and represents the situation at the end of the second iteration of development.

In Table 1 each cell represents an interface that was tested. This cell is used by the two components at the cross lines: each client component, or caller of one or many services interfaces, is represented in the rows, while the called component or service interface/s is represented in the columns.

Table 1: interface interaction matrix

	Web Portal	Wiki	Resource Explorer Tool	Booking Tool	Experiment Authoring Tool	Experiment Monitoring Tool	System Monitoring Tool	UxV Navigation Tool	Visualization Tool	Data Analysis Tool	EDL Compiler & Validator	Experiment Validation Service	Users & Rights Service	Booking Service	Launching Service	Experiment Controller	Data Analysis Engine	System Monitoring Service	Testbeds Directory Service	Accounting Service	Visualization Engine	Master Data Repository	Users & Rights Repository	Measurements Repository	Results Repository	Testbed Manager	Monitoring Manager	Network Controller	Resource Controller	Aggregate Manager (SFA)	UxV node	UxV Proximity	UxV - Network communication	UxV – Sensors & Localization	UxV – On board storage	UxV – On board processing	UxV – Device management	Schema Registry
Web Portal													R										L														
Wiki													L																								
Resource Explorer Tool				R														R																		
Booking Tool													R	R								O														
Experiment Authoring Tool											O	O			R							J														
Experiment Monitoring Tool															R			R				O														
System Monitoring Tool																		R																		
UxV Navigation Tool																														M		M	M				
Visualization Tool																				O										M		M	M		M		
Data Analysis Tool																M,R								R												
EDL Compiler & Validator												O									J															
Experiment Validation Service																					J															
Users & Rights Service																					O	L														
Booking Service													R								O															
Launching Service												R	R			M-p					O							M-p								
Experiment Controller														M-c						M-p	O				M-p			M-p							
Data Analysis Engine										R													O	R,O												R
System Monitoring Service																																				
Testbeds Directory Service																					O															
Accounting Service																					O															
Visualization Engine																M-c					J			J						M-c						
Master Data Repository																				J																
Users & Rights Repository																																				
Measurements Repository																																				
Results Repository																				J																
Testbed Manager																M-c		M-p										M-c								
Monitoring Manager																														M-c						
Network Controller																									M-p			M				M-c				
Resource Controller																M-c									M-p					M		M	M			
Aggregate Manager (SFA)																																				
UxV node								M	M								M			M-p						M-p		M			I	I	I	I	I	I	M
UxV Proximity																M									M					I		I		I	I	I	
UxV - Network communication								M	M								M								M			M		I	I						
UxV – Sensors & Localization								M	M								M											M		I							
UxV – On board storage																														I	I						
UxV – On board processing									M																					I	I						
UxV – Device management																														I	I						
Schema Registry																																				

MessageBus	M
Rest	R
SOAP	S
UxV internal	I
JDBC or JPA	J
Other	O
Success	
Partial Success	
Fail	
Not Tested	
Not applicable	

Table 2 -Interface types used in interface testing

Type	Description
M-c	Message bus consumer (receives messages from the message bus)
M-p	Message bus producer (sends messages to the message bus)
REST or R	REST (via HTTP) web service
SOAP or S	SOAP web service
LDPA or L	LDPA
JDBC or J	JDBC
JPA	Java Persistence API
I	UxV internal: UxV OS dependent

Note: For interface of type M-p, a related component is not included (or only "Message Bus" is mentioned). This is for example the case when the component acts as producer. The rationale behind this is that the producer of an Avro message just sends to the Bus agnostic of which will receive it. This message may be received by multiple consumers and this interaction will be depicted in the interface table of each receiver component including information for the exact producer. Therefore, there is no need to replicate this for the producer by including several similar rows.

2.5.1 Front-end integration

In the front-end tier, the integration activities included:

- Integration of *User and Rights Service* with the Web Portal as the main authorization mechanism for gaining access to the RAWFIE platform
- The following tools were integrated and become accessible via the web portal:
  - Wiki Tool
  - Resource Explorer Tool
  - Booking Tool
  - Experiment Authoring Tool
  - Experiment Monitoring Tool
  - System Monitor Tool
  - Visualisation Tool
  - Data Analysis Tool

Details on the interface testing activities performed for each front-end tool mentioned above are provided in the tables that follow:

Table 3: Test of the Web portal interfaces

Component***: Web Portal***	Conducted by**: Fraunhofer**	Date**: Feb 2017**	Test Category**: Interface testing**
Preconditions	Users are entered in the User & Rights Repository Wiki Tool has some help pages		

Related Component		Type[7]	Message or API Call	Status	Remarks/comments
1	User & Rights Repository	LDAP	Lookup	Success	Lookup user with the given password from the login page worked
2	Wiki Tool	Other	HTTP open web page	Success	Open web page in the Wiki Tool containing help for the current page.

Table 4: Test of the Wiki Tool interfaces

Component***: Wiki Tool***		Conducted by**: Fraunhofer**		Date**: Feb 2017**		Test Category**: Interface testing**
Preconditions		Users are entered in the User & Rights Repository				
Related Component		**Type**	**Message or API Call**	**Status**		**Remarks/comments**
1	User & Rights Repository	LDAP	Lookup	Success		Lookup user with the given password from the login page worked

Table 5: Test of the Resource explorer interfaces

Component***: Resource Explorer***		Conducted by**: Fraunhofer**		Date**: Feb 2017**	Test Category**: Interface testing**
Preconditions		Resources are entered in the Master Data repository			
Related Component		**Type**	**Message or API Call**	**Status**	**Remarks/comments**
1	Testbeds Directory Service	REST	searchResource	Success	Search resource by resource id passig a JSON in input
2			getAllResources	Success	Got all resources/UxVs
3			searchTestbed	Success	Search testbed by testbed id passing a JSON in input
4			getAllTestbeds	Success	Got all testbeds
5			getResources	Success	Got all resources/UxVs for a specific testbed id passing a JSON in input
6			testbed/identifier//{id}	Success	Testbed by testbed id
7			testbed/name/{name}	Success	Testbed by testbed name
8			testbeds?param1=value1 &param2=value2&param3= value3	Success	Testbeds by search parameters
9			resource/identifier/{id}	Success	Resource by resource id
10			resource/name/{name}	Success	Resource by resource name
11			resources?param1=value1 &param2=value2&param3= value3&param4=value4	Success	Resources by search parameters
12			testbeds/uav	Success	Testbeds supporting UAV
13			testbeds/usv	Success	Testbeds supporting UGV
14			testbeds/ugv	Success	Testbeds supporting USV
15			Testbeds/auv	Success	Testbeds supporting AUV
5	Booking Tool	HTTP	Redirect to page	Not tested	Booking Tool does not support a direct linkt to book a resource.

[7] Type refers to how the component interacts/interfaces with related component. For example if the component produces a message intended to be received by the related component the type should be M-p (acts as producer) while if it consumes a message type should be M-c.

Table 6: Test of the Booking Tool interfaces

Component: ***Booking Tool***		Conducted by:		Date: **Feb 2017**		Test Category: **interface testing**
Preconditions		• User must be logged in • UxV resources must be present in a testbed and advertised to the platform (browsable by the resource explorer tool) • Booking Service must be up and running • User & Rights Service must be up and running				
	Related Component	**Type**	**Message or API Call**	**Status**	**Remarks/comments**	
1	Booking Service	R	addReservation	Success		
2		R	editReservation	Success		
3		R	deleteReservation	Success		
4		R	getReservations	Success		
5		R	getReservation	Success		
6		R	checkForConflictingReservations	Success		
7		R	approveBooking	Success		
8		R	rejectBooking	Success		
9	User & Rights Service	R	checkLogin	Success	Used to ensure that user of tool is authorized	
10		R	checkTestbedRoles	Success	Used during approveBooking or rejectBooking	
11	Master Data Repository	JPA/JDBC	JPQL and/or JPA queries	Success	used to retrieve reservation & resource information for display in calendar view	

Table 7: Test of the Experiment Authoring Tool interfaces

Component: ***Experiment Authoring Tool***		Conducted by: **UoA**		Date: **Feb 2017**		Test Category: **Interface testing**
Preconditions		Users are entered in the RAWFIE Web Portal				
Related Component		**Type**	**Message or API Call**	**Status**	**Remarks/comments**	
1	Launching service	REST	manualStart	Success	Launching tool is correctly informed about the ID of the experiment that will be executed	
2		REST	schedule	Not tested	Schedule launch button not available yet in UI	
3	EDL Compiler & Validator	Other	-	Success	The compiler & validator is correctly adopted when needed	
4	Experiment validation service	Other	HTTP requests	Success	Compilation and validation are smoothly executed in the authoring tool	
5	Master Data Repository	JDBC	JDBC-SQL Queries	Success	Data are correctly retrieved	

Table 8: Test of the Experiment Monitoring Tool interfaces

Component***: Experiment Monitoring Tool***		Conducted by**: Fraunhofer**		Date**: Feb 2017**		Test Category**: Interface testing**
Preconditions		System Monitoring Service collected some data Experiment Status is up-to-date in database				
Related Component		**Type**	**Message or API Call**	**Status**	**Remarks/comments**	
1	Master Data Repository	JDBC	SQL – select experiments of user	Success		
2		JDBC	SQL – select experiment data and status	Success		
3		JDBC	SQL – select UxVs data of experiment	Success		
4	System Monitoring Service	REST	getComponentServiceHealths	Not tested	Not implemented in Experiment Monitoring Tool	
5	Launching Service	REST	cancel	Not tested	Not implemented	

Table 9: Test of the System Monitoring Tool interfaces

Component: System Monitoring Tool		Conducted by: **Fraunhofer**		Date: **Feb 2017**		Test Category: Interface testing
Preconditions		System Monitoring Service collected some data				
Related Component		**Type**	**Message or API Call**	**Status**	**Remarks/comments**	
1	System Monitoring Service	REST	getComponentServiceHealths	Success	Got all health statuses	

Table 10: Test of the Visualisation Tool interfaces

Component***: Visualisation Tool***		Conducted by**: Epsilon**		Date**: Feb 2017**		Test Category**: Interface testing**
Preconditions		• User must be logged in to the portal				
	Related Component	**Type**	**Message or API Call**	**Status**	**Remarks/comments**	
1	Visualisation Engine	Web-socket	startExperiment	Success	Connect to the visualisation engine and retrieve all the information about an experiment and get data for the movement of the UxVs	
2			stopExperiment	Success	Stop the visualisation of an experiment	
3			getExperiments	Success	List all available experiment for the user	
4			getExperimentDetails	Success	Get the details for an experiment that the user wants to visualise	

Table 11: Test of the Data Analysis Tool interfaces

Component: ***Data Analysis Tool***		Conducted by: **HESSO**	Date: **Feb 2017**		Test Category: **Interface testing**
Preconditions		• User must be logged in • Resources must be associated with a user • Resources must be associated with an experiment • Message Bus must be up and schema registry must be accessible • Results database must be accessible • Zeppelin & Spark must be operational			
	Related Component	**Type**	**Message or API Call**	**Status**	**Remarks/comments**
1	Results Database	REST	render()	Success	Graphite is able to be queried via REST and plots results
2	Data Analysis Engine	M-p	buildJob()	Success	Send the Analytics jobs to the Data Analysis Engine through the Kafka message bus
3		REST	Send the SPARK job directly from the Zeppelin UI	Success	Message sent to Spark Directly via REST interface. This is part of Zeppelin by default and works already.

2.5.1.1 Missing components

The following components are not yet implemented and therefore were not tested:

- UxV Navigation Tool

Their development and integration to the web portal is targeted for the next implementation iteration.

2.5.2 Middle tier integration

In the middle-tier integration, activities included testing of interfaces of the following services (with front-end tools, between them and through the message bus):

- EDL Compiler and Validator
- Experiment Validation Service
- User & Rights Service
- Booking Service
- Launching Service
- Experiment Controller
- Data Analysis Engine
- System Monitoring Service
- Testbed Directory Service
- Visualisation Engine

Details on the interface testing activities performed for each component mentioned above are provided in the tables that follow.

Table 12: Test of the EDL Compiler and Validator interfaces

Component***: EDL Compiler and Validator***		Conducted by**: UoA**		Date**: Feb 2017**	Test Category**: Interface testing**
Preconditions		Users are entered in the RAWFIE Web Portal			
Related Component		**Type**	**Message or API Call**	**Status**	**Remarks/comments**
1	Experiment validation service	Other	HTTP requests	Success	Experiments are smoothly validated
2	Master data Repository	JDBC	JDBC-SQL Queries	Success	The get are correctly retrieved

Table 13: Test of the Experiment Validation Service interfaces

Component***: Experiment Validation Service***		Conducted by**: UoA**		Date**: Feb 2017**	Test Category**: interface testing**
Preconditions		Users have entered into the RAWFIE portal.			
	Related Component	**Type**	**Message or API Call**	**Status**	**Remarks/comments**
1	Master data Repository	JDBC	JDBC-SQL Queries	Success	Data are correctly retrieved

Table 14: Test of the User & Rights Service interfaces

Component***: Users & Rights Service***		Conducted by**: Fraunhofer**		Date**: Feb 2017**	Test Category**: Interface testing**
Preconditions					
Related Component		**Type**	**Message or API Call**	**Status**	**Remarks/comments**
1	User & Rights repository	LDAP	bind	Success	User credential validated
2		LDAP	search	Success	Entries (users, groups etc.) listed
3		LDAP	create	Success	Entries (users, groups etc.) added
4		LDAP	modify	Success	Entries (users, groups etc.) edited
5	Master Data Repository	JDBC	SQL select testbed roles	Success	Read roles for testbeds
		JDBC	SQL edit testbed roles	Success	Edit roles for testbeds

Table 15: Test of the Booking Service interfaces

Component***: Booking Service***		Conducted by**: HAI**		Date**: February 2017**	Test Category**: interface testing**
Preconditions		• User must be logged in • UxV resource info must be present in a Master Data Repository • User & Rights Service must be up and running			
	Related Component	**Type**	**Message or API Call**	**Status**	**Remarks/comments**
1	Master Data Repository	JPA/ JDBC	Database call (insert)	Success	
2		JPA/ JDBC	Database call (update)	Success	
3		JPA/ JDBC	Database call (delete)	Success	
4	User & Rights Service	R	checkLogin	Success	Used to ensure that user of service is authorized
5	Aggregate Manager (SFA)	R	Not yet defined	Not tested	Synchronization with Aggregate Manager reservations. Aggregate Manager not yet implemented

