



## PROJECT DELIVERABLE REPORT



Introducing advanced ICT  
and Mass Evacuation Vessel design  
to ship evacuation and rescue systems

### **D7.6 Test cases and overall system testing results (v2)**

A holistic passenger ship evacuation and rescue ecosystem

MG-2-2-2018

Marine Accident Response

*"This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 814962"*



**Document Information**

Grant Number	Agreement	814962	Acronym	PALAEMON
Full Title	A holistic passenger ship evacuation and rescue ecosystem			
Topic	MG-2-2-2018: Marine Accident Response			
Funding scheme	RIA - Research and Innovation action			
Start Date	1 <sup>st</sup> JUNE 2019	Duration	36 months	
Project URL	www.palaemonproject.eu			
EU Project Officer	Georgios CHARALAMPOUS			
Project Coordinator	AIRBUS DEFENCE AND SPACE SAS			
Deliverable	D7.6 Test cases and overall system testing results (v2)			
Work Package	WP7 – PALAEMON Integrated System and Technology Validation Trials			
Date of Delivery	Contractual	M36	Actual	M43
Nature	R – Report	Dissemination Level		PU - Public
Lead Beneficiary	Atos Spain S.A.			
Responsible Author	David Gómez	Email	david.gomez@atos.net	
		Phone	N/A	
Reviewer(s):	Vassilis Chatzigiannakis (IMTL), Kyriakos Giannakis (KT)			
Keywords	Communications Platform, Testing, Results, Integration			

**Authors List**

<b>Name</b>	<b>Organization</b>
David Gómez, Julia Ruiz	ATOS
Dimitrios Kaklis, Artemis Flori	DANAOS
Kyriakos Giannakis	KT
Jens Hübel	JU
Nikos Triantafyllou	UAEG
Bogdan Gornea, Marius Curca	SIMAVI
Javier Peña, Manuel Ramiro	ADV
Alfredo Gardel	UAH
Anastasia Danopoulou, Alexandros Koimtzoglou	NTUA
Elias Chatzidouros, Antonis Chronakis	ESI
Francesco Piantini	THALIT

## Revision History

Version	Date	Responsible	Description/Remarks/Reason for changes
0.1	2021/01/04	ATOS	ToC first draft and initial list of contributors
0.2	2022/09/05	ALL	First round of inputs from partners
0.3	2022/09/15	ATOS	Section 2 (Software development)
0.4	2022/09/30	ATOS	Section 2 (Docker and Kubernetes)
0.5	2022/11/13	ATOS	Section 3 (Evacuation Process)
0.6	2022/06/15	ATOS	Section 4 (Cross-validation)
0.7	2022/06/23	ATOS	Section 5 (Individual testing)
0.8	2022/11/20	ATOS	Updated figures from testing campaign (hands-on sessions)
0.85	2022/12/12	ATOS	Intro + conclusions
0.9	2022/12/21	ITML, KT	Internal review
0.95	2022/12/2	ATOS	Update with internal reviewers' feedback
1.0	2022/12/23	ATOS	Review and Release (submitted to EC portal)

*Disclaimer: Any dissemination of results reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.*

**© PALAEMON Consortium, 2022**

*This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both. Reproduction is authorised provided the source is acknowledged.*

## Contents

1	Introduction .....	9
2	Software development and integration methodology .....	10
2.1	Docker & Kubernetes .....	11
2.2	GitLab Issues and Branches Management .....	15
2.3	GitLab Continuous Integration / Continuous Deployment (CI/CD) .....	17
3	Evacuation Process Management.....	19
4	Overall system testing results .....	22
4.1	DFB (Apache Kafka & Elasticsearch) as system validator.....	22
4.2	PALAEMON Evacuation Coordinator as system validator .....	23
4.3	Kibana (Canvas) as system validator .....	24
4.4	PIMM as system validator .....	28
4.5	Grafana as system validator .....	31
4.6	Voyage Report Generator as system validator .....	32
5	Individual validation.....	37
5.1	Smart Bracelets .....	37
5.2	Smart Cameras.....	40
5.3	Smart Safety System .....	41
5.4	VDES transceiver and gateway.....	43
5.5	Safety Management System .....	44
5.6	Ship Health Monitoring.....	46
5.7	Ship Stability Toolkit.....	46
5.8	Weather Service .....	47
5.9	Smart Risk Assessment Platform.....	50
6	Conclusions .....	52
7	References .....	53
Annex I	CI/CD Sample .....	55
Annex II	Evacuation Coordinator message exchange with dependent components .....	58
Annex III	List of Kafka Topics .....	61
Annex IV	PIMM API Assessment .....	64
Annex V	VDR PDF Report Sample (draft) .....	67
Annex VI	Smart Bracelets Evacuation Support Messages .....	70

## List of Figures

Figure 1. PALAEMON-1 Kubernetes cluster – “dfb” namespace (list of active pods).....	12
Figure 2. PALAEMON-01 Kubernetes cluster – “brokers” namespace (list of active pods) ..	13
Figure 3. Platform deployment repository - Docker Compose tree .....	13
Figure 4. PALAEMON-02 Kubernetes cluster – All namespaces (list of active pods) .....	15
Figure 5. Source Code Issues sample (project: Platform Deployment).....	16
Figure 6. Source code branching management sample (project: Platform Deployment).....	16
Figure 7. CI/CD pipeline sample overview (Project: Voyage Report Generator) .....	17
Figure 8. CI/CD pipeline sample stages details (Project: Voyage Report Generator) .....	18
Figure 9. Evacuation Coordinator protocol: resource discovery and heartbeat.....	19
Figure 10. Evacuation Coordinator protocol: evacuation status change management.....	21
Figure 11. Evacuation Coordinator Monitor layout (Normal Status) .....	23
Figure 12. Evacuation Coordinator Monitor layout (Boarding to MEV Status).....	24
Figure 13. Kibana Canvas SQL query example.....	25
Figure 14. PALAEMON Canvas – Front page .....	25
Figure 15. PALAEMON Canvas - Conning Panel (emulation) .....	26
Figure 16. PALAEMON Canvas - Passenger list.....	26
Figure 17. PALAEMON Canvas - Passenger monitoring.....	27
Figure 18. PALAEMON Canvas - VDES outgoing signals.....	28
Figure 19. PIMM General View (Decision Support Center) .....	29
Figure 20. PIMM general view (SRAP Danger in ship) .....	30
Figure 21. PIMM general view (Evacuation Status) .....	30
Figure 22. PIMM Postman collection - list of services.....	31
Figure 23. Grafana Kafka Monitoring Sample .....	31
Figure 24. PALAEMON Voyage Report Generator sequence diagram.....	32
Figure 25. MinIO (part of DFB) repository in the cloud - PALAEMON buckets .....	34
Figure 26. MinIO "palaemon-reports" bucket zoom .....	35
Figure 27. Voyage report zip file password request.....	35
Figure 28. Voyage Data Report content .....	36
Figure 29. Voyage Data Report smart-cameras folder content.....	36
Figure 30. NiFi MQTT to Kafka flow sample (screenshot) .....	37
Figure 31. Smart Bracelets MQTT communication sample (/heartbeat-request).....	38
Figure 32. Smart Bracelets MQTT message sample .....	39
Figure 33. Smart Cameras main dashboard layout (PIMM's Video Streaming Center).....	41
Figure 34. Smart Safety System Main layout .....	42
Figure 35. VDES Transceiver Prototypes (ship & shore).....	43
Figure 36. Safety Management System - Report updated to cloud repo (console proof) .....	45
Figure 37. Safety Management System - ISM Dashboard visualization of report generated by the VRG .....	45
Figure 38. Weather service REST API request and response example.....	48
Figure 39. Weather Service Map user interface sample .....	49
Figure 40. Voyage Report Example - Ship & Voyage Particulars .....	67
Figure 41. Voyage Report Example - Passenger list .....	68
Figure 42. Voyage Report Example - Ship Evacuation Status Timeline .....	68
Figure 43. Voyage Report Example - Ship position & trajectory .....	68
Figure 44. Voyage Report Example - Smart Cameras alarms timeline.....	69
Figure 45. Voyage Report Example - Smart Bracelets alarms (i.e., fall detection) timeline..	69



Figure 46. Voyage Report Example - Ship Health Monitoring alarms timeline .....	69
Figure 47. Smart Bracelets Evacuation Support Messages.....	70

## List of Code Snippets

Code snippet 1. PIMM docker-compose file example .....	14
Code snippet 2. Kafka logging sample .....	22
Code snippet 3. Voyage report generator message exchange .....	34
Code snippet 4. Smart Bracelets Kafka sample (/heartbeat-response) .....	38
Code snippet 5. Smart Bracelets Kafka sample (/smart-bracelet-sensor-data).....	39
Code snippet 6. Smart Cameras Kafka sample (/smart-camera) .....	40
Code snippet 7. Smart Cameras Kafka sample (/smart-camera-alarm).....	40
Code snippet 8. Smart Safety System Kafka sample (/smart-safety-system) .....	42
Code snippet 9. VDES Gateway Kafka sample (/ais-position) .....	43
Code snippet 10. VDES Gateway Kafka sample (/weather-service).....	44
Code snippet 11. VDES Gateway Kafka sample (/mayday-message).....	44
Code snippet 12. Ship Health Monitoring Kafka sample (/shm-report) .....	46
Code snippet 13. Ship Health Monitoring Kafka sample (/shm-notification).....	46
Code snippet 14. Ship Stability Toolkit Kafka sample (/stability-toolkit).....	47
Code snippet 15. Weather-service Kafka message sample (/weather-service) .....	49
Code snippet 16. Smart Risk Assessment Platform Kafka message sample (/srap – Situation assessment).....	50
Code snippet 17. Smart Risk Assessment Platform Kafka message sample (/srap – Mustering).....	51
Code snippet 18. Smart Risk Assessment Platform Kafka message sample (/srap – Preabandonment) .....	51
Code snippet 19. Platform Deployment repository .gitlab-ci.yml file (CI/CD example) .....	55
Code snippet 20. Evacuation Coordinator - /resource-discovery-request and /resource-discovery-response (sample) .....	58
Code snippet 21. Evacuation Coordinator - /heartbeat-request and /heartbeat-response (sample).....	59
Code snippet 22. Evacuation Coordinator - /evacuation-coordinator and /evacuation-component-status (sample).....	60

## List of Tables

Table 1. Kafka Topics compilation table .....	61
Table 2. PIMM Get Token Request .....	64
Table 3. PIMM Get Evacuation status .....	64
Table 4. PIMM Get Voyage Status .....	64
Table 5. PIMM Get Fire Sensor data .....	64
Table 6. PIMM Get Weather Forecast Toolkit .....	65

**Abbreviations**

AIS	Automatic Identification System
CI/CD	Continuous Integration / Continuous Deployment
CSV	Comma Separated Values
DPA	Designated Person Ashore
DSS	Decision Support System
ETL	Extract, Transform and Load
GMDSS	Global Maritime Distress and Safety System
HTTP	HyperText Transfer Protocol
ICT	Information and Communication Technologies
ISM	International Safety Management
NOAA	National Oceanic and Atmospheric Administration
PaMEAS	PALAEMON Mustering and Evacuation Process Automation System
PIMM	PALAEMON Incident Management Module
RAO	Response Amplitude Operator
RPC	Remote Procedure Call
SDN	Software Defined Radio
SQL	Structured Query Language
SRAP	Smart Risk Assessment Platform
SSL	Secure Sockets Layer
URL	Uniform Resource Location
VCS	Version Control System
VDES	VHF (Very High Frequency) Data Exchange System
VRG	Voyage Report Generator



## 1 Introduction

The definition, implementation and testing of the PALAEMON Information and Communication Technologies (ICT) platform has been a tough work that have lasted more than 3 years, with more than 200 conference meetings held in between. This deliverable aims to close WP7's activities, reporting all kind of evidence that comes to demonstrate that the PALAEMON Communications Platform is completely functional. After this, we could consider that the system is ready to be delivered to WP8 (PALAEMON Application Field Trials, Evaluation and Outcomes), where we will deploy and test the infrastructure under real conditions. Namely, the final assessment will take place in ANEK's Elyros Ro-Pax Ferry [1].