Table 16: Test of the Launching service interfaces

Component***: Launching Service***	Conducted by**: HAI**	Date**: Feb 2017**	Test Category**: interface testing**
Preconditions	• User must be logged in • An experiment must be present for a user • Resources must be associated with a user • Resources must be associated with an experiment • Message Bus must be up and configured with appropriate topics (ExperimentStartRequest topic, ExperimentCancelRequest topic) • Experiment Controller must be up and running		

	Related Component	**Type**	**Message or API Call**	**Status**	**Remarks/comments**
1	Experiment Validation Service	R	validateExperiment	Not tested	Experiment Validation Service does not yet exists
2	Experiment Controller	M-p	ExperimentLaunchRequest	Success	Message was sent successfully to Message Bus and consumed by Experiment Controller
3	Resource Controller	M-p	ExperimentCancelRequest	Success	Message was sent successfully to Message Bus
4	Master Data Repository	JPA/ JDBC	Database Interaction	Success	Connection to database succeeded Retrieval/update/insert of information succeeded
5	User & Rights Service	R	checkLogin	Success	Used to ensure that user of service is authorized

Table 17: Test of the Experiment Controller interfaces

Component: Experiment Controller		Conducted by: **CERTH**		Date: **Feb 2017**	Test Category: interface testing
Preconditions		• Message Bus must be up and configured with appropriate topics • Connection to the RAWFIE database is required • The related Resource Controller must be up and running			
	Related Component	**Type**	**Message or API Call**	**Status**	**Remarks/comments**
1	Launching Service	M-c	ExperimentLaunchRequest	Success	Message was successfully consumed by Experiment Controller
3	Master Data Repository	JDBC	Database Interaction	Success	Retrieval of the experiment Script succeeded
4		JDBC	Database Interaction	Success	Retrieval of the resources partitions ids succeeded
5		JDBC	Database Interaction	Success	Retrieval of the testbed coordination system succeeded
6		JDBC	Database Interaction	Success	Insertion/Update inside experimentlog/experiment_execution/ experiment tables succeeded
7	Resource Controller	M-p	ExperimentStartRequest	Success	Message was sent successfully to Message Bus and consumed by Resource Controller
8		M-c	ExperimentStatusMsg	Success	Message was consumed by Experiment Controller
9	Testbed Manager	M-p	ExperimentStartRequest	Success	Message was sent successfully to Message Bus and consumed by Testbed Manager
10	Visualization Engine	M-p	ExperimentStartRequest	Success	Message was sent successfully to Message Bus and consumed by Visualization Engine

Table 18: Test of the Data Analysis Engine interfaces

Component: Data Analysis Engine	Conducted by: **HESSO**	Date: **Feb 2017**	Test Category: Interface testing
Preconditions	• User must be logged in • Resources must be associated with a user • Resources must be associated with an experiment • Message Bus must be up and schema registry must be accessible • Results database must be accessible. • Spark must be operational • Landoop Schema browser must be operational		

	Related Component	**Type**	**Message or API Call**	**Status**	**Remarks/comments**
1	Schema Registry + Schema Browser	REST	/subjects	Success	Successfully iterate over all schemas via the augmented Landoop schema browser. Selection of features can also be done here.
2	Data Analysis Tool	REST	/api/notebook	Success	Data Analysis tool utilizes Zeppelin REST api to POST data
3	Results Database	REST / Sockets	graphite.send()	Success	A simple socket based connection from Spark sends online results to the graphite instance
4	Measurements Database	M-c	hbase.read()	Not Tested	Awaiting hadoop / hbase deployment

Table 19: Test of System Monitoring Service interfaces

Component***: System Monitoring Service***	Conducted by**: Fraunhofer**	Date**: Feb 2017**	Test Category**: Interface testing**
Preconditions			

	Related Component	**Type**	**Message or API Call**	**Status**	**Remarks/comments**
1	Servers (Computer)	O	various	Success	Servers health status collected
2	Testbed Manager	M-c	TestbedHealthStatus	Success	Testbed send their health status to the message bus
3		M-c	UvVHealthStatus	Not tested	Currently not sent to the message bus

Table 20: Test of the Testbed Directory Service interfaces

Component***: Testbed Directory Service***		Conducted by**: IES**		Date**: Feb 2016, April 2017**	Test Category**: interface testing**
Preconditions		Testbeds and Resources tables, as well as all related tables with linked information about testbeds and resources, are present in the Master Data Repository (PostgreSQL DBMS)			
Related Component		**Type**	**Message or API Call**	**Status**	**Remarks/comments**
1	Master Data Repository (PostgreSQL database)	JPA - JDBC Interaction	insertTestbed	Success	Operation performed by a RepositoryHandler class, to support the createTestbed() REST API
2			updateTestbed	Success	Operation performed by a RepositoryHandler class, to support the editTestbed() REST API
3			deleteTestbed	Success	Operation performed by a RepositoryHandler class, to support the deleteTestbed() REST API
4			insertResource	Success	Operation performed by a RepositoryHandler class, to support the createResource() REST API
5			updateResource	Success	Operation performed by a RepositoryHandler class, to support the editResource() REST API
6			deleteResource	Success	Operation performed by a RepositoryHandler class, to support the deleteResource() REST API
7			fetchTestbed	Success	Operation performed by a RepositoryQuery class, to support the searchTestbed() REST API (get details about a specific testbed)
8			fetchTestbeds	Success	Operation performed by a RepositoryQuery class, to support the getTestbeds() REST API (get details about the specified testbeds)
9			fetchResource	Success	Operation performed by a RepositoryQuery class, to support the searchResource() REST API (get details of a specific resource from a specific testbed)
10			fetchResourcesTestbed	Success	Operation performed by a RepositoryQuery class, to support the getResources() REST API (to get details of all resources from a specific testbed)
11			fetchResourcesAvailable	Success	Operation performed by a RepositoryQuery class, to support the getAvailableResources() REST API (get details of all resources which are AVAILABLE for booking tests from a specific testbed)

12			fetchTestbedById	Success	Operation performed by a RepositoryQuery class, to support the testbed search by id
13			fetchTestbedByName	Success	Operation performed by a RepositoryQuery class, to support the testbed search by name
14			fetchTestbedsByUAV	Success	Operation performed by a RepositoryQuery class, to support the testbed search by UAV support
15			fetchTestbedsByUGV	Success	Operation performed by a RepositoryQuery class, to support the testbed search by UGV support
16			fetchTestbedsByUSV	Success	Operation performed by a RepositoryQuery class, to support the testbed search by USV support
17			fetchTestbedsByAUV	Success	Operation performed by a RepositoryQuery class, to support the testbed search by AUV support
18			fetchTestbedsByParameters	Success	Operation performed by a RepositoryQuery class, to support the testbeds search by a combination of search criteria
19			fetchResourceById	Success	Operation performed by a RepositoryQuery class, to support the resource search by id
20			fetchResourceByName	Success	Operation performed by a RepositoryQuery class, to support the resource search by name
21			fetchResourcesByParameters	Success	Operation performed by a RepositoryQuery class, to support the resources search by a combination of search criteria

Table 21: Test of the Visualisation Engine interfaces

Component: ***Visualisation Engine***		Conducted by: **Aberon**		Date: **March 2017**	Test Category: **interface testing**
Preconditions		• User must be logged in to the portal • Measurements and Results repository should be available • Kafka should be available with the necessary topics			
Related Component		**Type**	**Message or API Call**	**Status**	**Remarks/comments**
1	Master Data Repository	JDBC	GetExperimentDetails	Success	Get Experiment Status
2	Resource Controller	M-c	getGoTo	Partial Success	Get the Goto Commands
3		M-c	ExperimentStatusMsg	Not Tested	Not implemented yet
4	Experiment Controller	M-c	ExperimentStartRequest	Success	Get the ExperimentStartRequest from the Message bus
5	UxV Node	M-c	getUxVLocation	Success	Get the location of an UxV
6			getUxVSensorList	Not Tested	Not implemented yet in UxVNode and in VE
7		M-c	getUxVSensorData	Partial Success	Get the sensor data from the UxVs. Not all sensor data is implemented yet.
8		M-c	getUxVStatus	Not tested	
9	Measurement Repository	JDBC	GetFinishedExperimentDetails	Not Tested	Get the experiment details for a finished experiment. Not implemented yet
10		JDBC	GetUxvData	Not Tested	Get the UxV information for a finished experiment. Not implemented yet
11		JDBC	GetSensorData	Not Tested	Get the UxV sensor data for a finished experiment. Not implemented yet

2.5.2.1 Missing components

The following components are not yet implemented and therefore were not tested:

- Accounting Service

Their development and integration is targeted for the next implementation

2.5.3 Testbed & UxV integration

At the testbed level integration, activities included testing of interfaces of the following components (between them and through the message bus with UxVs or middle-tier components):

- The Testbed Manager
- The Monitoring Manager
- The Resource Controller
- UxV node

Details on the interface testing activities performed for each component mentioned above are provided in the tables that follow.

Table 22: Test of the Tesbed Manager interfaces

Component***: Testbed Manager***		Conducted by**: HAI**	Date**: April 2017**		Test Category**: interface testing**
Preconditions		• Confluent platform properly configured, up and running • Related components must be up and running			
Related Component		**Type**	**Message or API Call**	**Status**	**Remarks/comments**
1	System Monitoring Service	M-p	TestbedHealthStatus	Success	System Monitoring properly consumes the message that describes the current health of the machine running the Testbed Manager
2	Resource Controller	M-c	ExperimentStatusMsg	Success	Testbed Manager properly consumes the message that described the status of an experiment from Resource Controller
3		M-p	ExperimentCancelRequest	Not Tested	Cancellation of experiments in emergency cases from Testbed Manager not implemented
4	Experiment Controller	M-c	ExperimentStartRequest	Success	Testbed Manager properly consumes that describes the start of an experiment from Experiment Controller

Table 23: Test of the Monitoring Manager interfaces

Component***: Monitoring Manager***		Conducted by**: HAI**	Date**: April 2017**		Test Category**: interface testing**
Preconditions		• Confluent platform properly configured, up and running • Reliable Internet connection with UxVs			
Related Component		**Type**	**Message or API Call**	**Status**	**Remarks/comments**
1	UxVNode	M-c	FuelUsage	Success	Real data from the devices
2		M-c	CpuUsage	Success	Real data from the devices
3		M-c	StorageUsage	Success	Real data from the devices
4		M-c	Location	Not Tested	Consumption of Location messages is not implemented yet
5		M-c	Attitude	Not Tested	Consumption of Attitude messages is not implemented yet

Table 24: Test of the Resource Controller interfaces

Component: ***Resource Controller***		Conducted by: **CERTH**		Date: **Feb 2017**		Test Category: **interface testing**
Preconditions		• Confluent platform properly configured, up and running • Experiment Controller must be up and running • Related UxV Nodes must be up and running •				
Related Component		**Type**	**Message or API Call**	**Status**	**Remarks/comments**	
1	UxV Node	M-p	WriteHealthStatus	Not tested	Send and receive real-time information to resources	
2		M-p	WriteUxVCommands	Success	Send and receive real-time information to resources	
3		M-p	WriteExperimentStatus	Success	Send real-time kafka messages regarding the status of the experiment	
4		M-c	ReadUxVStatus	Not tested	Resource Controller does not read UxV status yet	
5		M-c	Location	Success	Resource Controller is able to read the actual position of the vehicles	
6	Experiment Controller	M-c	ExperimentStartRequest	Success	Resource Controller successfully receives and parses the experiment to be executed	
7		M-p	ExperimentStatusMsg	Success	Message was sent successfully to Message Bus	
8	Launching Service	M-c	ExperimentCancelRequest	Success	Resource Controller successfully receives and executes cancel requests	
9	Testbed Manager	M-c	ExperimentCancelRequest	Not tested	Functionality not implemented yet	
		M-p	ExperimentStatusMsg	Not tested	Functionality not implemented yet	

Table 25: Test of the UxV Node interfaces

Component: ***UxV Node***		Conducted by: **Robotnik, MST**		Date: **Feb 2017**		Test Category: **interface testing**

Preconditions		• A server running the Confluent platform
		• UxV manufacturer's (e.g. Robotnik) specific preconditions:
		  • The necessary topics should be already registered
		  • A server running the Confluent platform should be available with the necessary topics
		  • Input from the resource controller
		  • Reliable Internet connection				

	Related Component	**Type**	**Message or API Call**	**Status**	**Remarks/comments**	
1	Resource Controller	M-c	Goto	Success	GPS coordinates accuracy and threshold for next waypoint needs to be configured	
2			KeepStation	Success	Tested with success by MST; Ground vehicles are accepting this command as no waypoint commanded	
3			Abort	Success	Tested with success	
4			Location	Success	Without GPS specifying an origin of coordinates is needed. For indoor scenarios Cartesian coordinates are given with standard goto message	
5	Visualization Tool	M-p	Location message	Success	Visualization indoors is now using specific images created with mapping tools normally using 2D scans	

6	Visualization Engine	M-p	Location message	Success	Get the location of an UxV
7		M-p	SensorReadingScalar	Partial Success	Get the sensor data from the UxVs. Not all sensor data is implemented yet.
8		M-p	UxVStatus	Not tested	
9	Data Analytics	M-p	SensorReadingScalar	Success	Tested Salinity, Conductivity, and SoundSpeed with water vehicles. Temperature measurements from both water and ground vehicles
10			Current	Partial Success	Tested with success by MST
11			Voltage	Partial Success	Tested with success by MST
12			StorageUsage	Partial Success	Tested with success by MST
13			FuelUsage	Partial Success	Tested with success by MST
14			CpuUsage	Partial Success	Tested with success by MST
15			SensorInfo	Partial Success	Tested with success by MST
16	Monitoring manager	M-p	FuelUsage	Success	Real data from the devices
17			CpuUsage	Success	Real data from the devices
18			StorageUsage	Success	Real data from the devices
19	Schema Registry	M	CachedSchemaRegistryClient	Success	Get the schema registry

2.5.3.1 Missing components

The following components are under implementation and therefore were not yet fully tested from an integration point of view:

- Network controller
- Aggregate Manager (SFA)
- UxV Proximity Component

Their integration are planned for the next implementation round.

2.6 Verification scenarios results

2.6.1 Frontend Tier

The verification of the Front-end tier mainly consists testing the Web Portal GUI elements.

2.6.1.1 Web Portal

Table 26: Verification test of the Web Portal - Login/ Logout

Test ID**: WP01**	Conducted by**: Fraunhofer**	Date**: April 2017**	Test Category**: Verification Tests (front end tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Web Portal - Login/ Logout***		
Preconditions	• User entered in the User & Rights repository		
Related Requirements	PT-WEB-P-001, PT-WEB-P-002		
Tools Used	• Browser		

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

Table 27: Verification test of the Web Portal – Language selection

Test ID**: WP02**	Conducted by**: Fraunhofer**	Date**: April 2017**	Test Category**: Verification Tests (front end tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Web Portal – Language selection***		
Preconditions	• Translation available		
Related Requirements	PT-WEB-P-001		
Tools Used	• Browser		

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	user opens RAWFIE any web page	web page with language selection displayed,	Success	
2	user changes the language	web page displayed in the selected language	Partial success	Language is changed, but only a few text are translated (missing translations)

Table 28: Verification test of the Web Portal – User management

Test ID**: WP03**	Conducted by**:**	Date**:**	Test Category**: Verification Tests (front end tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Web Portal – User management***		
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	Not tested	
2	Browser 1: Navigate to registration requests page	No registration request displayed	Not tested	
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.	Not tested	
4	Browser 2:Try to login with the submitted login credentials	Login failed. Display message that user is looked	Not tested	
5	Browser 1: Reload registration requests page	The new registration request is show	Not tested	
6	Browser 1: Accept the new user	The new user is now unlooked	Not tested	
7	Browser 2: Try to login with the submitted login credentials	Login successful.	Not tested	
8	Browser 1: Navigate to the user list and delete the new user	User deleted	Not tested	
9	Browser 2: Logout and try to login with the submitted login credentials	Login failed. Show invalid credentials messages	Not tested	

2.6.1.2 Wiki Tool

Table 29: Verification test of the Wiki Tool – Component Help

Test ID**: WT01**	Conducted by**: Fraunhofer**	Date**: April 2017**	Test Category**: Verification Tests (front end tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Wiki Tool – Component help***		
Preconditions	• Help pages added to the Wiki		
Related Requirements	PT-WEB-P-003		
Tools Used			

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

Table 30: Verification test of the Wiki Tool – Editing

Test ID**: WT02**	Conducted by**: Fraunhofer**	Date**: April 2017**	Test Category**: Verification Tests (front end tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Wiki Tool – Editing***		
Preconditions	• User for Wiki management defined		
Related Requirements	PT-WEB-P-003		
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	Success	
2	Try to edit the page	Editing not possible due to missing rights	Success	
3	Login as administrator and assign the Wiki manager right to the user	The user has now the Wiki manager right	Not tested	User right directly changed in the User & Rights repository
4	Login as the first user and open a page in the Wiki Tool	Wiki page displayed	Success	
5	Try to edit the page	Editing allowed as changes are save	Success	

2.6.1.3 Resource Explorer Tool

Table 31: Verification test of the Browse testbeds and UxVs and start booking

Test ID**: RET01**	Conducted by**: Fraunhofer**	Date**: April 2017**	Test Category**: Verification Tests (front end tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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	• Browser		

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	Success	
2	user selects a testbed	Resource Explorer Tool displays all testbed details and a list of available UxVs	Success	
3	user selects a UxV	Resource Explorer Tool displays all UxVs details	Success	
4	user starts booking	Booking Tool opened with the selected resources	Not tested	Not implemented

2.6.1.4 Booking Tool

Table 32: Verification test of the Booking Tool Calendar View and its display options

Test ID: **BT01**	Conducted by: **HAI**	Date: **February 2017**	Test Category: **Verification Tests (web tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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-S-008		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Click of Bookings menu item	Navigation to Booking Tool (Calendar View)	Success	
		Calendar view displays by default the present week with all defined bookings	Success	
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	Success	
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	Success	
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	Success	

Table 33: Verification test of the Booking Tool Calendar View Interactions

Test ID**: BT02**	Conducted by**: HAI**	Date**: February 2017**	Test Category**: Verification Tests (web tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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-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	Success	
		If click occurs on a future timeslot the "Create Reservation" window opens	Success	
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	Success	
	(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	Success	
3	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)	Success	

Table 34: Verification test of the Booking Tool Create Reservation

Test ID: **BT03**	Conducted by: **HAI**	Date: **February 2017**	Test Category: **Verification Tests (web tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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-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	Success	
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 reservation then the new reservation is created and displayed in the Calendar View. Reservation is put in PENDING state	Success	
		If common resources exist with the overlapping reservation then the new reservation is not created and a warning message is displayed	Partial Success	Result may depend on status of pre-existing reservation

Table 35: Verification test of the Booking Tool Edit Reservation Actions

Test ID: **BT04**	Conducted by: **HAI**	Date: **February 2017**	Test Category: **Verification Tests (web tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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-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	Success	
	status=OK user= owner of reservation role= EXPERIMENTER	Actions available: OK, CANCEL DELETE	Success	
	status=REJECTED user= owner of reservation role= EXPERIMENTER	Actions available: OK, CANCEL DELETE	Success	
	status=PENDING user= owner of reservation role= TESTBED_OP	Actions available: OK, CANCEL, DELETE, APPROVE, REJECT	Success	
	status=PENDING user= not owner of reservation role= TESTBED_OP	Actions available: CANCEL, APPROVE, REJECT	Success	
	status=OK user= owner of reservation role= TESTBED_OP	Actions available: CANCEL, DELETE, REJECT	Success	
	status=OK user= not owner of reservation role= TESTBED_OP	Actions available: CANCEL, REJECT	Success	
	status=REJECTED user= owner of reservation role= TESTBED_OP	Actions available: CANCEL, DELETE, APPROVE	Success	
	status= REJECTED user= not owner of reservation role= TESTBED_OP	Actions available: CANCEL, APPROVE	Success	
	user= not owner of reservation	No actions available	Success	

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	Success	
		If the changes do introduce conflicts in both timeslots and selected resources then a warning message appears and no further action is performed	Success	
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	Success	
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	Success	
5	User with TESTBED_OP role presses REJECT button	reservation status becomes REJECTED and color changes appropriately in the Calendar view	Success	

2.6.1.5 Experiment Authoring Tool

Table 36: Verification test of the in-Textual Editor Experiments definition

Test ID**: EAT01**	Conducted by**: UoA**	Date**: April 2017**	Test Category**: Verification Tests (front end tier – middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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			

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

Table 37: Verification test of the Textual Editor Experiments Update

Test ID**: EAT02**	Conducted by**: UoA**	Date**: April 2017**	Test Category**: Verification Tests (front end tier – middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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			

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

Table 38: Verification test of the in-Visual Editor Experiments Define

Test ID**: EAT03**	Conducted by**: UoA**	Date**: April 2017**	Test Category**: Verification Tests (front end tier – middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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	•		

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

Table 39: Verification test of the in-Visual Editor Experiments Update

Test ID**: EAT04**	Conducted by**: UoA**	Date**: April 2017**	Test Category**: Verification Tests (front end tier – middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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	•		

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

Table 40: Verification test of the Editor switching

Test ID: **EAT05**	Conducted by: **UoA**	Date: **April 2017**	Test Category: **Verification Tests (front end tier – middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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	•		

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

Table 41: Verification test of the experiment Launchings

Test ID**: EAT05**	Conducted by**: UoA**	Date**: April 2017**	Test Category**: Verification Tests (front end tier – middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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	•		

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

2.6.1.6 *Experiment Monitoring Tool*

Table 42: Verification test of the Visualisation of experiment status

Test ID: **EMT01**	Conducted by: **Fraunhofer**	Date: **April 2017**	Test Category: **Verification Tests (front end tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Visualisation of experiment status***		
Preconditions	• connection to the Launching Service ok • knowledge about the experiments state needed on user side (to check results)		
Related Requirements	PT-EXM-T-001,		
Tools Used	• Browser		

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	Success	
2	user selects a experiment	Experiment Monitoring Tool displays all experiment details (date / timespan; related testbed; list of used UxVs; execution state ; link to the used EDL)	Partial Success	Link to EDL not implemented

2.6.1.7 *System Monitoring Tool*

Table 43: Verification test of the Visualisation of system and UxV health status

Test ID: **SMT01**	Conducted by: **Fraunhofer**	Date: **April 2017**	Test Category: **Verification Tests (front end tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Visualisation of system and UxV health status***		
Preconditions	• connection to the System Monitoring Service (may not be necessary if System Monitoring Service collects all necessary data anyway) • administrative knowledge about the system state needed on user side (to check results)		
Related Requirements	PT-SYM-T-001		
Tools Used	• Browser		

Step	Action	Expected Result	Status	Remarks
1	user opens System Monitoring Tool in the Web Portal	the System Monitoring Tool displays views with status of, middleware components, testbeds components, UxVs components	Partial success	Servers and Testbeds displayed. UxVs did not send status information (to be implemented)

2.6.1.8 *UxV Navigation Tool*

Table 44: Verification test of the UxV navigation tool access and produced instructions validation

Test ID**: UxVNT01**	Conducted by**: CERTH**	Date**: April 2017**	Test Category**: Verification Tests (front end tier – middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Validate Experiments***		
Preconditions	• Requires Web Portal to be functioning and accessible		
Related Requirements	PT-EXV-S-001, PT-EXV-S-002, PT-EXV-S-003		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Access the UxV Navigation Tool through the portal	Ability to navigate the swarm	Not tested	Not implemented
2	Validate the produced instructions Validate the schema of the JSON output file Validate the data format of the JSON output file Validate the size of the JSON output file	All validation successful. The output data should be accessible and compatible with the required format	Not tested	Not implemented

2.6.1.9 *Visualisation Tool (Aberon)*

Table 45: Verification test of the User request handling

Test ID**: VIS01**	Conducted by**: Aberon**	Date**: April 2017**	Test Category**: Verification Tests (front end)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***User request handling***		
Preconditions	• Requires visualization tool to be functioning & accessible. • Requires visualization engine to be functioning & accessible.		
Related Requirements	PT-VIS-T-001, PT-VIS-T-007		
Tools Used	•		

Step	Action	Expected Result	Status	Remarks
1	User sends a predefined websocket request via the visualization tool	The visualization tool forwards it to the visualization engine	Success	
2	Handle the response from the visualization engine	The response is visualized on the user screen	Success	

Table 46: Verification test of the Geospatial data handling

Test ID: **VIS02**	Conducted by: **Aberon**	Date: **April 2017**	Test Category: **Verification Tests (front end)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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-T-001, PT-VIS-T-002, 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	Acquire predefined geospatial data (WMS, WFS) via the message bus	Data is properly received in the correct format at the VE	Success	
2	Modify the data to be suited for the VT and send it via websocket to VT	VT renders the data and plots it on the screen	Success	

Table 47: Verification test of the Geospatial data modification

Test ID: **VIS03**	Conducted by: **Aberon**	Date: **April 2017**	Test Category: **Verification Tests (front end)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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-T-001, PT-VIS-T-002, 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	Acquire predefined geospatial data (WMS, WFS) via the message bus	Data is properly received in the correct format at the VE	Success	
2	Add a layer of information data and send it to the VT	VT plots the data and the layer properly	Success	

Table 48: Verification test of the Experiment Controller communication

Test ID**: VIS04**	Conducted by**: Aberon**	Date**: April 2017**	Test Category**: Verification Tests (front end)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Experiment Controller communication***		
Preconditions	• Requires experiment controller to be functioning & accessible. • Requires visualization engine to be functioning & accessible.		
Related Requirements	PT-VIS-T-001		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Receive a message that the experiment has started from the Experiment Controller	The visualization tool starts the experiment	Partial success	Tested with previous component, with experiment controller not yet
2	Receive a message that the experiment has stopped from the Experiment Controller	The VT stops the experiment	Partial success	Tested with previous component, with experiment controller not yet

Table 49: Verification test of the Visualization Tool Interaction

Test ID**: VIS05**	Conducted by**: Aberon**	Date**: April 2017**	Test Category**: Verification Tests (front end)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Visualization Tool Interaction***		
Preconditions	• Requires visualization tool to be functioning & accessible. • Requires visualization engine to be functioning & accessible.		
Related Requirements	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/Disable different features of the visualization tool (e.g. show/hide speed web widget)	The user sees the updated plot (show/hide speed web widget)	Partial success	

Table 50: Verification test of the Camera interaction

Test ID: **VIS06**	Conducted by:	Date: **April 2017**	Test Category: **Verification Tests (front end)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Camera interaction***		
Preconditions	• Requires visualization tool to be functioning & accessible. • Requires visualization engine to be functioning & accessible. • Requires Experiment controller to be functioning & accessible.		
Related Requirements	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	Retrieve with the visualization engine quasi real time data from one UxV, processes it and send it to the visualization tool	The VT plots the data properly	Success	
2	Change the camera view for the scenario	Data camera is adjusted	Not tested	Not implemented

2.6.1.10 Data Analysis Tool

Table 51: Verification test of the provision of an interface to the Analysis Engine by the Analysis Tool

Test ID: **PT-DAA-T-001**	Conducted by:	Date: **April 2017**	Test Category: **Verification Tests (front end tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Analysis Tool will provide an interface to the Analysis Engine (DAE)***		
Preconditions	• Working message bus • Working schema registry • Working Data Analysis Tool		
Related Requirements	PT-DAA-T-002, PT-DAA-T-001, PT-DAA-T-004, PT-DAA-T-005		
Tools Used			

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	User logs in to the web portal	Login successful	Success	Note: there is still some integration work required to do SSO from the DAT
2	DAT queries available schemas from Schema Registry	All schemas are returned successfully	Success	Works natively due to the utilization of Landoop registry query
3	DAT allows user to select the data they want to work with as well as the machine learning algorithm and hyper-parameters	Job is sent via REST to the DAE	Success	The DAT provides 1) interface for user to enter their own code (Zeppelin) and 2) an interface to select the schema (i.e. landoop with rawfie adaptor).

Table 52: Verification test of the ability of the Analysis Tool to query available data schemas

Test ID: **PT-DAA-T-002**	Conducted by:	Date: **April 2017**	Test Category: **Verification Tests (front end tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Analysis Tool will be able to query available data schemas***		
Preconditions	• Working message bus • Working schema registry • Working Data Analysis Tool		
Related Requirements	PT-DAA-T-003		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	User logs in to the web portal	Login successful	Success	Note: there is still some integration work required to do SSO from the DAT
2	DAT queries available schemas from Schema Registry	All schemas are returned successfully	Success	Works natively as it is supported by base Landoop fork.