During the initial phase of the project, alongside the definition of the internal components (sensors, user interfaces, databases, communication protocols, high-level services, etc.), we nailed down the approach to be followed on the software development and further integration process. In a nutshell, we harness the possibilities behind open-sources collaborative platforms, including tools to report potential issues and automatically re-deploy all components without any user intervention. This is facilitated by the use of containers as standalone executable packages that permit handling each component independently. Through this so-called micro-service-oriented philosophy, we carry out a hybrid deployment system of containers based on a combination of Kubernetes and Docker Compose.

We also have to take into account the behaviour of all software components according to the current phases during an incident. For that purpose, we refresh the tailored communications protocol held between the PALAEMON Evacuation Coordinator, a core element that centralizes and keeps track of the status of not only the incident level, but also all software components.

Running on a development ecosystem, based on two virtual machines running on a cloud-based server, we have emulated two independent instances, i.e., ship and shore, trying to mimic the real infrastructure the PALAEMON system was conceived for. This report summarizes the main results we have gathered after carrying out a thorough validation campaign. On the one hand, we present the most remarkable outcomes offered by services or tools that show a holistic interaction among the different components, either in terms of message exchanges (e.g., based on Apache Kafka) or using a graphical user interface, thus giving a more immediate and intuitive feedback. As a second part of the validation process, we also include individual checks, showing the information generated by most of the main components of the system.

We have structured this document as follows: Section 2 refreshes the proposed the software development and integration methodologies compared to those undertaken during the project implementation stage. Section 3 illustrates how the PALAEMON platform handles the evacuation process, where the PALAEMON Evacuation Coordinator centralizes the and keeps track of all the process. Section 4 displays the main outputs from the various test benches we have used to assess that components behave as expected. Section 5 presents individual proofs of operation that come to complement the results of the previous section. Section 6 concludes the document and closes all the activities carried out in WP7. Finally, we must remark the presence of up to six (6) annexes that complement the validation process described in the core sections of the deliverable.

## 2 Software development and integration methodology

Deliverable D7.4 (Software Development and Integration Methodology) [2] settled down the philosophy all partners inside the consortium shall follow to develop their own software resources. In a nutshell, these were the main principles we agreed at that time:

- **Requirement-based design.** In Deliverable D2.6 (PALAEMON Architecture v1) [3] we built a complete list of user-based (i.e., stakeholders') requirements. This compilation gave rise to the first and second versions of the PALAEMON Architecture.
- **Micro-service-oriented architecture.** Splitting the classical monolith into a number of independent software modules means that each partner could choose their programming framework (e.g., Python, C, Java, etc.). At the end, we only have to focus on the way these modules communicate with each other and how they save the information (e.g., databases, etc.), leaving all the rest to developers' choice.
- **Hybrid Kubernetes/Docker deployment.** This is a direct consequence of the use of micro-services instead of a monolithic solution. The deployment of the components become extremely simple and seamless. Moreover, the utilization of Kubernetes [4] over Docker [5] containers permits the orchestration of the most critical or complex components (e.g., databases, streaming brokers, etc.)
- **Inter-component communication.** One of the critical decisions we took is that, instead of specifying individual HTTP (HypeText Transfer Protocol) or RPC (Remote Procedure Call) interfaces for each component, we rely on<sup>1</sup> the main communications platform, Apache Kafka [6], the most widespread distributed event streaming platform.
- **Open-source nature.** One of the main perks of this type of collaborative projects is that partners can learn from each other. This opens the door to sharing the breakthrough achieved on the software development phase. In the context of PALAEMON, we have opted for GitLab [7] as the Version Control System (VCS) and Sonatype Nexus Manager [8] to manage all the binaries and artifacts created, i.e., Docker images, Python PIP packages, etc.
- **Continuous Integration / Continuous Deployment (CI/CD).** Automating tasks permit developers to save tens of hours during the development process. Now, at the same time they save a (differential) copy of their source code, there are frameworks that seamlessly build all the infrastructure and redeploy the new and updated components.
- **Testing.** The main goal of this deliverable is to demonstrate that the PALAEMON Communication Platform is completely operative and, thus, WP8 receives a functional framework. We report throughout the document the main checks we have carried out to demonstrate that everything is in place to handle a real evacuation scenario.
- **Documentation.** There is no good development unless a good (or event better) documentation supports the source code. When it comes to replicate the conditions (e.g., initialization, configuration, expected operation, etc.), it is deemed necessary to have dedicated a fair amount on time writing down all this information.

To complement the reading of the software development and integration methodology, the reader shall refer to D2.6 (PALAEMON Architecture v1) [3] and D2.7 (PALAEMON Architecture v2) [9]. In these deliverables, a complete picture of the PALAEMON Reference Architecture is given.

---

<sup>1</sup> We can find some RESTful interfaces, but these are for secondary purposes.

## 2.1 Docker & Kubernetes

As described in D7.5 (System Integration & Final PALAEMON Prototype v2) [10], the ICT infrastructure of the PALAEMON Communications platform (final release) is split into 6 parts:

1. **PALAEMON Field Devices:** this category corresponds to external hardware (e.g., sensors, cameras, transceivers...) that bring information to the central system.
2. **User equipment:** similar to the previous case, but in this case a person (passenger/crew) uses/wears the device (Augmented Reality glasses, smart bracelets and smartphones).
3. **PALAEMON-01 (Elyros' emulated infrastructure).** This is the cornerstone of the PALAEMON platform. Thought to run on a ship's premises (i.e., servers), key services run here. Communication (Kafka) brokers, databases and high-level services are executed in what we could assume as the system's mainframe.
4. **PALAEMON-02 (Shore emulated infrastructure).** A lightweight version of the PALAEMON ship (main) system is replicated in an alike ashore environment, where different stakeholders (e.g., Port Authorities, Designated Person Ashore – DPA, etc.) can take part of the story.
5. **Internet/cloud services:** Some core components need to rely on external services to operate correctly. Hence, an Internet connection is required to have access to these.
6. **PALAEMON Academy:** Independent infrastructure that attends to passengers and training programme. Nothing to do with the actual evacuation process.

Concerning the deployment mode, we only have to pay attention here to PALAEMON-01 and PALAEMON-02 components, as the other groups do not use containers or orchestrators, as they natively run their software instead.

Despite the initial idea of running all containers on the same Kubernetes cluster, we opted for a split between using Kubernetes and Docker (via Docker Compose [11]). The rationale behind that is twofold: 1- The use of the orchestrator presents a steep learning curve to non-skilled partners. 2- Most of the smart services will not present high computational demands that may lead to have a stricter control, for example in the form of a high demand peak or an exclusive database. For that reason, only the critical services are on Kubernetes, as they do require a thorough management, for the sake of scalability, observability, persistence of the information, etc. Moreover, we have to say that the deployment phase is automatically carried out by GitLab's CI/CD framework, as we will detain Section 2.3.

As a matter of fact, we have used an external tool call Kubernetes Lens [12] (built on open-source and free for personal use) to visualize the content of the clusters. As it goes beyond the testing campaign to present all the elements that are orchestrated by Kubernetes, we only display the active pods<sup>2</sup> that are up-and-running in the cluster. For PALAEMON-01, we have created a single-node cluster. The load of the whole system does not demand much computational cost and a single virtual machine is able to withstand everything. Within the same cluster, we have configured a handful of so-called namespaces, which can be seen as logically separated domains, almost virtual sub-clusters. Some of them are devoted to internal cluster management and will be left aside this analysis. For our purposes, there are two main namespaces: “dfb” and “brokers”.

<sup>2</sup> According to Kubernetes' official documentation [41], a pod “is the smallest deployable units of computing than can be created and/or managed in Kubernetes).

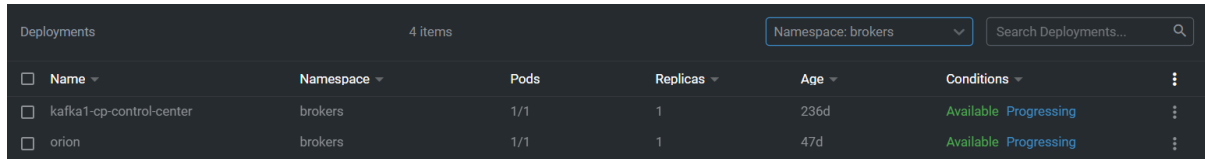
Figure 1 shows all the deployments is inside “dfb” namespace. We can observe the presence of a handful of background elements that are out of the scope of this analysis. Their mission is to offer management actions or observability features to the system orchestrator (i.e., Kubernetes). Below we summarize the components that have been deployed in the context of the PALAEMON platform:

- *grafana-deployment* – Component that help monitor Kafka’s flows and behaviour (we will cover this in Section 4.5)
- *es-connector-deployment* – Kafka to Elasticsearch connector
- *es-proxy-deployment* – DFB Endpoint API that serves the information from the database (i.e., Elasticsearch)
- *kapow* – Kapow! [13] is a service that turns a shell development into an HTTP API. It will work as part of the Voyage Report Generator component
- *kibana-deployment* – Data Visualization layer that works on top of Elasticsearch (Section 0 is based on this framework).
- *evac-coord* – PALAEMON Evacuation Coordinator
- *kafka-1 deployment* – Kafka Broker
- *mosquitto* – MQTT Broker that allows the communication with some devices (i.e., smart bracelets and VDES – VHF Data Exchange System – transceivers).
- *nginx-deployment* – Reverse Proxy that maps between internal IP addresses and final URLs (Uniform Resource Location).
- *nifi* – Apache NiFi [14] is a frameworks that takes the (raw) data, transforms it on-the-fly and streams to the next level (e.g., database, services, etc.)
- *nifi-registry* – NiFi’s Companion component[15],
- *vdr* – Voyage Report Generator component.
- *zookeeper-deployment* – Apache Zookeeper [16] is a Kafka companion component, responsible for its configuration and internal management