Table 53: Verification test of the ability of the Analysis Tool to read results from the results database

Test ID: **PT-DAA-T-003**	Conducted by:	Date: **April 2017**	Test Category: **Verification Tests (front end tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Analysis Tool will be able to read results from the results database***		
Preconditions	• Working message bus • Working schema registry • Working Data Analysis Tool • Working results database [graphite]		
Related Requirements	PT-DAA-T-001, PT-DAA-T-005		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	User logs in to the web portal	Login successful	Success	Note: there is still some integration work required to do SSO from the DAT
2	User builds job	Job successfully built (or error) and sent to DAE	Success	Zeppelin will alert user on a failure. Furthermore, status can be observed on Spark page.
3	Results are shown in Results DB or error is shown in Zeppelin	Job results are shown as they are processed via graphite UI / shown in Zeppelin interpreter	Success	

2.6.2 Middle Tier (Services and Communication components)

2.6.2.1 Testbed Directory Service (IES)

Table 54: Verification test of the resources information retrieval and resources search

Test ID**: TD01**	Conducted by**: IES**	Date**: April 2017**	Test Category**: Verification Tests (Middle Tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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 the criteria for selecting specific resources		
Related Requirements	PT-DIR-S-003, PT-DIR-S-004, PT-DIR-S-006		
Tools Used	SOAP UI or Web Browser		

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 resource in case the *getAllResources()* interface is used	Success	
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, 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 resources matching the search criteria	Success	
2.b	The ***/request/resource/name/{name}*** REST interface is called, specifying the name of a specific resource			
3.b	The ***/request/resources?param1=value1&param2=value2&param3=value3&param4=value4*** 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 4 possible search parameters: • *resource_status* • *resource_status_message* • *resource_type* • *health*			

Table 55: Verification tests for adding or removing a testbed facility

Test ID**: TD02**	Conducted by**: IES**	Date**: February 2016**	Test Category**: Verification Tests (Middle Tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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**)	Success	
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)	Success	
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		Success	

Table 56: Verification test of the registration or removal of a new UxV node into a testbed facility

Test ID**: TD03**	Conducted by**: IES**	Date**: February 2016**	Test Category**: Verification Tests (Middle Tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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 he is looking for, or the list criteria for selecting specific testbeds		
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**)	Success	
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**)	Success	
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		Success	

Table 57: Verification test of the testbeds information retrieval and testbeds search

Test ID: **TD04**	Conducted by: **IES**	Date: **April 2017**	Test Category: **Verification Tests (Middle Tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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 it can just provide one or a set of 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	Success	
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	Success	
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.	Success	
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*	The Testbed Directory Service gives back a JSON response message, containing details about the available testbeds conforming to the search criteria	Success	
4.c	The ***/request/testbed/uav*** REST interface is called, looking for all testbeds supporting UAV resources		Success	
5.c	The ***/request/testbed/ugv*** REST interface is called, looking for all testbeds supporting UGV resources		Success	
6.c	The ***/request/testbed/usv*** REST interface is called, looking for all testbeds supporting USV resources		Success	
7.c	The ***/request/testbed/auv*** REST interface is called, looking for all testbeds supporting AUV resources		Success	

2.6.2.2 EDL Compiler and Validator

Table 58: Verification test of the Experiments compilation

Test ID**: ECV01**	Conducted by**: UoA**	Date**: April 2017**	Test Category**: Verification Tests (front end tier – middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Compile Experiments***		
Preconditions	• User entered in the RAWFIE Portal		
Related Requirements	PT-CPV-001, PT-CPV-002, PT-CPV-003, PT-CPV-004, PT-EXV-S-001, PT-EXV-S-002, PT-EXV-S-003		
Tools Used			

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Access to the authoring tool through the RAWFIE Web Portal	Redirection to the editors interface	Success	
2	Write a simple experiment	The experiment workflow is presented in the available editors	Success	
3	Compile the experiment	The necessary files required by the remaining RAWFIE components are produced	Success	

Table 59: Verification test of the Experiments validation

Test ID: **ECV02**	Conducted by: **UoA**	Date: **April 2017**	Test Category: **Verification Tests (front end tier – middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Validate Experiments***		
Preconditions	• User entered in the RAWFIE Portal		
Related Requirements	PT-CPV-001, PT-CPV-002, PT-CPV-003, PT-CPV-004, PT-EXV-S-001, PT-EXV-S-002, PT-EXV-S-003		
Tools Used	•		

Step	Action	Expected Result	Status	Remarks
1	Access to the authoring tool through the RAWFIE Web Portal	Redirection to the editors interface	Success	
2	Write a simple experiment	The experiment workflow is presented in the available editors	Success	
3	Validate the experiment	Validation is performed and error / warning messages are presented in the editors	Success	

2.6.2.3 Users & Rights Service

Table 60: Verification test of the Users & Rights Service login checking

Test ID: **URS01**	Conducted by: **Fraunhofer**	Date: **April 2017**	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Login checking***		
Preconditions	• Valid user name and password known		
Related Requirements	• PT-USR-S-001		
Tools Used	• SOAPUI REST client		

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

Table 61: Verification test of the Users & Rights Service roles/rights checking

Test ID: **URS02**	Conducted by: **Fraunhofer**	Date: **April 2017**	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Roles/rights checking***		
Preconditions	• User with the tested roles is available		
Related Requirements	PT-USR-S-002		
Tools Used	• SOAPUI REST client		

Step	Action	Expected Result	Status	Remarks
1	Role request with not available roles for a user is sent to the Users & Rights Service	Users & Rights Service returns failure	Success	
2	Role request with available roles for a user is sent to the Users & Rights Service	Users & Rights Service returns OK	Success	

Table 62: Verification test of the user rights checks

Test ID: **URS03**	Conducted by: **Fraunhofer**	Date: **April 2017**	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Check user rights***		
Preconditions	• Valid user rights known		
Related Requirements	PT-USR-S-001, PT-USR-S-002, PT-USR-S-003		
Tools Used	• SOAPUI REST client		

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

2.6.2.4 Booking Service

Table 63: Verification test of Booking Service add reservation functionality

Test ID: **BS01**	Conducted by: **HAI**	Date: **February 2017**	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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 (experiment level booking) PT-BOO-S-002 PT-BOO-S-004 PT-BOO-S-005 PT-BOO-S-007 PT-BOO-S-011		
Tools Used	Maven, Java test client, HttpRequestor Firefox plugin Booking Tool UI		

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	Success	
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)	Success	
		If email sending is enabled then email is send to both the creator and the testbed operator of the reserved resources	Success	
		The returned response contains the newly created reservationId and the reservation status	Success	
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	Success	

Table 64: Verification test of Booking Service edit reservation functionality

Test ID: **BS02**	Conducted by: **HAI**	Date: **February 2017**	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Booking Service edit 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		
Tools Used	Booking Tool UI		

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	Success	
		If existing reservation status!= PENDING then no update should be possible and a proper failure message is returned	Success	
		If time related changes refer to an interval in the past then a proper failure message is returned	Success	
	(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	Success	
	(If status= PENDING & user credential match)	If no resources conflicts are detected the changes are accepted and the corresponding DB tables updated	Success	
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	Success	Success of reservation edit depends on whether overlaps introduce conflicts according to the steps described in step 1

Table 65: Verification test of Booking Service approve reservation functionality

Test ID: **BS03**	Conducted by: **HAI**	Date: **February 2017**	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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-011 PT-NF-002		
Tools Used	Maven, Java test client, HttpRequestor Firefox plugin Booking Tool UI		

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	Success	
		If provided credentials do not refer to an authorized platform user with role=TESTBED_OP then a proper failure message is returned	Success	
		If reservationId refers to a reservation with status !=PENDING then a proper failure message is returned	Success	
		If reservationId refers to a past reservation then then a proper failure message is returned	Success	
		If conflicts are detected with any other APPROVED reservation then then a proper failure message is returned	Success	
2	(If status= PENDING & caller=TESTBED_OP & no conflicts detected	Status change is accepted and corresponding DB tables updated	Success	
		An email is send to the owner of the reservation	Success	
		A ReservationStatusMsg is send to Message bus	Success	

Table 66: Verification test of Booking Service reject reservation functionality

Test ID: **BS04**	Conducted by: **HAI**		Date: **February 2017**	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3			
Software Configuration	See section 2.3.3			
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-011 PT-NF-002		
Tools Used		Maven, Java test client, HttpRequestor Firefox plugin Booking Tool UI		

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	Success	
			If provided credentials do not refer to an authorized platform user with role=TESTBED_OP then a proper failure message is returned	Success	
			If reservationId refers to a reservation with status !=PENDING or APPROVED then a proper failure message is returned	Success	
			If reservationId refers to a past reservation then then a proper failure message is returned	Success	
2	(If status= PENDING & caller=TESTBED_OP		Status change is accepted and corresponding DB tables updated	Success	
			An email is send to the owner of the reservation	Success	
			A ReservationStatusMsg is send to Message bus	Success	

Table 67: Verification test of Booking Service delete reservation functionality

Test ID**: BS05**	Conducted by**: HAI**	Date**: February 2017**	Test Category**: Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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	Maven, Java test client, HttpRequestor Firefox plugin Booking Tool UI		

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	Success	
		If reservationId refers to a past reservation then a proper failure message is returned	Success	
		If reservationId refers to a reservation with resources involved in a currently running experiment a proper failure message is returned	Success	
		If none of the above then status change to CANCELLED	Success	

Table 68: Verification test of Booking Service retrieve reservation(s) functionality

Test ID: **BS06**	Conducted by: **HAI**	Date: **February 2017**	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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	HttpRequestor Firefox plugin		

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	Success	
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	Success	

Table 69: Verification test of Booking Service check for conflicts functionality

Test ID: **BS07**	Conducted by: **HAI**	Date: **February 2017**	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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-008		
Tools Used	HttpRequestor Firefox plugin		

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	Success	
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	Success	

Table 70: Verification test of Booking Service simultaneous reservations support

Test ID**: BS08**	Conducted by**: HAI**	Date**: February 2017**	Test Category**: Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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-009		
Tools Used	soapUI		

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	Success	

2.6.2.5 Launching Service

Table 71: Verification test of the Launching Service manual start (short term launching)

Test ID**: LS01**	Conducted by**: HAI**	Date**: March 2017**	Test Category**: Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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	Experiment Authoring Tool UI Maven, Java test client, HttpRequestor Firefox plugin		

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

Table 72: Verification test of the Launching Service schedule (long term launching)

Test ID: **LS02**	Conducted by: **HAI**	Date: **March 2017**	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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-012 PT-LAU-S-013 (by design) PT-BOO-S-011		
Tools Used	Maven, Java test client, HttpRequestor Firefox plugin		

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

		LaunchingServiceActionResp json message is returned containing the generated executionId and the status of the experiment	Success	
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	Success	
		Launching service generates an ExperimentStartRequest to the Message Bus (targeting the Experiment Controller).	Success	
		Scheduled job (for the executionId) is removed from scheduler	Success	

Table 73: Verification test of the Launching Service cancellation request

Test ID: **LS03**	Conducted by: **HAI**	Date: **March 2017**	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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	Maven, Java test client, HttpRequestor Firefox plugin		

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	Success	
		If supplied user credentials do not match an authorized user then a proper failure message is returned	Success	
		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)	Success	
2	(case executionId exists)	If the experiment is already running (status= ONGOING) then cancellation is not possible and a proper failure message is returned	Success	
		If no schedule job is found in Launching scheduler then a proper failure message is returned	Success	
3	(executionId exists and the execution is still in the scheduler)	Job is removed from the scheduler	Success	
		Master DB tables are properly updated (tables Experiment_Execution, Reservation_item). The status of the experiment changes to CANCELLED	Success	
		LaunchingServiceActionResp json message is returned containing with the executionId, status= CANCELLED and empty message field	Success	
		The user (owner) of the experiment and the testbed operator are informed by an appropriate notification (email)	Failure	Functionality not implemented

Table 74: Verification test of Launching Service simultaneous launching capability

Test ID: **LS04**	Conducted by: **HAI**	Date: **March 2017**	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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		
Tools Used	soapUI		

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	Success	

2.6.2.6 *Visualisation Engine*

Table 75: Visualisation engine user request handling

Test ID: **VE01**	Conducted by: **Aberon**	Date: **April 2017**	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***User request handling***		
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	Success	
2	Visualization engine sends through websocket the response	Visualization tool receives response	Success	

Table 76: Visualization engine geospatial data modification

Test ID: **VE02**	Conducted by: **Aberon**	Date: **April 2017**	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Geospatial data modification t***		
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 the message bus	The visualization engine handles the request	Success	
2	Visualization engine update data in database	Data is properly stored in the database for future retrieval	Success	

Table 77: Visualization engine camera interaction

Test ID: **VE03**	Conducted by: **Aberon**	Date: **April 2017**	Test Category: **Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Camera interaction***		
Preconditions	• Requires visualization tool and visualization engine to function and be accessible and the UxV to send video data		
Related Requirements	VIS01		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Visualization engine receive request from visualization tool to start the camera stream	Visualization engine forward this request to the UxV	Not tested	Not implemented yet

2.6.2.7 Data Analysis Engine

Table 78: Verification test of the ability of the Analysis Engine to query message bus streams & schemas from the schema registry

Test ID**: PT-DAA-E-001**	Conducted by**: HESSO**	Date**: April 2017**	Test Category**: Verification Tests (front end tier)**
Hardware Configuration	Hardware will be provisioned via Ansible and will conform to the Spark Requirements.		
Software Configuration	• Spark 2.0 • Graphite 0.10.0 • Confluent 3.0 • Zeppelin 0.8		
Test Name:	***Analysis Engine will be able to query message bus streams & schemas from the schema registry***		
Preconditions	• Working message bus • Working schema registry • Working Data Analysis Tool		
Related Requirements	PT-DAA-S -001, PT-DAA-S -002		
Tools Used	Spark, Landoop Schema Registry w/ RAWFIE Adaptor , Graphite, Grafana		