Deployments						
17 items			Namespace: dfb		Search Deployments...	
<input type="checkbox"/>	Name	Namespace	Pods	Replicas	Age	Conditions
<input type="checkbox"/>	busybox-deployment	dfb	1/1	1	2y4d	Available Progressing
<input type="checkbox"/>	dfb-admin-api-deployment	dfb	1/1	1	2y2d	Available Progressing
<input type="checkbox"/>	es-connector-deployment	dfb	1/1	1	49d	Available Progressing
<input type="checkbox"/>	es-proxy-deployment	dfb	1/1	1	679d	Available Progressing
<input type="checkbox"/>	evac-coord	dfb	1/1	1	8h	Available Progressing
<input type="checkbox"/>	grafana-deployment	dfb	1/1	1	680d	Available Progressing
<input type="checkbox"/>	jmx-exporter-1-deployment	dfb	1/1	1	680d	Available Progressing
<input type="checkbox"/>	kafka-1-deployment	dfb	1/1	1	629d	Available Progressing
<input type="checkbox"/>	kapow	dfb	1/1	1	22d	Available Progressing
<input type="checkbox"/>	kibana-deployment	dfb	1/1	1	326d	Available Progressing
<input type="checkbox"/>	mosquitto	dfb	1/1	1	277d	Available Progressing
<input type="checkbox"/>	nginx-deployment	dfb	1/1	1	2y1d	Available Progressing
<input type="checkbox"/>	nifi	dfb	1/1	1	242d	Available Progressing
<input type="checkbox"/>	nifi-registry	dfb	1/1	1	238d	Available Progressing
<input type="checkbox"/>	prometheus-deployment	dfb	1/1	1	680d	Available Progressing
<input type="checkbox"/>	vdr	dfb	1/1	1	22d	Available Progressing
<input type="checkbox"/>	zookeeper-deployment	dfb	1/1	1	2y2d	Available Progressing

Figure 1. PALAEMON-1 Kubernetes cluster – “dfb” namespace (list of active pods)

The second namespace, “brokers”, as shown in Figure 2, only manages the two communication brokers: Apache Kafka and Orion-LD Context Broker, with a single pod per deployment.



Name	Namespace	Pods	Replicas	Age	Conditions
kafka1-cp-control-center	brokers	1/1	1	236d	Available Progressing
orion	brokers	1/1	1	47d	Available Progressing

Figure 2. PALAEMON-01 Kubernetes cluster – “brokers” namespace (list of active pods)

Aside the picture, we must mention Keycloak [17], the main Identity Manager that handles the authentication and authorization of external accesses to the main components).

Still on PALAEMON-01, as we mentioned at the beginning of the section, there are other software modules that are deployed via Docker Compose. To illustrate this, Figure 3 shows the (tree-like) structure of all the files<sup>3</sup> defined in the Platform Deployment repository on GitLab (private repository only for consortium members). In the “*docker-compose*” folder we manage each of the components independently, each one with its respective environment variables file(s)<sup>4</sup>, aside their *docker-compose-\*.yaml* file, of course. In addition, we differentiate between those elements that are to be deployment on the ship infrastructure (PALAEMON-01) and that of shore (PALAEMON-02).

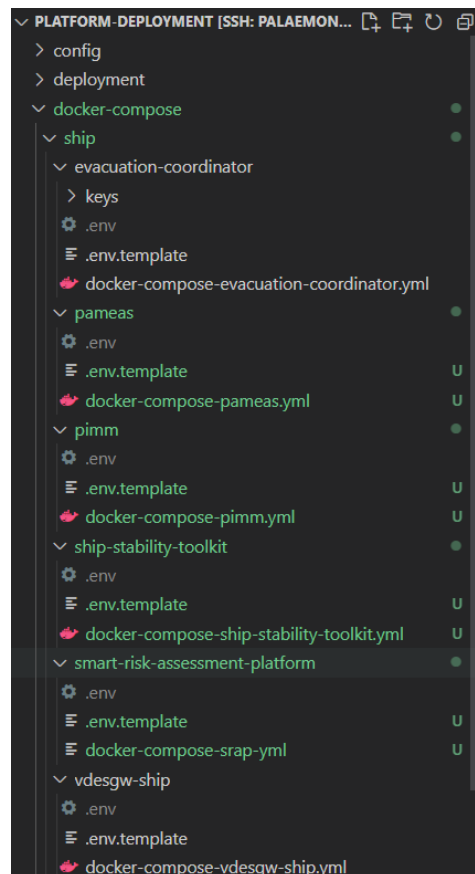


Figure 3. Platform deployment repository - Docker Compose tree

<sup>3</sup> The screenshot was taken from the popular Visual Studio Code Source code editor, from Microsoft

<sup>4</sup> In order not to upload to the source code repository sensitive environment variables, we include an “.env.template” that is pushed to GitLab; on the other hand, “.env” files are only kept locally.

Namely, we use Docker Compose to deploy the following components on PALAEMON's mainframe:

- *Ship Stability Toolkit* [18] – ship motion forecasting service
- *VDES Gateway* [19] – middleware between VDES transceiver (onboard) and the PALAEMON system. As a matter of fact, there is a twin component deployed ashore (PALAEMON-02).
- *Smart Risk Assessment Platform (SRAP)* [20] – smart evacuation component, quantifying the risk during mustering and abandonment phases.
- *PALAEMON Incident Management Module (PIMM)* [21] – visual information hub during the evacuation.

To illustrate in a nutshell how a *docker-compose\*.yml*, Code snippet 19 shows the main components deployed alongside the PIMM. In this particular example, we can see a handful of companion elements (e.g., Django to provide a backend server, NGINX as an internal reverse proxy), the images taken as reference, the volumes used and the environment variables. It is important to see that PIMM spans the deployment of *Decision Support System (DSS)* and *Weather Forecast Toolkit (WFT)*.

```
version: '3'
services:
  django:
    container_name: pimm-back
    build:
      context: .
      dockerfile: ./docker/pimm/Dockerfile
    command: /usr/src/app/start_pimm_back.sh
    volumes:
      - static_volume:/usr/src/app/static/
      - ../certs-kt:/usr/src/app/certs/
    environment:
      DJANGO_SETTINGS_MODULE: PIMM-Back.settings.production
      EVAC_COORD_ADDRESS: ${PALAEMON-01_EVAC_COORDINATOR_IP}
  nginx:
    build:
      context: .
      dockerfile: ./docker/nginx/Dockerfile
    ports:
      - 127.0.0.1:8000:80
    volumes:
      - static_volume:/home/app/web/static/
    restart: always
  front:
    image: pimm-front
    ports:
      - 127.0.0.1:8080:80
    restart: always
    depends_on:
      - nginx
  wft:
    image: wft
    environment:
      PIMM_ADDRESS: http://django:8000
    volumes:
      - ../certs-kt:/usr/src/app/certs/

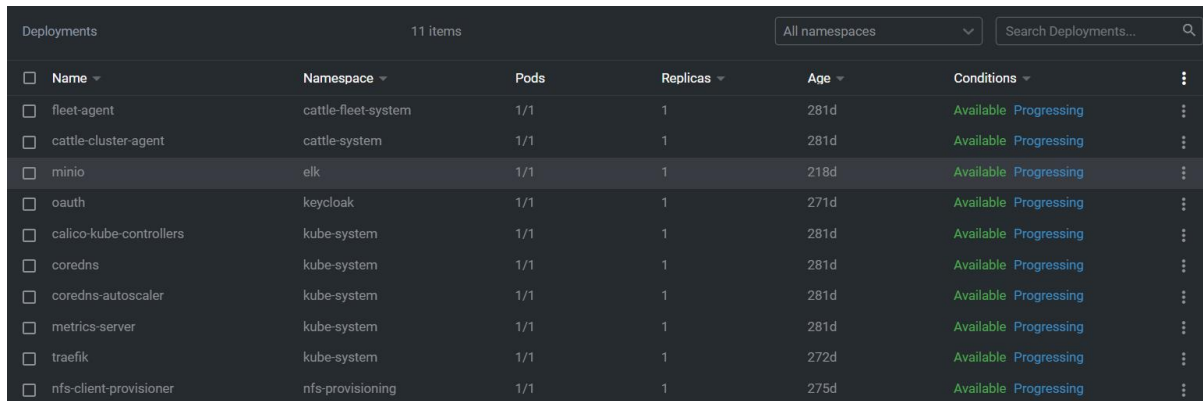
volumes:
  static_volume:
```

Code snippet 1. PIMM docker-compose file example



There is a special case that was not deployed on Kubernetes or Docker (Compose). It is the case of DANAOS' Safety Management System, a proprietary software only available for Windows systems (the reader shall know that PALAEMON-01 and PALAEMON-02 are virtual machines based on Ubuntu OS).

On the other hand, PALAEMON-02 is a lightweight version of the main PALAEMON platform and only needs a small subset of all the available software elements. From all the list of components that appear in Figure 4, we can only refer to two as part of the system: MinIO [22] an open-source object storage to keep all Voyage Reports and Traefik [23], an application proxy that embraces authentication and networking (i.e., reverse proxy) tasks.



Name	Namespace	Pods	Replicas	Age	Conditions
fleet-agent	cattle-fleet-system	1/1	1	281d	Available Progressing
cattle-cluster-agent	cattle-system	1/1	1	281d	Available Progressing
minio	elk	1/1	1	218d	Available Progressing
oauth	keycloak	1/1	1	271d	Available Progressing
calico-kube-controllers	kube-system	1/1	1	281d	Available Progressing
coredns	kube-system	1/1	1	281d	Available Progressing
coredns-autoscaler	kube-system	1/1	1	281d	Available Progressing
metrics-server	kube-system	1/1	1	281d	Available Progressing
traefik	kube-system	1/1	1	272d	Available Progressing
nfs-client-provisioner	nfs-provisioning	1/1	1	275d	Available Progressing

Figure 4. PALAEMON-02 Kubernetes cluster – All namespaces (list of active pods)

*At the time of closing this report, all PaMEAS (PALAEMON Mustering and Evacuation Process Automation System) is running in an independent and standalone environment. It is expected that it will be installed in the main infrastructure (i.e., Elyros) for the real scenario evaluation, that is, WP8's activities.*

## 2.2 GitLab Issues and Branches Management

Version Control Systems (VCSs) have deservedly become of the main game changers on software development in the last decade. Their presence is so incontestable that one cannot imagine a professional environment without using this kind of platforms. Their success has been so humongous that, aside keeping track of the source code, documentation, licenses, etc., the main VCSs' providers (e.g., GitHub, GitLab) have incorporated a number of extra features that come to enrich the experience. Amongst all of them, we mainly take advantage of two: issues management and CI/CD.

On issue tracking and management, we cannot rely only on a collaborative environment share ideas, but also a way to assign changes, new features and other development tasks among the main developers. For that purpose, we have defined a series of labels, based on three main categories.

- **Priority:** Critical, High, Medium, Low, Optional
- **Status:** Doing, To Do, Accepted, Blocked, On Hold, Review Needed, Abandoned
- **Type:** Bug, Documentation, Enhancement, Maintenance, New feature.