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	User deploys job via CLI or via web portal in ipython-style notebooks	1) DAE checks if job is a pre-existing jar, else compiles a new one 2) Zeppelin packages notebook	Success	
2	DAE verifies schema from registry and starts a spark job that acquires data from the message bus	The job is successfully build and uploaded to the job server	Success	

Table 79: Verification test of the ability of the Analysis Engine to receive messages from the Analysis Tool

Test ID**: PT-DAA-E-002**	Conducted by**: HESSO**	Date**:**	Test Category**: Verification Tests (front end tier)**
Hardware Configuration	• Hardware will be provisioned via Ansible and will conform to the Spark Requirements.		
Software Configuration	• Spark 2.0, Graphite 0.10.0, Confluent 3.0, Zeppelin 0.8		
Test Name:	***Analysis Engine will be able to receive messages from the Analysis Tool***		
Preconditions	• Working message bus • Working schema registry • Working Data Analysis Tool		
Related Requirements	PT-DAE-001 (PT-DIR-S-001)		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	User builds a job on the Data Analysis Tool	Job is successfully checked for errors	Success	
2	Data Analysis Engine receives job via REST and builds a job	The job is successfully compiled (or an error returned)	Success	Provided natively by Zeppelin
3	Data Analysis Engine builds job and sends data to Spark	The job is converted to a JAR if using the CLI; otherwise the job is packaged by Zeppelin	Success	

Table 80: Verification test of the ability of the Analysis Engine to write data to the results database

Test ID**: PT-DAA-E-003**	Conducted by**: HESSO**	Date**:**	Test Category**: Verification Tests (front end tier)**
Hardware Configuration	• Hardware will be provisioned via Ansible and will conform to the Spark Requirements.		
Software Configuration	• Spark 2.0 • Graphite 0.10.0 • Confluent 3.0 • Zeppelin 0.8		
Test Name:	***Analysis Engine will be able to write data to the results database***		
Preconditions	• Working message bus • Working schema registry • Working Data Analysis Engine • Working Graphite Instance		
Related Requirements	PT-DIR-S-002		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	User builds a job and the jar is uploaded to the spark / user writes custom code in Zeppelin	Job is uploaded successfully and the job is registered in spark	Success	
2	Spark Engine sends results to the Graphite instance as it processes the data	Graphite displays a runtime stream of processed data	Success	

2.6.2.8 System Monitoring Service

Table 81: Verification test of the System Monitoring

Test ID**: SYMS01**	Conducted by**: Fraunhofer**	Date**: April 2017**	Test Category**: Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***System Monitoring***		
Preconditions	•		
Related Requirements	PT-SYM-S-001, PT-SYM-S-002		
Tools Used	Browser		

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	Success	
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	Success	
3	System Monitory Tool request status information	Service collects the information and returns it	Success	

Table 82: Verification test of the System Monitoring Problem Notifications

Test ID**: SYMS02**	Conducted by**: Fraunhofer**	Date**: April 2017**	Test Category**: Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***System Monitoring Problem Notifications***		
Preconditions	• Notification receivers are configured • Status information is collected		
Related Requirements	PT-SYM-S-003		
Tools Used	SSH client, Browser Email client		

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Problem occurred (server down etc.)	Services send email notifications of the configured receivers.	Failure	Emails currently not configured
2	System Monitory Tool request status information	Problems are visualized in the System Monitory Tool	Success	

2.6.2.9 Accounting Service

Table 83: Verification test of the Accounting data collection

Test ID**: ACCS01**	Conducted by**:** **Fraunhofer**	Date**: April 2017**	Test Category**:** **Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
Test Name:	***Accounting data collection***		
Preconditions	• Accounting data is empty for the used user		
Related Requirements	PT-ACC-S-002, PT-ACC-S-003		
Tools Used	Browser Email client		

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Experiment is completed. Notifications sent on the message bus.	Accounting received the event and computes the charge for the experiments	Not tested	Not implemented
2	Billing period ends	Bill is sent to the user	Not tested	Not implemented

2.6.2.10 Experiment Controller

Table 84: Verification test of Experiment Controller workflow

Test ID**: EC01**	Conducted by**: CERTH**	Date**: April 2017**	Test Category**: Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	Java 8		
Test Name:	***Execute experiment workflow***		
Preconditions	• Requires web portal to be functioning and accessible. • The experimenter has already created the script for the experiment of interest • The chosen resource must be completely available and ready to use • Requires RAWFIE DB to be properly defined and continuously accessible		
Related Requirements	PT-EXP-C-001, PT-EXP-C-002, PT-EXP-C-003, PT-EXP-C-004, PT-EXP-C-005, PT-EXP-C-006, PT-EXP-C-007		
Tools Used			

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	The experimenter forwards the script to the Experiment Controller in order to start or barely execute the next action of the resource mission	Successful forwarding and start of execution	Success	
2	The instructions are forwarded to the corresponding testbed facility	Testbed facility received the instructions correctly	Success	
3	The resource receives the new set of instructions as generated from the script for overriding the experiment workflow	The resource overrides its current experiment according to the new instructions	Success	
4	The Experiment Controller supports the execution of experiments that involve multiple testbeds	Simultaneous visualization of different experiment on different locations	Not Tested	
5	Update the status of a running experiment inside the database	The status update can be utilized by any RAWFIE component	Success	

2.6.3 Testbed Tier (Testbeds and Resources control components)

2.6.3.1 Monitoring Manager

Table 85: Verification test of Monitoring Activity

Test ID**: MM01**	Conducted by**: HAI**	Date**: April 2017**	Test Category**: Verification Tests (middle tier)**
Hardware Configuration	Testbed site PC		
Software Configuration	• Java 8 • Confluent Platform 2.01		
Test Name:	***Check Monitoring Activity***		
Preconditions	• Requires the resource controller to be accessible. • Requires the network controller to be accessible. • Requires the data tier to be accessible.		
Related Requirements	PT-SYM-T01, TB-MOM-001, TB-MOM02, TB-MOM-003, TB-MOM-004		
Tools Used			

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	The Monitoring Manager'checks' the status of the resources through the Resource Controller.	The Resource Controller informs the Monitoring Manager for malfunctions of the status of UxVs	Success	Monitoring Manager implemented as subcomponent of Testbed Manager application. The use of message bus from UxVs enables the direct consumption of messages relevant to resources statuses from Monitoring Manager without the intervention of Resource Controller
2	Monitoring Manager periodically forwards the messages to the message bus	Topics about the UxVs system status are updated by Monitoring Manager	Not Tested	Messages relevant to UxVs statuses are directly produced from UxVs in the message bus and are accessible from all components without the intervention of Monitoring Manager

2.6.3.2 Network Controller

Table 86: Verification test of network interface switching due to connectivity problems

Test ID**: NC01**	Conducted by**: CSEM**	Date**: April 2017**	Test Category**: Verification Tests (middle tier)**
Hardware Configuration	See section 2.3.3		
Software Configuration	See section 2.3.3		
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.	Not Tested	Component implementation not complete
2	The Network Controller receives the incoming messages from the Resource Controller.	The appropriate network interface is selected.	Not Tested	Component implementation not complete

2.6.3.3 Resource Controller (plus Navigation Service sub-component)

Table 87: Verification test of Connection and of Accuracy validation of the given Instructions

Test ID**: RC01**	Conducted by**: CERTH**	Date**: April 2017**	Test Category**: Verification Tests (middle tier)**
Hardware Configuration	Testbed site PC		
Software Configuration	Java 8		
Test Name:	***Connection Test and Validation of the Accuracy of the Given Instructions***		
Preconditions	• The proxy should be connected to the testbed • Experiment Controller must be up and running • Requires the UxV to be ready to operate		
Related Requirements	PT-LAU-S-001, TB-PRO-001, PT-EXP-C-001, TB-MAN-001, TB-MAN-004, TB-MAN-002, TB-MAN-003, TB-MAN-005		
Tools Used			

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Receive instructions from the Experiment Controller	Instructions received	Success	
4	Send basic instructions to the UxVs	The UxV follows the instruction correctly, in order and timely, according to the specified parameters.	Success	
5	Transmit information about the progress of the current experiment back to the Experiment Controller	The experiment controller successfully receives the status of the experiment and updates the corresponding fields on the database	Success	

2.6.3.4 UxV Proximity component

Table 88: Verification test of Proximity component Backup communication

Test ID**: UxP01**	Conducted by**: CSEM**	Date**: April 2017**	Test Category**: Verification Tests (UxV tier)**
Hardware Configuration	UxV with Proximity component (CSEM WiseNode)		
Software Configuration	UxV Embedded OS + CSEM WiseNET		
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.		Ok	Tested in another context
2	The UxVs lose the connection with the primary RAWFIE communication system	The Proximity communication system takes over	Not Tested	Component implemented and unit-tested but integration with the UxVs still on-going.
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.	Not Tested	Component implemented and unit-tested but integration with the UxVs still on-going.
4	The UxVs come back and the logged information is analysed	The communication statistics exhibits low packet error rate and low latency	Not Tested	Component implemented and unit-tested but integration with the UxVs still on-going.

Table 89: Verification test of UxV retrieval using the communication system of the Proximity component

Test ID**: UxP02**	Conducted by**:**	Date**: April 2017**	Test Category**: Verification Tests (UxV tier)**
Hardware Configuration	UxV with Proximity component (CSEM WiseNode)		
Software Configuration	UxV Embedded OS + CSEM WiseNET		
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.		Ok	Tested in another context
2	The UxVs perform their mission and one of them exhausts its main power source		Not tested	
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	Not tested	
4	The other UxVs transmit the location and status of the stopped UxV to the RAWFIE resource manager		Not tested	

Table 90: Verification test of Swarm motion using the Proximity component

Test ID**: UxP03**	Conducted by**:**	Date**: April 2017**	Test Category**: Verification Tests (UxV tier)**
Hardware Configuration	UxV with Proximity component (CSEM WiseNode)		
Software Configuration	UxV Embedded OS + CSEM WiseNET		
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.		Ok	Tested in another context
2	The UxVs perform their mission moving in a coordinated fashion		Not tested	
3	The UxVs log all position		Not tested	
4	The UxVs come back and the logged information is analysed	The UxV relative locations were within the acceptable margin	Not tested	

2.6.3.5 Testbed Manager

Table 91: Verification test of Testbed Manager Experiment Handling

Test ID: **TM01**	Conducted by: **HAI**	Date: **April 2017**	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration Details	Testbed site PC		
Software Configuration Details	• Java 8 • Confluent Platform 2.01 • PostgreSQL 9.4		
Test Name:	***Testbed Manager Experiment Handling***		
Preconditions	• Requires middle tier to be accessible (Experiment Controller Service) • Requires Resource Controller running at Testbed • Requires local PostgreSQL Server accessible		
Related Requirements	TB-MAN-004 TB-MAN-001 TB-MAN-005 TB-MAN-007		
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	Success	
2	Testbed Manager receives an ExperimentStart message from Message Bus	A new experiment is registered in the local database. Testbed Manager rejects experiments not intended for this testbed	Success	
3	Testbed Manager receives an ExperimentStop message from Message Bus	The experiment is registered as successful in the experiments history log in the local database	Not Tested	The end of the experiment is perceived from Testbed Manager through the consumption of ExperimentStatusMsg message received from Resource Controller. The expected result was achieved using this message
4	Testbed Manager receives an ExperimentCancel message from Message Bus	The experiment is registered as failed / partially completed in the experiments history log in the local database	Not Tested	The cancellation of the experiment is perceived from Testbed Manager through the consumption of ExperimentStatusMsg message received from Resource Controller. The expected result was achieved using this message
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	Success	

Table 92: Verification test of Experiment management without middle-tier connection

Test ID: **TM02**	Conducted by:	Date: **April 2017**	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration Details	See section 2.3.3		
Software Configuration Details	See section 2.3.3		
Test Name:	***Manage the experiments without middle-tier connection***		
Preconditions	• Requires local PostgreSQL Server accessible		
Related Requirements	TB-MAN-008		
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	Not Tested	
2	Connection with middle-tier is lost (observed by absence of ECStatus messages received from Experiment controller in message bus)		Not Tested	
3	Testbed manager informs Resource Controller and initiates local storage mode	Resource controller enters in "emergency" mode Resource controller stores all sensor data from current active UxVs missions in the local database	Not Tested	
4	Connection with middle-tire is restored	Resource controller returns to normal mode and all sensor data are directed to RAWFIE master database	Not Tested	
5	Testbed manager sends all locally stored sensor data in the master database	Master database is updated with the missing data during middle-tier connection loss	Not Tested	

Table 93: Verification test of Check Testbed health status

Test ID: **TM03**	Conducted by: **HAI**	Date: **April 2017**	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration Details	Testbed site PC		
Software Configuration Details	• Java 8 • Confluent Platform 2.01 • PostgreSQL 9.4		
Test Name:	***Check Testbed health status***		

Preconditions	• Requires middle tier to be accessible (System Monitoring Service)
	• Initial Testbed Manager configuration:
	  ○ CPU usage WARNING > 50%, CRITICAL >90%
	  ○ Memory usage WARNING > 50%, CRITICAL >90%
	  ○ Disk usage WARNING > 50%, CRITICAL >90%
	  ○ Frequency of sending messages 30 sec		

Related Requirements	TB-MAN-003		
Tools Used			

Step	Action	Expected Result	Status	Remarks
1	Testbed Manager started	1. Testbed manager successfully initialized 2. Testbed Manager checks periodically CPU load, memory and disk usage	Success	
2	Testbed 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	Success	
3	Check System monitoring Service UI display at Middle Tier	Display of Testbed Manager status. Initial status OK	Success	
4	Artificially increase CPU or Memory usage	Status message sent to the message bus	Success	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 Manager status. Status changes to WARNING or CRITICAL	Success	
6	Decrease CPU or Memory usage and recheck System monitoring Service UI display at Middle Tier	Display of Testbed Manager status. Status changes back to OK	Success	Close extra running applications

Table 94: Verification test of Check the status of all services running at testbed level

Test ID**: TM04**	Conducted by**: HAI**	Date**: April 2017**	Test Category**: Verification Tests (Testbed tier)**
Hardware Configuration Details	Testbed site PC		
Software Configuration Details	• Java 8 • Confluent Platform 2.01 • PostgreSQL 9.4		
Test Name:	***Check the status of all services running at testbed level***		
Preconditions	• Requires middle tier to be accessible (Experiment Controller Service) • Requires Resource Controller and Network Manager running at Testbed • Requires local PostgreSQL Server accessible		
Related Requirements	TB-MAN-003 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	Success	
2	Testbed manager receives periodical status messages from Resource Controller, Network Manager and Monitoring Manager in the Message Bus		Partial success	Only Resource Controller tested. Network Manager is not implemented yet and Monitoring Manager is part of Testbed Manager (not a separate service)
3	User is able to see the availability of the components that run at testbed level by selecting the appropriate action from the menu	Show current status of components running at testbed level	Success	

2.6.3.6 UxV Node

Table 95: Verification test of UxV Return to base

Test ID**: UxV01**	Conducted by**: Rob, UoA, Certh**	Date**: 15/12/16**	Test Category**: Verification Tests (Testbed tier)**
Hardware Configuration	RobSim-SummitXL, Laser scan, IMU, camera)		
Software Configuration	RobSim-VirtualBox VM(ROS, Ubuntu 14.04,Gazebo)		
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-VIS-T-001, TB-REC-001, TB-REC-004, UXV-NET-009, UXV-SEN-003, UXV-SEN-005, UXV-PRC-001, UXV-MGT-002		
Tools Used	Network, Servers, Personal Computer, Skype		

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Establish the communication with the UxV	Communication established	OK	
2	Establish a secure control session	Secured control session established	Partial	Secured kafka bus to be implemented. The acquisition of commands is protected with a keyed message.
3	Send the return to base command	Return to base command received	OK	It is treated as a waypoint to the origin
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	OK	Either with provided waypoint for path planning or just one waypoint
5	Close the secure control session.	The UxV is home after a safe return. Connection closed	-	Secured kafka bus to be implemented.