With all this, any modification in the repositories must be identified as an issue, which should have a responsible. Optionally, we can include deadlines, time tracking, etc. Figure 5 shows an example of the issue tracking (board layout) on the Platform Deployment project taken in the middle of the development phase.

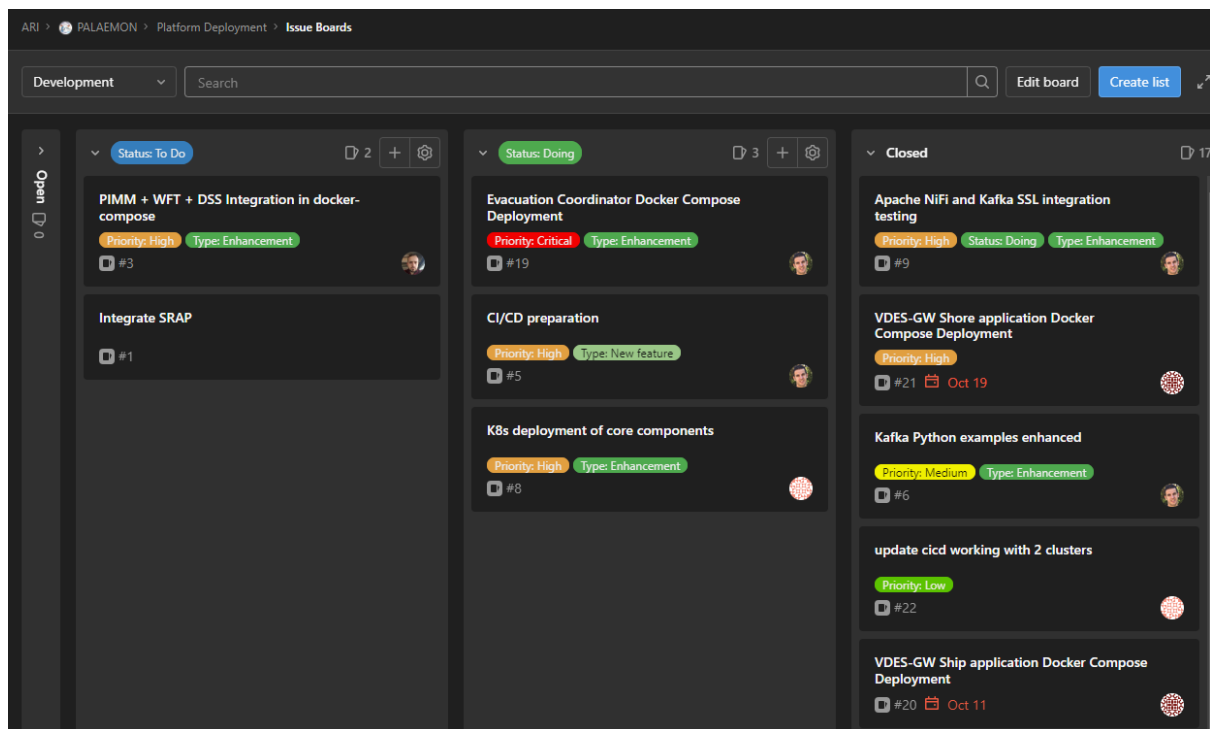


Figure 5. Source Code Issues sample (project: Platform Deployment)

For us, repository branching is extremely linked to the issue management described above. In fact, every issue should have its dedicated branch, thus the development should be parallelized and would not affect others' contributions. When an issue in particular is solved, its responsible generates a "Merge Request" and the corresponding reviewer, if the new content is valid, accepts the request and merges the code with the "master" branch (the main one, also known as "production"). As an illustrative example of how this branching works, Figure 6 shows how a handful of parallel works could be done simultaneously, converging always as part of the master branch at the end of the process.

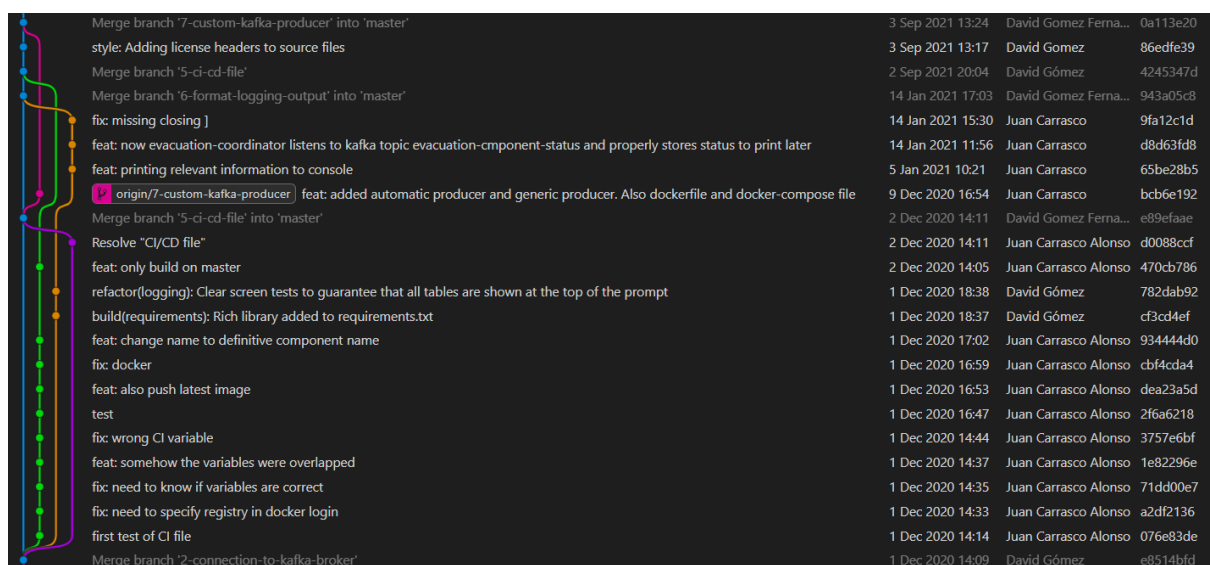


Figure 6. Source code branching management sample (project: Platform Deployment)



## 2.3 GitLab Continuous Integration / Continuous Deployment (CI/CD)

Current trends in software development target to maximize, to the extent possible, the number of automatized tasks during the process. In PALAEMON we believe in this philosophy, and we have followed its principles by harnessing the built-in CI/CD engine provided by GitLab. Explained in layman's terms, every time a new version of the code is pushed onto GitLab's remote repository, a CI/CD pipeline is activated. Users can configure how these actions are triggered (for instance, we only "unleash" the process when the changes are made in the "master" branch, and only in the modules that have undergone modifications, keeping intact those ones that have not been touched). Figure 7 presents a screenshot of GitLab summarizing the status of the last 6 CI/CD actions of the Voyage Report Generator at a particular time during the development process.

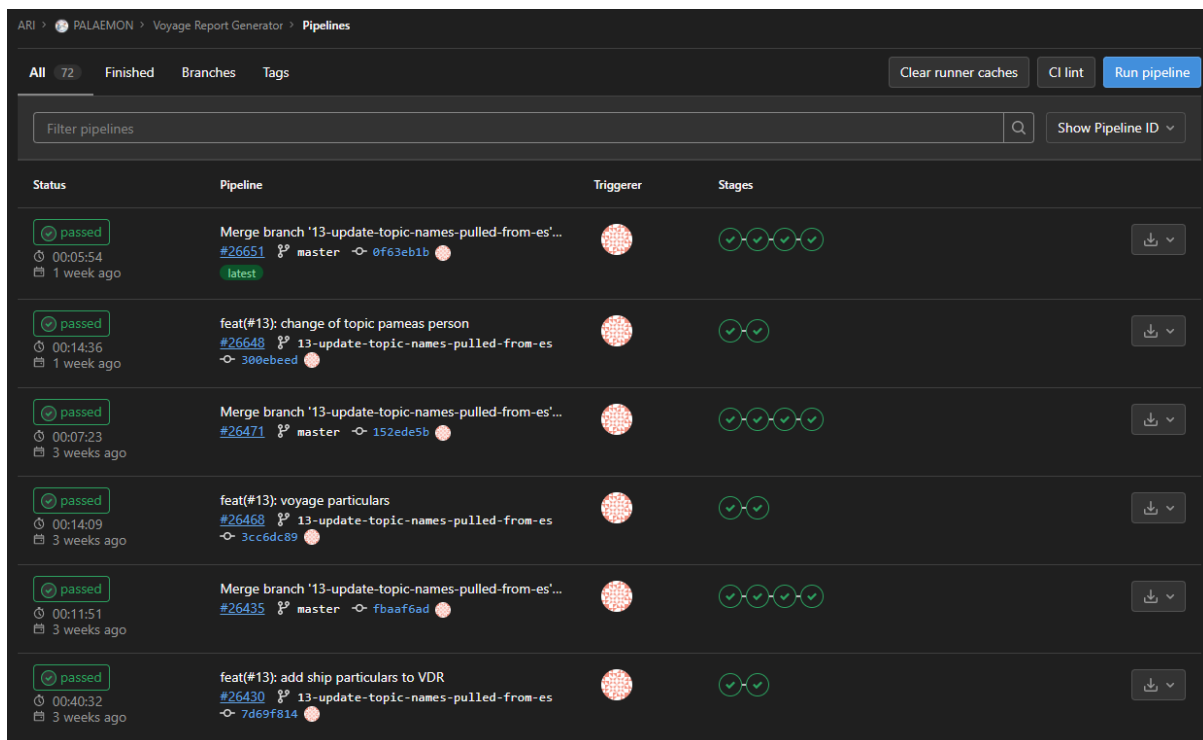


Figure 7. CI/CD pipeline sample overview (Project: Voyage Report Generator)

The figure above gives us relevant information about the pipeline execution, summarized in the following elements:

- **Status:** Either "Passed" or "Error", the time elapsed to execute the pipeline and when it happened.
- **Pipeline:** Action that trigger the execution of the pipeline. Namely, the Git "push" event, including the commit message and the affected branch ("master" and "13-update-topic-names-pulled-from-es").
- **Triggerer:** Developer who initiated the pipeline.
- **Stages:** The CI/CD pipeline can be split into a handful of independent stages. To illustrate this with a more detailed example, Figure 8 "zooms" into one of these pipelines (i.e., the most recent) and shows the different jobs (corresponding with the stages) carried out.

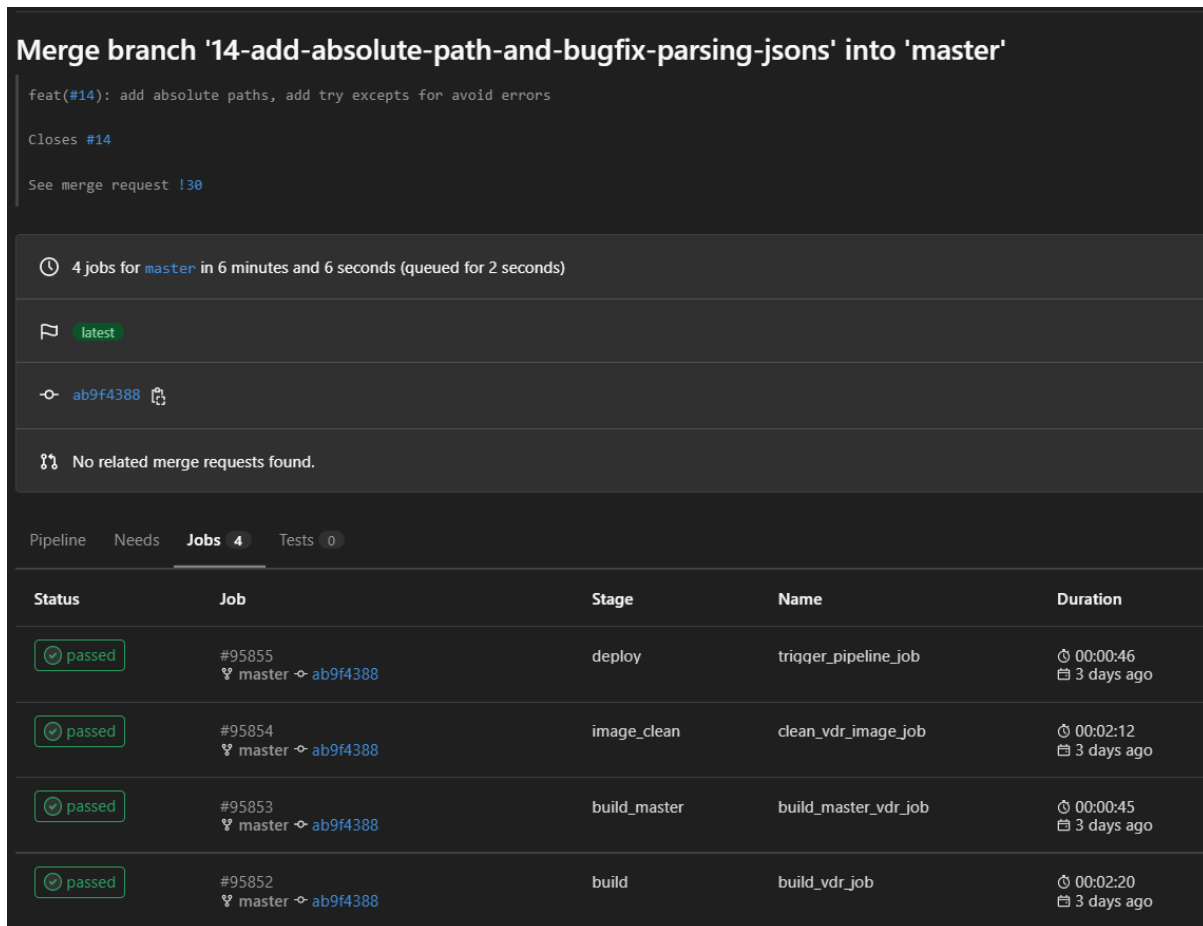


Figure 8. CI/CD pipeline sample stages details (Project: Voyage Report Generator)

There are countless configurations to address when implementing a CI/CD pipeline. In the scope of PALAEMON, we did not specify any mandatory stages or best practices when it comes to specifying it. Technically speaking, GitLab automatically takes the content of a file named **.gitlab-ci.yml** and runs it every time a new version of the code is pushed onto GitLab. In this particular example, we identify four stages:

1. **Build:** Automatic creation of Docker images corresponding to the component. In this stage we also upload the image to Nexus, the project's artifact repository.
2. **Build master:** Alike the previous one but only activated when the changes come from the "master branch".
3. **Image clean:** Removal of temporary images that will not be used anymore
4. **Deploy:** Download the image(s) and automatic deployment on destination (e.g., PALAEMON-01 VM).

As a matter of fact, the full CI/CD file (**.gitlab-ci.yml**) can be found in Annex I. In this example, the component is split in two Docker images, one for the native Python software and another one for the complementary solution based on Kapow! [13]. Moreover, we can also identify some of the development best practices we have followed in the project:

- Rely on environment variables to yield a dynamic and automatic CI/CD ecosystem.
- Do not include any sensitive information (URLs, passwords, certificates, etc.).
- Granularity: micro-services and stages are clearly identified and separated.

### 3 Evacuation Process Management

This section analyses what happens, from a platform or ICT perspective, during an evacuation. The cornerstone of the whole process is the PALAEMON Evacuation Coordinator, central communications element that keeps track of the status of the process, not only the evacuation per se, but the way the different software elements react and adapt their behaviour, according to the current phase during an evacuation. It is worth highlighting that the **Master of the ship is the only person who has the rights** to modify the status during an evacuation. To do so, he/she will interact with the system by using the PIMM's interface (see Section 4.4).

As illustrated in Figure 9, when the voyage starts, the PALAEMON Platform is initiated. The first step corresponds to the so-called "Resource Discovery" process. We assume that the DFB database (i.e., Elasticsearch) only contains non-voyage specific information, such as ship particulars, blueprints, passengers and crew lists<sup>5</sup>, etc. at this point. Technically speaking, the Master "starts" the system by clicking the "Start Voyage" button on the PIMM<sup>6</sup>. Internally, the PIMM sends a notification (i.e., REST interface) to the Evacuation Coordinator, which proceed to generate a `/resource-discovery-request` that broadcasts to all the listeners (via Kafka).

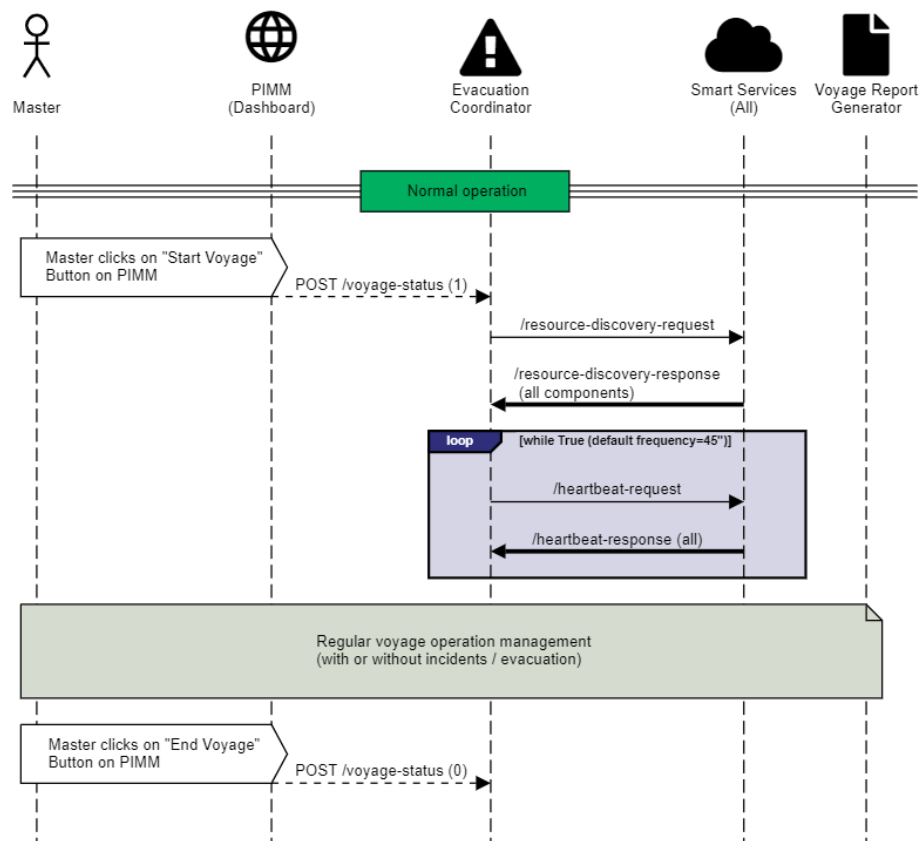


Figure 9. Evacuation Coordinator protocol: resource discovery and heartbeat

<sup>5</sup> Passenger and crew lists are obtained prior to embark by means of PaMEAS Registration Service. The information is kept anonymized (GDPR restrictions) in DFB database.

<sup>6</sup> This button emulated the actual departure of the ship. A further version may change this for an automatic trigger based on e.g., the ship position, voyage particulars, etc.

On the other side, components subscribe and receive notification to the topic or topics they want to receive notifications for. Upon this request, smart services (i.e., smart cameras, smart bracelets, SRAP, PIMM, PaMEAS, SSS, SST, VDES Gateway onboard, SHM) get the notification. As a reply, they send back a message (i.e., **/resource-discovery-response**), where they share with the PALAEMON Evacuation Coordinator their unique identifier, a timestamp (to keep track of) and the current operation mode of the component (this is documented in D7.5 – Uniform Data Exchange Modules – Interoperability Layer v2 ) [24]. Once all components have responded, the PALAEMON Evacuation Coordinator is completely aware of all the active modules that are dependent of the actual evacuation status.

Beside this discovery phase, there is a periodic “health check” of all components. To deal with this, the PALAEMON Evacuation Coordinator generates, every 45 seconds<sup>7</sup> broadcasts a **/heartbeat-request** notification, where the payload contains the ID of the originator of the message (the PALAEMON Evacuation Coordinator), the timestamp and the current status of the evacuation. In turn, components send back a **/heartbeat-response**, confirming their availability and their current mode of operation. As a matter of fact, one of the features that went beyond the scope of the project breakthrough is the reaction upon loss of contact with components, i.e., what to do if one or more devices lose contact with the main system. We studied the possibility of including a set of actions in such a case, but we mainly identified this as a future operation to be tackled in forthcoming release. The reader must know that this “liveness probes” are sent during all the voyage, regardless an incident has occurred or not.

However, in this project we must deal with evacuation scenarios, hence things will get worse and the whole PALAEMON system must be ready for that and present a holistic management process. Figure 10 presents the sequence diagram between the PALAEMON Evacuation Coordinator and the rest of the components. For the sake of simplicity, the schematic disregards all additional messages that may be generated as a consequence of this exchange.

When an incident happens, the Master gathers all the awareness of the situation, e.g., via radio communications with the crew, Integrated Bridge System (IBS), information displayed on PIMM<sup>8</sup> that comes from the PALAEMON devices, etc. With all this information on the table, he/she may take the decision of increase the evacuation level to “Situation Assessment”. Thus, beside the legacy voice alert to officers and crew via “Direct Public Alert System”, the Master has to switch from “Normal Operation” to “Situation Assessment” on PIMM’s interface (Figure 19 shows main PIMM’s dashboard in page 29). When this happens, the process is analogous to that of we described for the resource discovery. In this case, the notification is sent using the **/evacuation-coordinator** topic, announcing in this case the new status of the evacuation (alongside the ID of the originator and the current timestamp). In the reverse sense, all the dependent components send back an acknowledgement (i.e., **/evacuation-component-status**), confirming the correct reception of the new status and, at the same time, sharing their current operation mode, which may have changed with this modification. Likewise for this step, the process is replicated in the next phases of the evacuation<sup>9</sup>. Of course, it goes without saying that these ICT and manual notifications go in parallel with legacy evacuation management procedures (e.g., General Public Alarm System, etc.).