Test ID: **UxV01**	Conducted by: **MST**	Date: **Feb 2016**	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration	rawfie.mst.auv-1, rawfie.mst.auv-2, rawfie.mst.asv-1		
Software Configuration	OceanScan Proxy 2016.02		
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-VIS-T-001, TB-REC-001, TB-REC-004, UXV-NET-009, UXV-SEN-003, UXV-SEN-005, UXV-PRC-001, UXV-MGT-002		
Tools Used	Neptus Command & Control Software		

Step	Action	Expected Result	Status	Remarks
1	Establish the communication with the UxV	Communication established	Success	
2	Establish a secure control session	Secured control session established	Partial Success	At this point only network level security is used (i.e., WPA2)
3	Send the return to base command	Return to base command received	Success	
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	Success	
5	Close the secure control session.	The UxV is home after a safe return. Connection closed	Partial Success	See remark on step 2

Table 96: Verification test of the ability of the UxV to follow a route

Test ID: **UxV02**	Conducted by: **Rob, UoA, Certh**	Date: **15/12/16**	Test Category: **Verification Tests (testbed tier)**
Hardware Configuration	RobSim-SummitXL, Laser scan, IMU, camera)		
Software Configuration	RobSim-VirtualBox VM(ROS, Ubuntu 14.04,Gazebo)		
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-VIS-T-001, TB-REC-001, TB-REC-004, UXV-NET-009, UXV-SEN-003, UXV-SEN-005, UXV-PRC-001		
Tools Used	Network, Servers, Personal Computer, Skype		

Step	Action	Expected Result	Status	Remarks
1	Resource controller computes mission and send waypoint	Robot proceeds to the specified point,	OK	Care to choose reachable waypoints
2	Robot continuously sends actual location	RC receives position and check if WP have been reached	OK	
3	RC sends next point	Robot receives and proceed to next point	OK	Reached target location with desired location must be checked carefully by RC

Test ID**: UxV02**	Conducted by**: MST**	Date**: Feb 2016**	Test Category**: Verification Tests (testbed tier)**
Hardware Configuration	rawfie.mst.auv-1, rawfie.mst.auv-2, rawfie.mst.asv-1		
Software Configuration	OceanScan Proxy 2016.02		
Test Name:	***Follow a route***		
Preconditions	1 Requires the RAWFIE system to be operational (e.g. Resource controller reachable) 2 Requires the mission to be defined and running. 3 Requires the UxV to be ready to operating (e.g. en route). 4 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-VIS-T-001, TB-REC-001, TB-REC-004, UXV-NET-009, UXV-SEN-003, UXV-SEN-005, UXV-PRC-001		
Tools Used	Neptus Command & Control Software		

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

Table 97: Verification test of Acquire sensor samples

Test ID: **UxV03**	Conducted by: **Rob, UoA, Certh**	Date: **15/12/16**	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration	RobSim-SummitXL, Laser scan, IMU, camera)		
Software Configuration	RobSim-VirtualBox VM(ROS, Ubuntu 14.04,Gazebo)		
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-005, UXV-STO-001, UXV-STO-002, UXV-NET-006, UXV-NET-007, 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		
Tools Used	Network, Servers, Personal Computer, Skype		

Step	Action	Expected Result	Status	Remarks
1	Establish the communication with the UxV	Communication established	OK	
2	Establish a secure control session (if not done already)	Secured control session established	Partial	Secured kafka bus to be implemented. The acquisition of commands is protected with a keyed message.
3	Send the acquisition commands	Commands received and executed	OK	Set of commands to be completed
4	Store sensor samples and, if possible, transmit them via the data communication system	Samples stored and, if possible, transmitted	Partial	Command start/stop broadcast received. Command for storage to be implemented
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	Partial	Secured kafka bus to be implemented. Connection not closed. Listener stops reading the bus.

Test ID: **UxV03**	Conducted by: **MST**	Date: **Feb 2016**	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration	rawfie.mst.auv-1, rawfie.mst.auv-2, rawfie.mst.asv-1		
Software Configuration	OceanScan Proxy 2016.02		
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-005, UXV-STO-001, UXV-STO-002, UXV-NET-006, UXV-NET-007, 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		
Tools Used	Neptus Command & Control Software		

Step	Action	Expected Result	Status	Remarks
1	Establish the communication with the UxV	Communication established	Success	
2	Establish a secure control session (if not done already)	Secured control session established	Partial Success	At this point only network level security is used (i.e., WPA2)
3	Send the acquisition commands	Commands received and executed	Success	Output of sensors is controlled via the SensorPublishControl message.
4	Store sensor samples and, if possible, transmit them via the data communication system	Samples stored and, if possible, transmitted	Success	
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	Partial Success	See remark on step 2

Table 98: 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		
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		

Test ID: **UxV04**	Conducted by: **MST**	Date: **Feb 2016**	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration	rawfie.mst.auv-1, rawfie.mst.auv-2, rawfie.mst.asv-1		
Software Configuration	OceanScan Proxy 2016.02		
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		
Tools Used	Neptus Command & Control Software		

Step	Action	Expected Result	Status	Remarks
1	Establish the communication with the UxV	Communication established	Success	
2	Establish a secure control session (if not done already)	Secured control session established	Partial Success	At this point only network level security is used (i.e., WPA2)
3	Send repeatedly pre-defined sets of commands, covering the full range of possible UxV actions,	Commands received and executed	Success	
4	Check the conformance of the undertaken actions and corrections (if necessary) to the commands,	Undertaken actions in conformance to the commands	Success	
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	Success	
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	Partial Success	See remark on step 2

Table 99: Verification test of Continuous communication

Test ID**: UxV05**	Conducted by**: Rob, UoA, Certh**	Date**: 15/12/16**	Test Category**: Verification Tests (Testbed tier)**
Hardware Configuration	RobSim-SummitXL, Laser scan, IMU, camera)		
Software Configuration	RobSim-VirtualBox VM(ROS, Ubuntu 14.04,Gazebo)		
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		
Tools Used	Network, Servers, Personal Computer, Skype		

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

Test ID: **UxV05**	Conducted by: **MST**	Date: **Feb 2016**	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration	rawfie.mst.auv-1, rawfie.mst.auv-2, rawfie.mst.asv-1		
Software Configuration	OceanScan Proxy 2016.02		
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		
Tools Used	Neptus Command & Control Software		

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

Table 100: Verification test of Continuous communication

Test ID: **UxV06**	Conducted by: **MST**	Date: **Feb 2016**	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration	rawfie.mst.auv-1, rawfie.mst.auv-2, rawfie.mst.asv-1		
Software Configuration	OceanScan Proxy 2016.02		
Test Name:	***Continuous 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		
Tools Used	Neptus Command & Control Software		

Step	Action	Expected Result	Status	Remarks
1	Establish the communication with the UxV	Communication established	Success	
2	Establish a secure control session (if not done already)	Secured control session established	Partial Success	At this point only network level security is used (i.e., WPA2)
3	Check communication parameters	Communication parameters and status are correct and matching	Success	
4	Exchange a pre-defined set of commands and data,	Commands and data correctly exchanged	Success	
5	Close the communication session.	Communication closed	Success	

Table 101: Verification test of Secure communication

Test ID: **UxV07**	Conducted by: **MST**	Date: **Feb 2016**	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration	rawfie.mst.auv-1, rawfie.mst.auv-2, rawfie.mst.asv-1		
Software Configuration	OceanScan Proxy 2016.02		
Test Name:	***Secure 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		
Tools Used	Neptus Command & Control Software		

Step	Action	Expected Result	Status	Remarks
1	Establish the communication with the UxV	Communication established	Success	
2	Establish a secure control session (if not done already)	Secured control session established	Partial Success	At this point only network level security is used (i.e., WPA2)
3	Send safe commands and measure the temporal characteristics of the communication (e.g. response time, synchronization 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	Success	The time of flight of messages is greater when the producer registers with the message bus, sometimes reaching more than 10 seconds. This latency is perfectly tolerated by MST vehicles
4	Close the secure control session.	Connection closed	Partial Success	See remark on step 2

Table 102: Verification test of Real-time communication

Test ID: **UxV08**	Conducted by: **MST**	Date: **Feb 2016**	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration	rawfie.mst.auv-1, rawfie.mst.auv-2, rawfie.mst.asv-1		
Software Configuration	OceanScan Proxy 2016.02		
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. - 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		
Tools Used	Neptus Command & Control Software		

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

Table 103: Verification test of UxV Device Management

Test ID: **UxV09**	Conducted by: **Rob**	Date: **20/04/2016**	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration	Summit XL		
Software Configuration	ROS Indigo, Ubuntu 14.04		
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		
Tools Used	Secured Remote Desktop Application		

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

Test ID**: UxV9**	Conducted by**: MST**	Date**: Feb 2016**	Test Category**: Verification Tests (Testbed tier)**
Hardware Configuration	rawfie.mst.auv-1, rawfie.mst.auv-2, rawfie.mst.asv-1		
Software Configuration	OceanScan Proxy 2016.02		
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		
Tools Used	Neptus Command & Control Software		

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Establish the communication with the UxV	Communication established	Success	
2	Establish a secure control session (if not done already)	Secured control session established	Partial Success	At this point only network level security is used (i.e., WPA2)
3	Send device management commands	Command received and applied	Success	
4	Check and log the status of the device	Device has responded to the commands according to the specification	Success	
5	Close the secure control session.	The UxV is home after a safe return. Connection closed	Partial Success	See remark on step 2

Table 104: Verification test of the UxV connection

Test ID: **UxV10**	Conducted by: **Rob, UoA, Certh**	Date: 27/2/2016	Test Category**: Verification Tests (testbed tier)**
Hardware Configuration	Summit XL		
Software Configuration	Ros Indigo, Ubuntu 14.04		
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	**Robot, Porto MST Facilities Network, PC**		

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

Test ID: **UxV10**	Conducted by: **MST**	Date: **Feb 2016**	Test Category**: Verification Tests (testbed tier)**
Hardware Configuration	rawfie.mst.auv-1, rawfie.mst.auv-2, rawfie.mst.asv-1		
Software Configuration	OceanScan Proxy 2016.02		
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	OceanScan Proxy 2016.02 Testsuit		

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

Table 105: Verification test of Sensor Data Acquisition 1

Test ID: **UxV11**	Conducted by: **Rob, UoA, Certh**	Date: 27/2/2016	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration	Summit XL		
Software Configuration	Ros Indigo, Ubuntu 14.04		
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		
Tools Used	**Robot, Porto MST Facilities Network, PC**		

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

Test ID: **UxV11**	Conducted by: **MST**	Date: **Feb 2016**	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration	rawfie.mst.auv-1, rawfie.mst.auv-2, rawfie.mst.asv-1		
Software Configuration	OceanScan Proxy 2016.02		
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		
Tools Used	Neptus Command & Control Software		

Step	Action	Expected Result	Status	Remarks
1	Establish the communication with the UxV	Communication established	Success	
2	Establish a secure control session (if not done already)	Secured control session established	Partial Success	At this point only network level security is used (i.e., WPA2)
3	Acquire sensor data	Data acquired (every sensor works as specified)	Success	Individual sensor data is tested
4	Send acquired data	Data received	Success	Provides data gathered by each sensor placed on the robot. Data streamed of every sensor is tested individually
5	Close the secure control session.	The UxV is home after a safe return. Connection closed	Partial Success	See remark on step 2

Table 106: Verification test of Sensor Data Acquisition 2

Test ID**: UxV12**	Conducted by**: Rob, UoA, Certh**	Date**:** 27/2/2016	Test Category**: Verification Tests (Testbed tier)**
Hardware Configuration	Summit XL		
Software Configuration	Ros Indigo, Ubuntu 14.04		
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		
Tools Used	**Robot, Porto MST Facilities Network, PC**		

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

Test ID: **UxV12**	Conducted by: **MST**	Date: **Feb 2016**	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration	rawfie.mst.auv-1, rawfie.mst.auv-2, rawfie.mst.asv-1		
Software Configuration	OceanScan Proxy 2016.02		
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		
Tools Used	Neptus Command & Control Software		

Step	Action	Expected Result	Status	Remarks
1	Establish the communication with the UxV	Communication established	Success	
2	Establish a secure control session (if not done already)	Secured control session established	Partial Success	At this point only network level security is used (i.e., WPA2)
3	Instruct the robot to move to a know location	Robot at the specific location	Success	Robot is moved to a precisely located point and a comparison is done later
4	Acquire current location data	Location data acquired (location sensor works as specified)	Success	Localization of the robot is tested.
5	Send acquired location data	Data received	Success	Provides data about the location of the robot. Location is compared to known location.
6	Close the secure control session.	The UxV is home after a safe return. Connection closed	Partial Success	See remark on step 2

Test ID: **UxV13**	Conducted by: **MST**	Date: **Feb 2016**	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration	rawfie.mst.auv-1, rawfie.mst.auv-2, rawfie.mst.asv-1		
Software Configuration	OceanScan Proxy 2016.02		
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		
Tools Used	Neptus Command & Control Software		

Step	**Action**	**Expected Result**	**Status**	**Remarks**
1	Establish the communication with the UxV	Communication established	Success	
2	Establish a secure control session (if not done already)	Secured control session established	Partial Success	At this point only network level security is used (i.e., WPA2)
3	A request for storing certain data is done	Command received and data is stored locally	Partial Success	At this point no such command exists and the UxVs will store all data
4	After a given mission, data storage in the system is checked.	Data was correctly stored and kept.	Success	The data is stored and identified in the robot system
5	Close the secure control session.	The UxV is home after a safe return. Connection closed	Partial Success	See remark on step 2

Table 107: Verification test of Waypoints Processed

Test ID**: UxV14**	Conducted by**: Rob, UoA, Certh**	Date**: 15/12/16**	Test Category**: Verification Tests (Testbed tier)**
Hardware Configuration	RobSim-SummitXL, Laser scan, IMU, camera)		
Software Configuration	RobSim-VirtualBox VM(ROS, Ubuntu 14.04,Gazebo)		
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,		
Tools Used	Network, Servers, Personal Computer, Skype		

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

Test ID: **UxV14**	Conducted by: **MST**	Date: **Feb 2016**	Test Category: **Verification Tests (Testbed tier)**
Hardware Configuration	rawfie.mst.auv-1, rawfie.mst.auv-2, rawfie.mst.asv-1		
Software Configuration	OceanScan Proxy 2016.02		
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,		
Tools Used	Neptus Command & Control Software		

Step	Action	Expected Result	Status	Remarks
1	Establish the communication with the UxV	Communication established	Success	
2	Establish a secure control session (if not done already)	Secured control session established	Partial Success	At this point only network level security is used (i.e., WPA2)
3	Waypoints are sent to the UxV	UxV receives and processes the waypoints	Success	Semi-autonomous mission is tested. The UxV has to process a set of waypoints and move to each waypoint in sequence. The UxV processes the data.
4	The calculated route is applied to the UxV	The actual trajectory matches the route calculated by the navigation.	Success	
5	Iterate step 4 until assessment is complete	UxV stops, informs and recalculate its route to next waypoint if an unexpected obstacle is found.	Not Tested	The UxVs used in this test are not equipped with obstacle avoidance systems.
6	Close the secure control session.	The UxV is home after a safe return. Connection closed	Partial Success	See remark on step 2

2.7 Benchmarking of different Message Bus topologies and configurations

2.7.1 Purpose

The message bus is a key element of the RAWFIE system, both from the point of view of the features and of the performance. Benchmarking kafka on reference platforms will give valuable and reliable indications for the dimensioning of the RAWFIE system so that, in similar conditions, it can increase the chances for meeting the time constraints during most of the experimentation execution.

2.7.2 Scenarios and setup

The detailed description of the test setup, kafka configuration and other hardware and software parameters are given in section 3.2.4 of deliverable D4.7. The next paragraphs give the most important aspects of the considered scenarios. Scenario A corresponds to a Single centralised Apache Kafka Broker. The scenario B corresponds to Multiple Apache Kafka Brokers with the

same topics on each different Testbed. The scenario C corresponds to the Multiple Apache Kafka Brokers with different topics per testbed.

For scenario A, a Kafka cluster with 4 nodes was created. All VMs were running in 2GB RAM. Every VM was running a producer and a consumer. Jconsole was used for collecting metrics and exporting them.

For scenario B and C a cluster of 5 computers with 3 Kafka nodes and 3 Zookeeper instances were used. Acting as the simulated Testbed environments 2 Virtual Machines each in a different network were connected to the internet with a regular ADSL connection. In scenarios B and C, all the messages were sent in the VPN network as was established in all testbeds for security reasons.

For Scenario A the metrics described in the following were collected. This is the complete result set for 1000 records. All messages were sent to one topic from the same remote machine (i.e., running on a different country than the Kafka server). The consumer and producer run on separate threads. Each dispatched record contains a timestamp that can be used to measure the round-trip time (RTT). Two scenarios were tested:

a) burst produce/consume: the producer dispatches a burst of 1000 records back to back to the message bus and the elapsed time is recorded (TX). The consumer reads those 1000 records from the message as soon as they are available and the elapsed time is recorded (RX). In this scenario we try to measure the latency characteristics of records that are not used for automatic control of UxVs (i.e., payload sensor data, basic telemetry) and therefore will not trigger any reply.
b) synchronous produce/consume: the producer dispatches one record to the message bus and the elapsed time is recorded (TX) it then waits for the consumer to read the record from the message bus and this elapsed time is recorded (RX). In this scenario we try to measure the latency characteristics of records that may trigger a reply (i.e. waypoint references).

2.7.3 Results

Table 108 summarises the execution performance of kafka in the two metrics in the scenario A. The test runs over more than 100s and 20s respectively.

Table 108: Sync and Burst cased tested in scenario A

	Sync Test (TX/RX) | 1000 records	**Burst Test (TX/RX) | 1000 records**
Subscribed Topics	1	1
Elapsed Time	113226 ms	21662 ms
Schema Initialization	8 ms	11 ms
Kafka Producer Initialization	3 ms	3 ms

Kafka Consumer Initialization	5266 ms	5075 ms
Kafka Consumer Shutdown	0 ms	611 ms

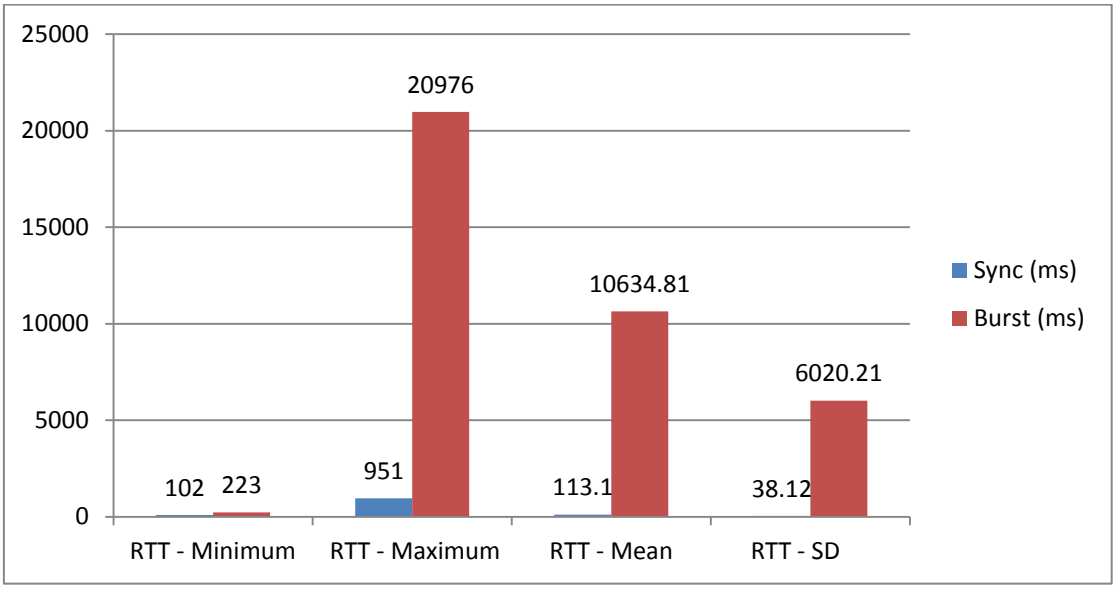


Figure 7 – Round Trip Time metrics in scenario A

Note: Y axis is duration in millisecond.

In the burst test, which results are displayed in Figure 7, the producer does not wait for the consumer to complete. The Round Trip Time is measured using the timestamp in the transmitted/received record. The interpretation of the observed phenomenon is that the first dispatched messages takes longer to return to the consumer than the next dispatched messages. This is usually due to on-demand resource allocation, routing, queue establishment, handshaking, etc. to which kafka may be also sensitive.

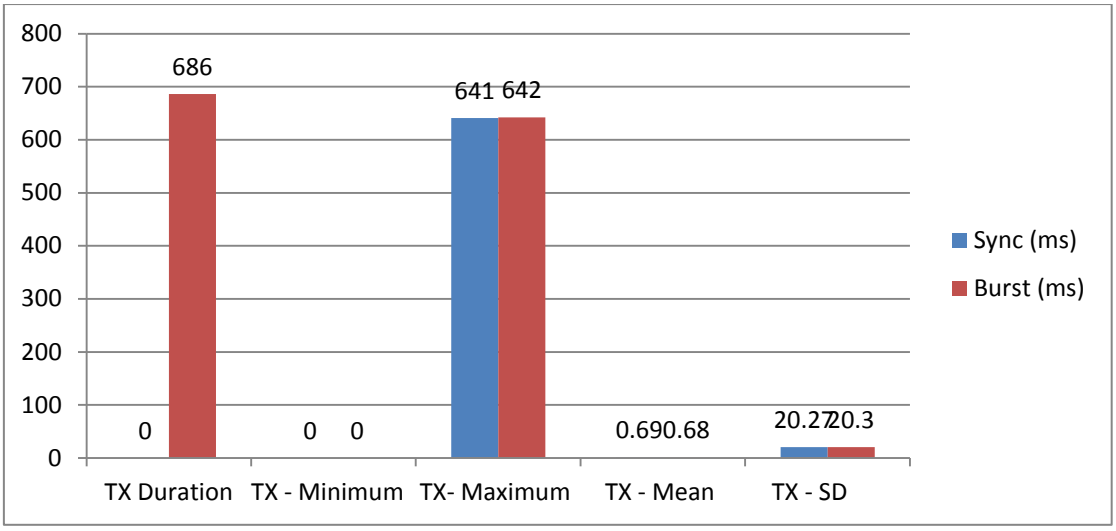


Figure 8: TX metrics in Scenario A

Note: Y axis is always duration in millisecond.

The TX duration on Figure 8 is the time it takes to pass the message to the Kafka infrastructure. Only the producer side is accounted for.

For scenarios B and C, Kafka metrics from the TotalTimeMs family were collected. Each virtual machine was running 1 Kafka broker and in the case of the third scenario 1 Zookeeper instance. In each scenario, we had 2 producers sending 50 messages per second and 10 consumers running locally in every VM, emulating the traffic in a Testbed environment where UxV devices performing the produce and consume operations pointed to their local broker. For the third scenario we also had the Apache Kafka Mirror Maker tool performing the mirroring from the virtual machine's broker to the cluster located in the UoA premises . TotalTimeMs is the total time taken to service a request (be it a produce, fetch-consumer, or fetch-follower request) from Jconsole. The TotalTimeMs measurement itself is the sum of four metrics:

- queue: time spent waiting in the request queue
- local: time spent being processed by leader
- remote: time spent waiting for follower response (only when requests.required.acks=-1)
- response: time to send the response

TotalTimeMs family of metrics provide measurements for different requests in a Kafka cluster. These are:

- produce: requests from producers to send data
- fetch-consumer: requests from consumers to get new data
- fetch-follower: requests from brokers that are the followers of a partition to get new data

We used the produce and fetch-consumer measurements in each scenario and the results are shown bellow

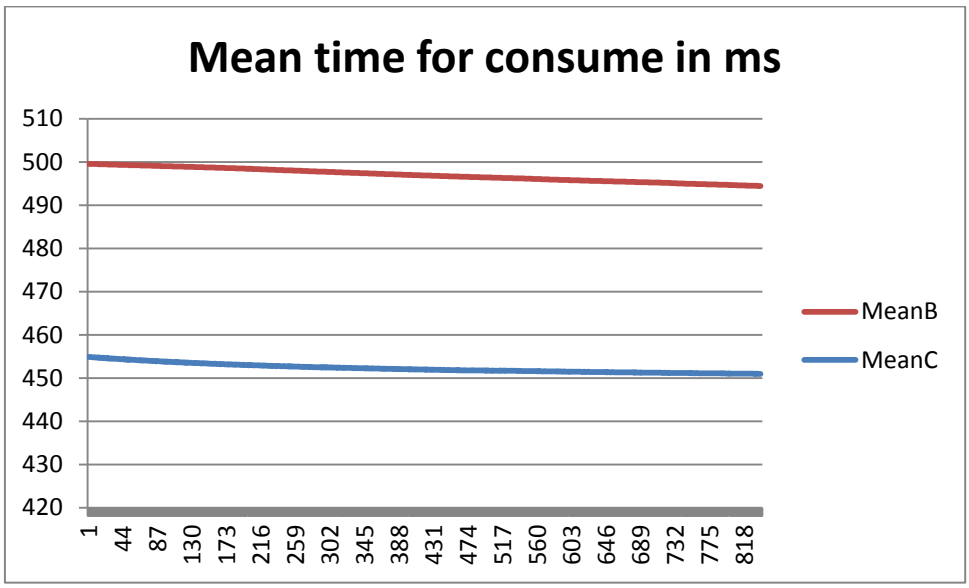


Figure 9: Mean Time for consuming messages in Scenarios B and C

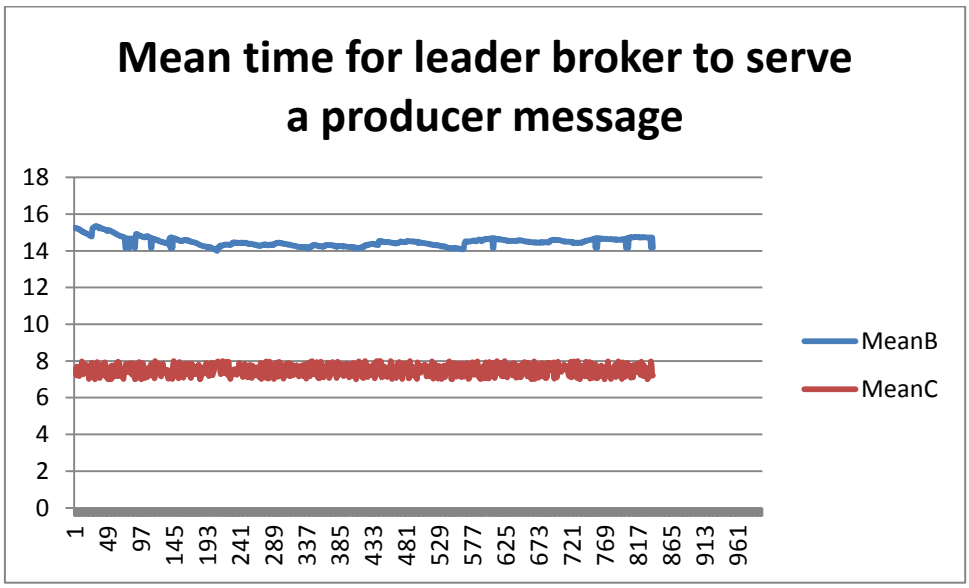


Figure 10: Mean Time for leader broker to serve messages in Scenarios B and C

Figure 7 shows the results of the consumer measurements from the time that a consumer sends a request to consume from a partition in the Kafka broker until it's request is serviced

Figure 8 shows the results of the producer measurements from the time a producer sends a produce request to the time the leader broker in the UoA Kafka cluster send a response that the produce request was completed.