<sup>7</sup> The sending rate can be configured.

<sup>9</sup> It is worth stating that the evacuation process is reversible and the evacuation might be aborted at any time.

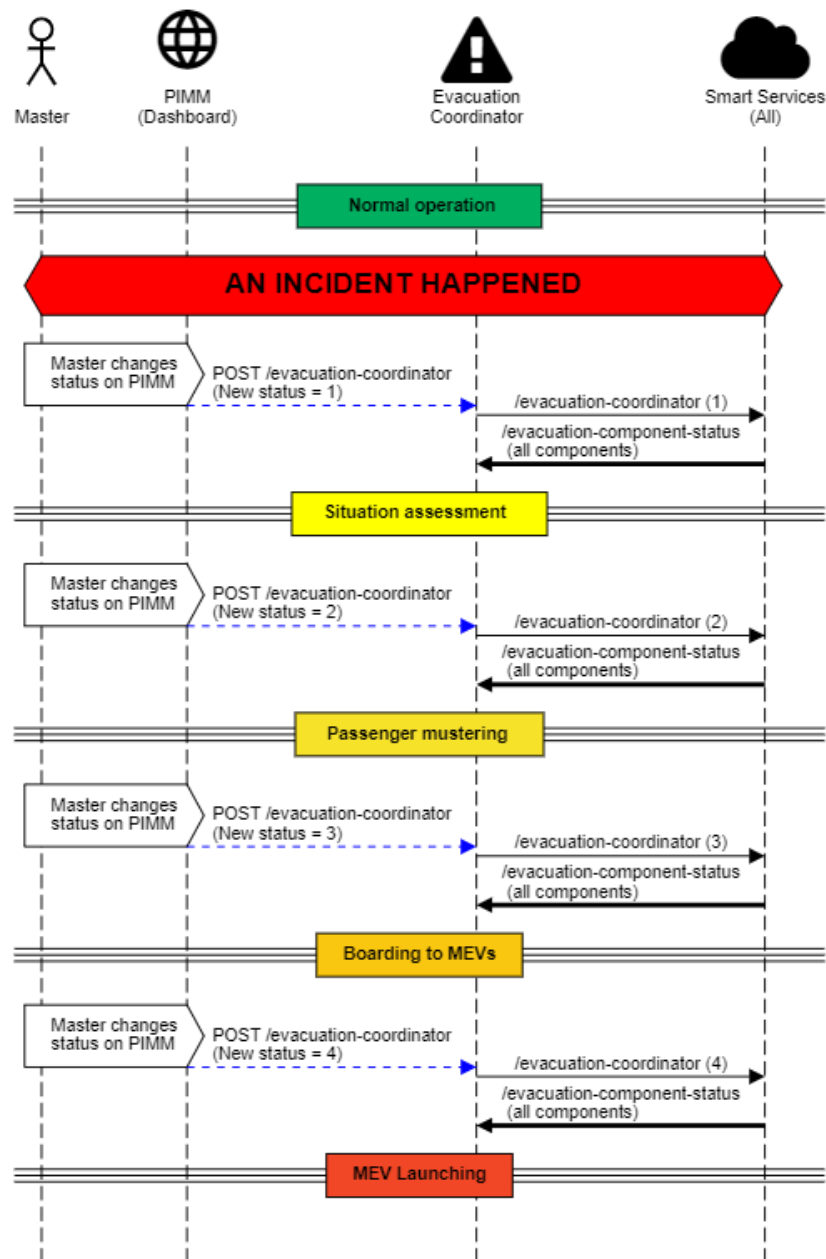


Figure 10. Evacuation Coordinator protocol: evacuation status change management

To complement the content of this section, Annex II presents the message primitive exchange between the PALAEMON Evacuation Coordinator and the underlying dependant components.

## 4 Overall system testing results

This section gathers the main integration evidence that demonstrate that the elements of the PALAEMON Communications platform exchange information accordingly. In particular, we reflect here the output of some transversal platforms or tools that come to showcase that

### 4.1 DFB (Apache Kafka & Elasticsearch) as system validator

The cornerstone of the PALAEMON Communications platform is based on Apache Kafka. We opted for this streaming platform, instead of a more traditional approach based on a RESTful interface for two main reasons:

1. The utilization of mainstream RESTful interfaces would have required as individual specification per interface. Moreover, we should know beforehand a valid endpoint (i.e., IP address + TCP port) for each one. Instead, with Kafka, the different modules only need to know to which topic they should read (i.e., subscribe) or write (i.e., publish).
2. The use of dedicated SSL (Secure Sockets Layer) certificates associated to an Access Control List (ACL) offers a robust authentication and authorization solution. This applies not only to external attacks (nobody can read or write on the Kafka Broker without a valid certificate), but also internally, as only allowed partners have rights to ready or write on some message topics.

We observe in Code snippet 2 an example of communication between two components, where we print out the topic name, the offset and partition<sup>10</sup> and the JSON object sent. In this particular case, the PALAEMON Evacuation Coordinator broadcast a **/resource-discovery-request**, as explained in the previous section; right after, the Smart Safety System replies back a **/resource-discovery-response**.

```
=====
Received message (resource-discovery-request:0:119)
{'originator': 'Evacuation Coordinator', 'timestamp': '2022-11-
23T11:32:37.640964Z', 'evacuation-status': 0}
=====
Received message (resource-discovery-response:0:553)
{'component_id': 'SSS', 'operation_mode': 0, 'timestamp': '2022-11-
23T11:32:37.914780+00:00'}
```

*Code snippet 2. Kafka logging sample (/resource-discovery-request and /resource-discovery-response notifications)*

It goes without discussion that Kafka messaging plays a critical role in the PALAEMON information schema. In this section we only include this example; Section 5 includes more snippets to showcase the correct operation of the components. Furthermore, Annex III compiles the list of all Kafka topics utilized in the context of an evacuation.

<sup>10</sup> Offset and partition are internal attributes that Kafka uses to handle its internal queues. This goes out of the scope of this report.



## 4.2 PALAEMON Evacuation Coordinator as system validator

In Section 3 we summarized the main protocol defined to communicate the current Ship's evacuation status and the components' operation mode (in the reverse sense of communication). Designed as a companion service to be displayed as part of the IBS, we have forked the baseline of the PALAEMON Evacuation Coordinator and have added a Command-Line Interface to have all the information visible at a simple glance. This lightweight component basically subscribes to all the topics that have to do with the evacuation management and displays the data in a visual way.

Figure 11 shows the main layout, split into two main parts: 1- the top part presents the current evacuation status ("Normal status" in this example). 2- The main part of the visualization compiles the list of devices and components that "listen" to the PALAEMON Evacuation Coordinator. This is represented as a table where each of the "dependent" components (first column) show: the unique identifier of the module (there might be more than one device per component, for instance the case of smart cameras and bracelets), the number of active components; the operation mode and, finally, the time the component replied to one of the coordinator messages (i.e., timestamp).

PALAEMON Evacuation Coordinator					Wed Nov 30 16:45:30 2022
Evacuation status					
<u>Normal status</u>	Situation Assessment	Passenger mustering	Boarding to MEVs	MEV Launching	
Component status					
Component	IDs	#Components	Operation mode	Last message	
Smart Risk Assessment Platform	srp	1	0	2022-11-30 16:44:46.915491	
VDES App	VDES App	1	0	2022-11-30T16:45:09.765744123Z	
Smart Bracelets	SB0001 SB0002	2	0	2022-11-30 16:44:46.915496	
Smart Cameras	camera-01 camera-02 camera-03 camera-04	4	0	2022-11-30T17:45:09.783687	
Voyage Report Generator	VDR	1	0	2022-11-30T16:45:09.767747	
PaMEAS Location	PaMEAS-Location	1	1	2022-11-30 16:44:47.345	
Ship Health Monitoring	SHM	1	0	2022-11-30 16:44:46.915498	
Smart Safety System	SSS	1	0	2022-11-30 16:44:46.915499	
Ship Stability Toolkit	SST	1	0	2022-11-30 16:44:46.915500	
PIMM + DSS + WFT	PIMM+DSS+WFT	1	1	2022-11-30T16:45:09.291335	

Figure 11. Evacuation Coordinator Monitor layout (Normal Status)

Figure 12 complements the previous view with another example; this case corresponding to a more advanced phase in the evacuation chain, i.e., "Passenger Mustering". At this stage, the General Public alarm has sounded and passenger, with the support of the crew and PALAEMON Smart Evacuation Devices (through their Smart Bracelets and PaMEAS messaging services) proceed to reach their safest/closest mustering point. In terms of operation mode, we can appreciate as some of the services have modified their behaviour, adapting to the severity of the situation. For instance, Smart Cameras start recording the videos they process and transmit the multimedia file to the Voyage Report generator. In

addition, they notify the detection of hazardous events (e.g., stampede, blocked corridor). Aside this example, Deliverable D7.2 (Uniform Data Exchange Modules – Interoperability Layer v2) [25] contains the mapping tables between components' operation modes and their actual description.

PALAEMON Evacuation Coordinator					Wed Nov 30 16:50:29 2022
Evacuation status					
Normal status	Situation Assessment	Passenger mustering	Boarding to MEVs	MEV Launching	
Component status					
Component	IDs	#Components	Operation mode	Last message	
Smart Risk Assessment Platform	srp	1	2	2022-11-30 16:50:18.170082	
VDES App	VDES App	1	1	2022-11-30T16:50:23.204239348Z	
Smart Bracelets	SB0001 SB0002	2	2	2022-11-30 16:50:18.170088	
Smart Cameras	camera-01 camera-02 camera-03 camera-04	4	2	2022-11-30T17:50:23.217663	
Voyage Report Generator	VDR	1	1	2022-11-30T16:50:23.209279	
PaMEAS Location	-	-	-	-	
Ship Health Monitoring	SHM	1	1	2022-11-30 16:50:18.170090	
Smart Safety System	SSS	1	1	2022-11-30 16:50:18.170091	
Ship Stability Toolkit	SST	1	1	2022-11-30 16:50:18.170092	
PIMM + DSS + WFT	-	-	-	-	

Figure 12. Evacuation Coordinator Monitor layout (Boarding to MEV Status)

### 4.3 Kibana (Canvas) as system validator

Elasticsearch [26] was chosen as the main database to persist all the information coming from the PALAEMON Communications Platform, that is, from data sources and smart evacuation services. Elasticsearch supports a handful of frameworks that bring new features to the ecosystem. One of them is Kibana [27], an open user interface (i.e., web-based) that adds a visualization layer on top of the data stored in Elasticsearch. It is worth highlighting that this framework or tool does not belong to the evacuation process, but we use it to check that the information is correctly saved in the repository. Besides, we use one of its most attractive features, Canvas [28], an add-on that produces eye-catching infographics with little effort, extracting the information out of Elasticsearch. For the sake of testing, we include in this section a handful of visual representations that come to showcase that the information generated in the system has successfully reached the database.