From the figures above we can notice that the time for serving a produced message is lower in scenario C than in the related values in scenarios A and B. This was expected because the broker in its testbed is assigned to handle a bunch of messages produced and consumed by a small number of the devices. The small amount of partitions enhances the handling of the messages between the entities.

2.8 Deviations with respect to D6.1

In this deliverable we follow an approach based on the enrichment of the previous experience and documentation, in order to focus on the new or extended integration and verification results, as derived from the first iteration. The second development cycle included integration and verification actions for the new components developed by partners or by the third parties of ROC1.

The Integration and Testing section was enriched with the descriptions of the infrastructure and the tools used by the partners for implementing the methodology described in the relevant section of D6.1. Integration and verification test results have been grouped to this version to the tree tiers described in the RAWFIE architecture, i.e. front-end, middle and testbed tier. In addition we present the results of verification and integration by using the same template tables proposed in D6.3.

Part III: Conclusion & Roadmap

The RAWFIE integration process is still under consolidation. Numerous technical options and tools have been experimented, stabilised and validated (the kafka configurations, the coordinate system) and others are still being considered as potential candidates or under evaluation (proximity component, network controller, etc.). The currently integrated RAWFIE platform can be deployed in selected sites under the guidance of the RAFIE consortium, with a number of non-blocking restrictions (e.g. only one network connection defined at startup, etc.). This second platform is used for getting the feedback of the professional users involved in Open calls. This includes UxV and Testbed owners.

The components have been tested individually as well as once integrated. The consortium has tested and evaluated the performance of the underlying infrastructure, in particular the message, to make sure that it meets the requirements of typical installations.

The next period will focus on the validation of the missing components or components that have not been yet fully tested to have a fully validated reference RAWFIE platform that can be easily and safely deployed at large. This platform will be made widely available for extended use, for covering the maximum of use cases for validating all features and characteristics of RAWFIE. In parallel, the integration of external entities and the customization of the RAWFIE will be supported. Later, the RAWFIE consortium may address the continuous feedback given by the RAWFIE stakeholders (experimenters, resource providers, third parties, regulatory bodies, etc.) to support a semi-automatic notification and improvement process.

Part IV: Annex

Annex A Glossary

The RAWFIE glossary consists of generic terms, contributed by all partners, used across the entire RAWFIE project.

A

Accounting Service

RAWFIE component. Component that keeps track of resources usage by individual users.

Aggregate Manager

Slice Federation Architecture (SFA) term. The Aggregate Manager API is the interface by which experimenters discover, reserve and control resources at resource providers.

Avro

Apache Avro: a remote procedure call and data serialization framework

B

Booking Service

RAWFIE component. The Booking Service manages bookings of resources by registering data to appropriate database tables.

Booking Tool

RAWFIE component. The Booking tool will provide the appropriate Web UI interface for the experimenter to discover available resources and reserve them for a specified period.

C

Common Testbed Interface

RAWFIE component. The set of software and hardware functionalities each Testbed provider should ensure, for the communication with Middle Tier software components of RAWFIE, therefore for the integration with the RAWFIE platform

Component

A reusable entity that provides a set of functionalities (or data) semantically related. A component may encapsulate one or more modules (see definition) and should provide a well defined API for interaction

D

Data Analysis Engine

RAWFIE component. The Data Analysis Engine enables the execution of data processing jobs by sending requests to a processing engine which will perform the computations specified when the analytical task was defined through the Data Analysis Tool to be transmitted to the processing engine for execution.

Data Analysis Tool

RAWFIE component. The Data Analysis Tool enables the user to browse available data sources for subject to analytical treatment as well as previous analysis tasks' outcomes.

E

EDL Compiler & Validator

RAWFIE component. The EDL validator will be responsible for performing syntactic and semantic analysis on the provided EDL scripts.

Experiment Authoring Tool

RAWFIE component. This component is actually a collection of tools for defining experiments and authoring EDL scripts through RAWFIE web portal. It will provide features to handle resource requirements/configuration, location/topology information, task description etc.

Experiment Controller

RAWFIE component. The Experiment Controller is a service placed in the Middle tier and is responsible to monitor the smooth execution of each experiment. The main task of the experiment controller is the monitoring of the experiment execution while acting as ‘broker’ between the experimenter and the resources.

Experiment Monitoring Tool

RAWFIE component. Shows the status of experiments and of the resources used by experiments.

Experiment Validation Service

RAWFIE component. The Experiment Validation Service will be responsible to validate every experiment as far as execution issues concern.

M

Master Data Repository

RAWFIE component. Repository that stores all main entities that are needed in the RAWFIE platforms. Is an SQL-database

Measurements Repository

RAWFIE component. Stores the raw measurements from the experiments

Message Bus

Also known as Message Oriented Middleware. A message bus is supports sending and receiving messages between distributed systems. It is used in RAWFIE across all tiers to enable asynchronous, event-based messaging between heterogeneous components. Implements the Publish/Subscribe paradigm.

Module

A set of code packages within one software product that provides a special functionality

Monitoring Manager

RAWFIE component. Monitors the status of the testbed and the UxVs belonging to it, at functional level, e.g. the 'health of the devices' and current activity.

N

Network Controller

Manages the network connections and the switching between different technologies in the testbed in order to offer seamless connectivity in the operations of the system.

L

Launching Service

RAWFIE component. The Launching Service is responsible for handling requests for starting or cancellation of experiments.

R

Resource Controller

RAWFIE component. The Resource Controller can be considered as a cloud robot and automation system and ensures the safe and accurate guidance of the UxVs.

Resource Explorer Tool

RAWFIE component. The experimenter can discover and select available testbeds as well as resources/UxVs inside a testbed with this tool. Administrators can manage the data.

Results Repository

RAWFIE component. Stores the results of data analyses.

Resource Specification (RSpec)

SFA term. This is the means that the SFA uses for describing resources, resource requests, and reservations (declaring which resources a user wants on each Aggregate).

S

Schema Registry

A schema registry is a central service where data schemas are uploaded to. As an added benefit each schema has versions with it can convert allowable formats to other ones (e.g.: float to double) It maintains schemas for the data transferred and keeps revisions to be able to upgrade the definitions as with the simple field conversion. Used in RAWFIE for messages on the message bus.

Service

A component that is running in the system, providing specific functionalities and accessible via a well known interface.

Slice Federation Architecture (SFA)

SFA is the de facto standard for testbed federation and is a secure, distributed and scalable narrow waist of functionality for federating heterogeneous testbeds.

Subsystem

A collection of components providing a subset of the system functionalities.

System

A collection of subsystems and/or individual components representing the provided software solution as a whole.

System Monitoring Service

RAWFIE component. Checks readiness of main components and ensure that all critical software modules will perform at optimum levels. Predefined notification are triggered whenever the corresponding conditions are met, or whenever thresholds are reached

System Monitoring Tool

RAWFIE component. Shows the status and the readiness of the various RAWFIE services and testbed

T

Testbed

A testbed is a platform for conducting rigorous, transparent, and replicable testing of scientific theories, computational tools, and new technologies.

In the context of RAWFIE, a testbed or testbed facility is a physical building or area where UxVs can move around to execute some experiments. In addition, the UxVs are stored in or near the testbed.

Testbeds Directory Service

RAWFIE component. Represents a registry service of the middleware tier where all the integrated testbeds and resources accessible from the federated facilities are listed, belonging to the RAWFIE federation.

Testbed Manager

RAWFIE component. Contains accumulated information about the UxVs resources and the experiments of each one of the federation testbeds.

Tool

A GUI implementation to do a special thing, e.g. the “Resource Explorer tool” to search for a resource

U

Users & Rights Repository

RAWFIE component. Management of users and their roles. Is a directory services (LDAP).

Users & Rights Service

RAWFIE component. Manages all the users, roles and rights in the system.

UxV

The generic term for unmanned vehicle. In RAWFIE, it can be either:

USV - Unmanned Surface vehicle.

UAV - Unmanned Aerial vehicle.

UGV - Unmanned Ground vehicle.

UUV - Unmanned Underwater vehicle.

UxV Navigation Tool

RAWFIE component. This component will provide to the user the ability to (near) real-time remotely navigate a squad of UxVs.

UxV node

RAWFIE component. A single UxV node. The UxV is a complete mobile system that interacts with the other Testbed entities. It can be remotely controlled or able to act and move autonomously.

V

Visualisation Engine

RAWFIE component. Used for providing the necessary information to the Visualisation tool, to communicate with the other components, to handle geospatial data, to retrieve data

for experiments from the database, to load and store user settings and to forward them to the visualisation tool.

Visualisation Tool

RAWFIE component. Visualisation of an ongoing experiment as well as visualisation of experiments that are already finished

W

Web Portal

RAWFIE component. The central user interface that provides access to most of the RAWFIE tools/services and available documentation.

Wiki Tool

RAWFIE component. Provides documentation and tutorials to the users of the platform.

Annex B Requirements

The requirements listed in Table 109: Requirements considered for the integration are considered in the context of the integration.

Table 109: Requirements considered for the integration

PT-WEB-P-001	A web portal interface shall be provided to the users of the platform to access almost all main functionalities.
PT-BOO-T-003	Booking Tool should delegate all its actions related to Booking of a resource to the Booking Service
PT-BOO-T-004	Booking Tool may also interact with the Testbeds Directory Service in order to retrieve information on unallocated testbed resources
PT-REE-T-004	Link to the Booking Tool should be provided
PT-EXM-T-003	Cancellation of running experiments should be possible via Web Portal
PT-VIS-T-002	A 3D visualization should be available for the tracking of all moving resources
PT-VIS-T-004	The Visualisation Tool shall provide access to information / features associated to each UxV device on the geographic map
PT-DAA-T-001	Analysis tool will provide interface to data engine.
PT-DAA-T-002	Analysis tool will provide ability to query available data schemas
PT-DAA-T-003	Analysis tool will be able to read results from Results Database
PT-DAA-E-001	Analysis Engine will be able to query message bus streams
PT-DAA-E-001	Analysis Engine will be able to receive messages from Analysis Tool
PT-DAA-E-002	Analysis Engine will be able to write data to the Results Database
PT-DIR-S-007	The Testbed Directory Service shall provide the possibility to register new resources belonging to a specific testbed in the RAWFIE platform, as well as to unregister (delete) resources
PT-CPV-001	A tool for translating EDL into user directives shall be provided
PT-CPV-002	An experimenter should have the opportunity to use a code generation engine
PT-CPV-003	Experiments defined via EDL shall be validated after their authoring
PT-CPV-004	The compiler and validator should communicate with the authoring tool in order to transfer error indications and hints for solving them
PT-BOO-S-006	Booking Service should be able to compute and return feedback on conflicting bookings for a provided booking request
PT-LAU-S-001	Launching Service should support short-term or manual launching of an experiment initiated directly by an experimenter
PT-VIS-E-001	The Visualization Engine shall handle the communication with the Message Bus, for the information that will be coming from the UxVs
PT-EXP-C-002	RAWFIE platform shall allow experimenters to remotely navigate UxVs.
PT-EXP-C-006	The Experiment Controller shall support receiving feedback at regular intervals from all testbed facilities about the progress of the experiment in this time interval

PT-EXP-C-008	The Experiment Controller shall be able to continuously feed the front-end tier (Experiment Monitoring Tool) giving the experimenter a clear view of the experiment workflow as a whole
PT-EXA-T-001	Experiment Description Language (EDL) shall be used as a language for the definition of experiment scenarios
PT-EXA-T-002	The EDL shall allow the definition of all necessary requirements for an experiment
PT-EXA-T-003	For each defined experiment specific metadata, i.e. name, version, date and description shall be defined.
PT-EXA-T-004	An experimenter shall be able to provide initial conditions and/or configuration parameters for an experiment
PT-EXA-T-005	An experimenter shall be able to manage/guide the available booked resources during experiment authoring
PT-EXA-T-008	An experimenter shall be able to provide navigation or movement directives during experiment authoring
PT-EXA-T-009	An experimenter should be able to create groups of UxVs resources, for which specific directives will apply.
PT-EXA-T-010	A textual editor shall be provided for the authoring of RAWFIE experiments
PT-EXA-T-011	A visual/graphical editor shall be provided for the authoring of RAWFIE experiments
PT-EXA-T-012	Platform shall allow saving, editing and/or deletion of an experiment defined via EDL
PT-EXA-T-013	The visual editor should allow the definition of movement and location waypoints from a map
PT-EXA-T-015	Validation of EDL script should be possible prior to or during saving
PT-EXV-S-001	RAWFIE shall provide a validator to constantly check experiment scenarios during runtime
PT-EXV-S-002	The validation service should perform syntactic checking
PT-EXV-S-003	The validation service should perform semantic checking
TB-MOM-004	Testbed monitoring manager should be able to transmit the current status to the System Monitoring Service.
TB-REC-003	The Resource Controller shall receive location messages from the vehicles at regular intervals
TB-REC-005	For the experiment accomplishment the Resource Controller shall operate in close coordination with the Experiment Controller
TB-MAN-005	Testbed Manager shall be periodically informed about the status of all running experiments in the testbed
UXV-NET-006	UxV communication interoperability with RAWFIE (incoming)
UXV-NET-007	UxV communication interoperability with RAWFIE (outgoing)
UXV-SEN-005	UxVs should sent a notification to the Resource Controller when they reach the desired location

References

[1] Xtext: https://eclipse.org/Xtext/index.html
[3] OpenLayers: http://openlayers.org/