An interesting feature of Canvas is that it includes a rather complete built-in Structured Query Language (SQL) query syntax that works over a document-oriented database. This is not usual, as these domain-specific languages are tailored to work with relational databases (e.g., MySQL). Nonetheless, with some basic queries we can extract and display valuable voyage information. For instance, Figure 13 shows, on the left hand, an SQL query defined on Kibana dashboard<sup>11</sup>. In this particular query, we are searching the latest value, in terms of timestamp, stored as part of the “evacuation-coordinator\*” index<sup>12</sup>, where the “\*” illustrates that the use of wildcards or regular expressions are permitted. At the right, a tabular representation

<sup>11</sup> Canvas is a fully-fledged feature completely embedded as part of Kibana.

<sup>12</sup> An “index” can be seen as independent logical namespaces that separate different types of data. As analogy, this could be a table or database in a relational database.



of the response. As a matter of fact, this value will be displayed as part of the Canvas visualizations we include below.

**Selected element** ↑ ^ ∨ ↓

Display Data

**Elasticsearch SQL** >

Query [Learn ES SQL query syntax](#)

```
SELECT * FROM "evacuation-coordinator*" ORDER BY "timestamp" desc LIMIT 1
```

[Preview data](#) [Save](#)

**Datasource preview** ×

The following data will be available to the selected element upon clicking **Save** in the sidebar.

evacuation-status #	originator t	timestamp
0	Evacuation Coordinator	2022-12-05T19:31:09+01:00

< 1 >

a) Canvas SQL query

b) Elasticsearch response preview

Figure 13. Kibana Canvas SQL query example

Explained in layman's terms, Canvas can be seen as a dynamic "PowerPoint"-like presentation, where the displayed data is directly taken from Elasticsearch. Figure 14 could be used as the front page of the infographics, where we take some static information, such as some ship particulars (e.g., Vessel name, Flag, IMO number, etc.). We combine this with some voyage-specific information: departure and arrival ports, departure time, etc. Additionally, we include some aggregated data, as the number of passengers and crew member registered in the ship. This information can be calculated by summing up all the users registered through the PaMEAS Registration process. The reader might recall that all personal information is anonymized and stored in Elasticsearch; here we only calculate the total figures.



PALAEMON

## Voyage particulars



Vessel Name	Elyros
Flag	Greece
IMO Number	9178599
Departure Port	Piraeus
Departure Time	2022/11/23 14:30:00 EET
Arrival Port	Chania
Passengers	174
Crew	64

Figure 14. PALAEMON Canvas – Front page

The second “slide” mimics a conning display that prints basic information captured by the PALAEMON system, as shown in Figure 15. Namely, the VDES transceiver overhears the vessels’ legacy AIS (Automatic Identification System) transmission and gets the current position, navigational status, type of manoeuvre, speed over ground and heading. The bottom right frame prints the latest value measured by the motion sensors of the Ship Health Monitoring: yaw, pitch, roll, surge, sway and heave. In addition, the top left corner indicates the current evacuation status, where “0” means “Normal Status”.

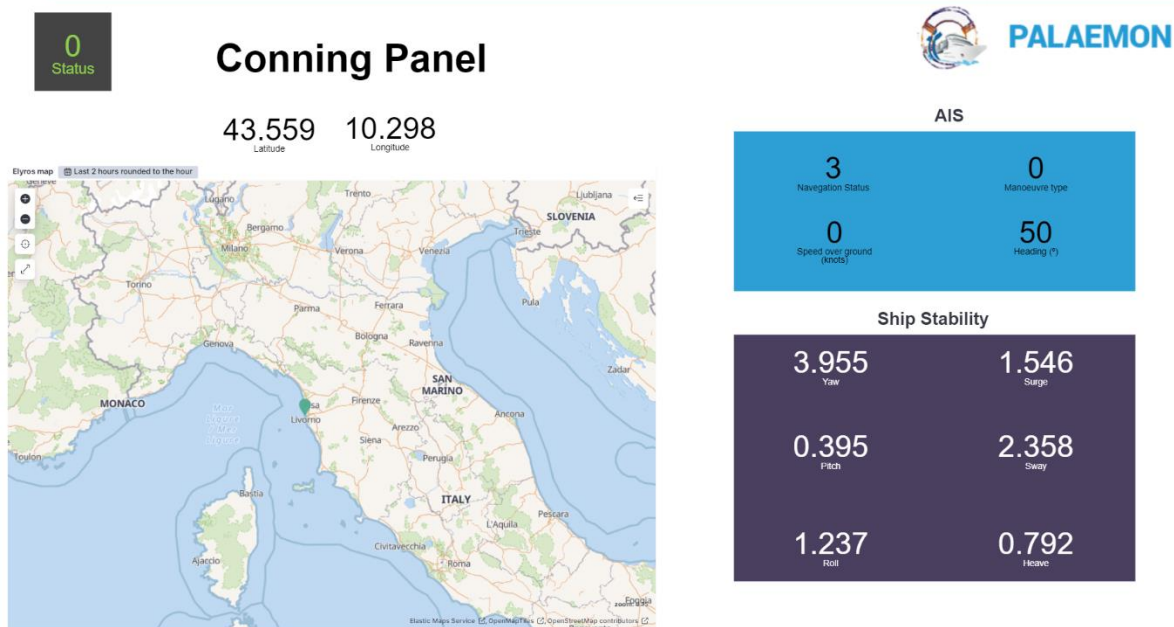


Figure 15. PALAEMON Canvas - Conning Panel (emulation)

Figure 16 shows a sample of the passengers list. In a normal situation, this information is sensitive and, according to GDPR regulation [29], personal data shall not be shared with external services. Technically speaking, this should be either save in a different and dedicated database or anonymized before its storage. In the context of PALAEMON, we have opted for the second option. PaMEAS, when users (i.e., passengers and crew) register into the platform, handles the information and saves an anonymized copy in Elasticsearch. Of course, the process is reversible, but only PaMEAS services have the capacity to “undo” the process.

Surname	Name	Birth	Gender	Embarkation Port	personalInfo.disembarkationPort	personalInfo.countryOfResidence	personalInfo.mobilityIssues	personalInfo.preferredLanguage
Yost	Oleta	61	female	Piraeus	Chania	Greece		EN
Howe	Alycia	4	female	Piraeus	Chania	Greece		EN
Bashirian	Brendon	38	male	Piraeus	Chania	Greece		EN
Simonis	Aurore	24	female	Piraeus	Chania	Greece		EN
Garcia	Jon	29	male	Piraeus	Chania	Spain	Yes	ES

Figure 16. PALAEMON Canvas - Passenger list

Nonetheless, when people’s lives are in danger, GDPR restrictions are disclosed and the use of this personal data becomes possible. Attending to GDPR’s Recital 46 [30]: “The processing of personal data should also be regarded to be lawful where it is necessary to protect an

*interest which is essential for the life of the data subject or that of another natural person*". With this example we present the possibility of "opening" this sensitive data in some critical situations (PaMEAS supports a RESTful API that returns this information "un anonymized"). To give the reader an example, a passenger with disabilities may require special assistance and may need to directly ping a responsible person (e.g., familiar, doctor, etc.). As a matter of fact, the passenger list presented in the figure was created for demonstration purposes. As we have stated above, the information is saved anonymized, and Kibana and Canvas cannot use PaMEAS' API to fetch the data.

Still on the personal plane, Figure 17 display a sample of the biometrics information generated by one of the Smart Bracelets, worn by an arbitrary passenger (anonymous in this case, as the evacuation status is "0"). Another feature of these wristbands is, as shown in the figure, the possibility of either automatically or manually (there is a physical button that should be pressed for 5 seconds) trigger an alarm, in this case associated to an "Accidental Fall". As a side note, this information can be processed by the SRAP, which can perform an individual risk analysis based on biomedical information, i.e., a level of oxygen saturation below a particular threshold may mean that a person has fainted, hence assistance is necessary.

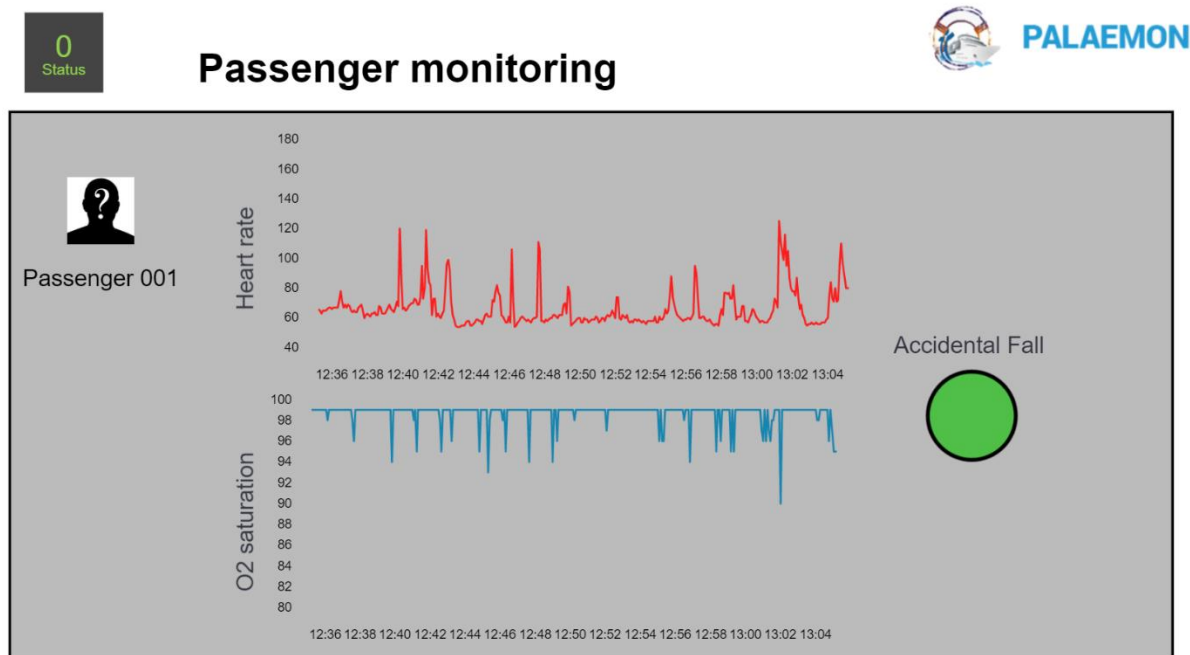


Figure 17. PALAEMON Canvas - Passenger monitoring

Canvas also demonstrates that the PALAEMON system is able to spread messages to the outer world. As we thoroughly cover in Deliverable D7.3 (Deployment of VDES – v2) [19], we have designed and developed a handful of VDES transceivers, used to send messages from ship to shore and vice versa. Figure 18 presents some distress messages generated via VDES transmissions, where the current situation: ship name, position, incident type, assistance required, total number of passengers and crew, injured persons, damage of the ship. The reader shall take into account that the last two are synthetic data and only illustrates potential information that may be conveyed to Port Authorities or Search & Rescue teams.

To the best of our knowledge, the format of these messages (i.e., digitalized) lack any kind of regulation or standard. Current actions are limited to transmitting VHF radio messages (i.e., voice) over a reserved channel, according to Global Maritime Distress and Safety System

(GMDSS)[31]. In this project, we present our own vision of what we consider a breakthrough in terms of mayday message format.



## Outgoing signals (VDES)



Time	Ship Name	Position	Incident Type	Assistance?	# Passengers	#Crew	#Injured	Damage %
2021-01-26T15:25:00+01:00	Costa Concordia	42.3717	Contact - Breach - Blackout	None	4229	1023	0	75%
2021-01-26T15:25:00+01:00	Costa Concordia	42.3717	Contact - Breach - Blackout	None	4229	1023	0	75%
2021-01-26T15:25:00+01:00	Costa Concordia	42.3717	Contact - Breach - Blackout	None	4229	1023	0	75%
2021-01-26T15:25:00+01:00	Costa Concordia	42.3717	Contact - Breach - Blackout	None	4229	1023	0	75%
2021-01-26T15:25:00+01:00	Costa Concordia	42.3717	Contact - Breach - Blackout	None	4229	1023	0	75%
2022-10-13T15:21:48+02:00	Ship for VDES Demo	43.549484	Contact	None	1234	789	56	42%
2022-10-13T15:21:48+02:00	Ship for VDES Demo	43.549484	Contact	None	1234	789	56	42%
2022-10-13T15:31:08+02:00	Ship for VDES Demo	43.549484	Contact	None	1234	789	56	42%
2022-10-13T15:32:08+02:00	Ship for VDES Demo	43.549484	Contact	None	1234	789	56	42%
2022-10-13T15:33:08+02:00	Ship for VDES Demo	43.549484	Contact	None	1234	789	56	42%

Figure 18. PALAEMON Canvas - VDES outgoing signals

Finally, we must state once again that this Canvas cannot be seen as a usable asset to be part of the evacuation management. We have leaned on it for purely illustrative purposes, as it shows in a visual way that the information generated across the platform actually reaches the main database of the system (i.e., Elasticsearch).

#### 4.4 PIMM as system validator

*NOTE: The layout presented in this report may be altered in the forthcoming weeks, framed under WP8's activities. The last feedback we have received from end users during the last days drives us to carry out a handful of modifications tailored to the needs of PIMM's main users, ships' Master.*

As can be seen in Figure 19, PIMM serves as a visualization hub that prints the information coming from a number of data sources and services. The screenshot presents the dashboard, the "Decision Support Center". Aside this, there are two companion panels, one to show video streams coming from the smart cameras ("Video Streaming Center") and the other to display the main outputs of PaMEAS and SRAP ("Incident Assessment Center"). In a nutshell, these are the elements that are shown on the dashboard:

- **Ship Stability Toolkit (SST):** Motion predictions that come from the inputs of the Ship Health Monitoring, AIS position, speed and heading and weather forecast information (DANAOS' API).
- **Weather Forecast Toolkit (WFT):** From a Maritime Incident Dataset analysis done by KT [32] and applying some Machine-Learning-based classification algorithms, the system presents a dual good-bad practices approach as a function of the current and forecasted weather conditions.
- **Decision Support System (DSS):** According to the incident nature (e.g., fire, grounding, etc.), this components presented the Master a number of suggestions taken from International Safety Management (ISM) Code [33]. Displayed actions depend on the current status of the evacuation phase.
- **Smart Risk Assessment Platform (SRAP).** This component evaluates and quantifies the risk throughout three phases during the evacuation: situation assessment, mustering and pre-abandonment. The main purpose of these figures is to increase the awareness of the Master and Bridge Command Team so that they can take more



reliable decisions. The main outputs are split into two main parts: a general output (part of the “Decision Support Center” that suggests the Master to: 1- Sound the General Alarm (or not); 2- Abandon the Ship (or not). Besides, the “Incident Management Center” breaks down the vessel into a number of independent blocks (see Figure 20), where the risk level is handled independently. These so-called “blocks” do correspond with the concept of “geo-fences” handled by PaMEAS in their Real-Time Location System.

- **PALAEMON Evacuation Coordinator:** We described in Section 3 the role of this component as the main notifications manager during an incident. On PIMM, the Master is the only person actually capable of switching between evacuation states. The way to do that is as intuitive as clicking on the respective status, as shown in Figure 21, where all the chain is displayed. As a matter of fact, for this late version there are some internal status (i.e., Activation of Evacuation Protocol, Alert Passengers, Mustering Complete), but these are internal stages that update PaMEAS and SRAP’s internal operation modes (these modifications are not transmitter following the main `/evacuation-status` Kafka topic).
- **Smart Cameras:** As of the moment the General Alarm sounds (not before), the Smart Cameras stream real-time video that can be displayed on the PIMM.
- **PaMEAS:** This module is responsible for the evacuation management and interaction with passengers and crew. Thanks to the real-time location service, PaMEAS is aware of the current position of all the people. Moreover, the utilization of smartphones and smart bracelets allows a bidirectional communication with users, leading to the delivery of messages and recommendations.

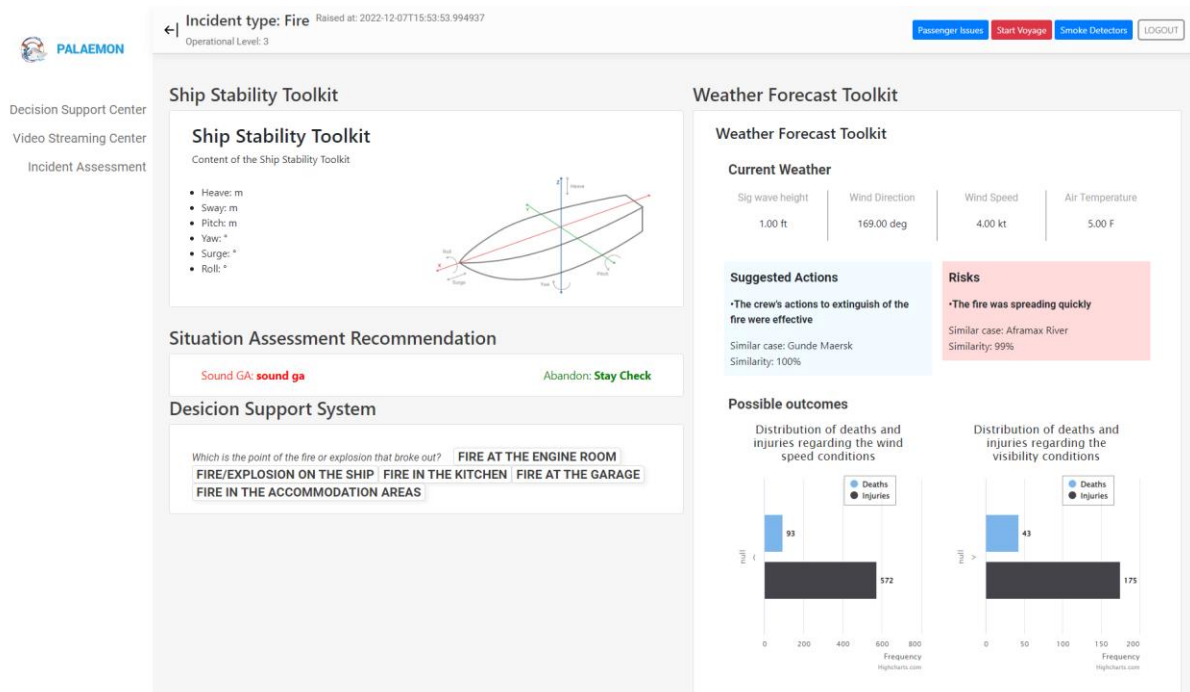


Figure 19. PIMM General View (Decision Support Center)



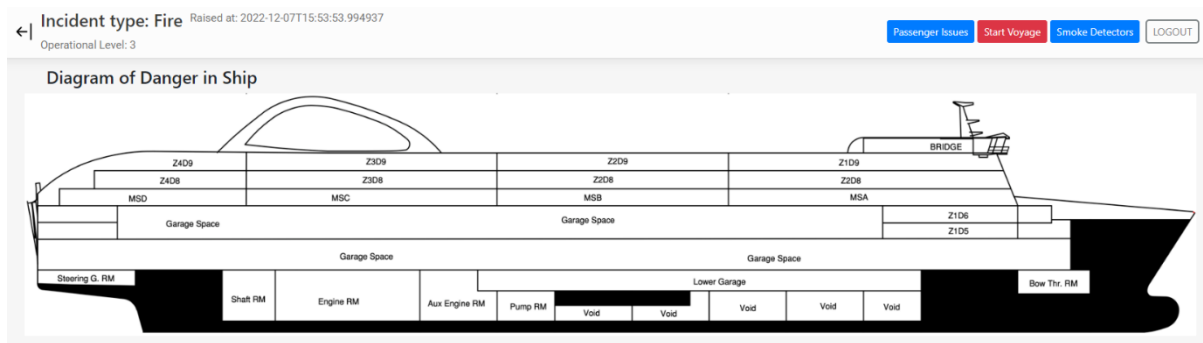


Figure 20. PIMM general view (SRAP Danger in ship)

Beside all the above elements, Smart Safety System's is indirectly connected to the PIMM. Namely, we use this component to initiate (emulate) an incident, e.g., fire, grounding, etc. When the SSS generates an event, PIMM display parses the type (displayed at the top left corner) and adapts the DSS suggestions to it.

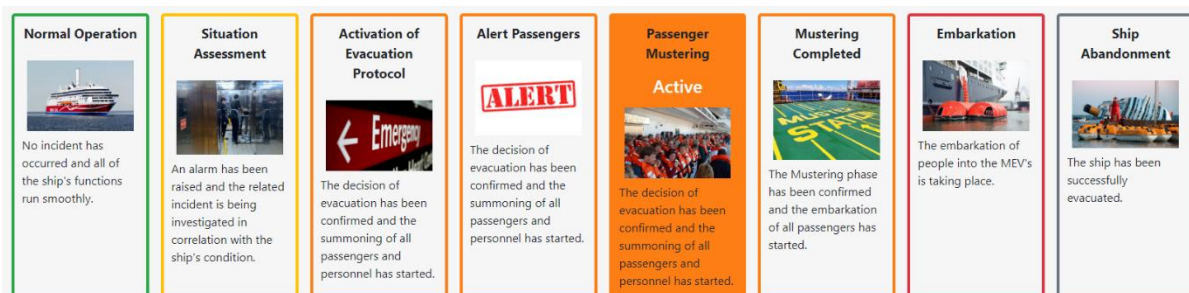


Figure 21. PIMM general view (Evacuation Status)

Under the hood, PIMM also behaves as an information sink that gets data from almost all services. Mainly, via Kafka, where PIMM subscribes to almost all topics that go across the PALAEMON platform. Nonetheless, the component offers a connection hook in the form of a RESTful API, where we can assess the trustworthiness of the information managed and displayed on the screen. Figure 22 presents the list of available services that we can use to retrieve information. The complete request/response exchange for all these methods can be found in Annex IV.

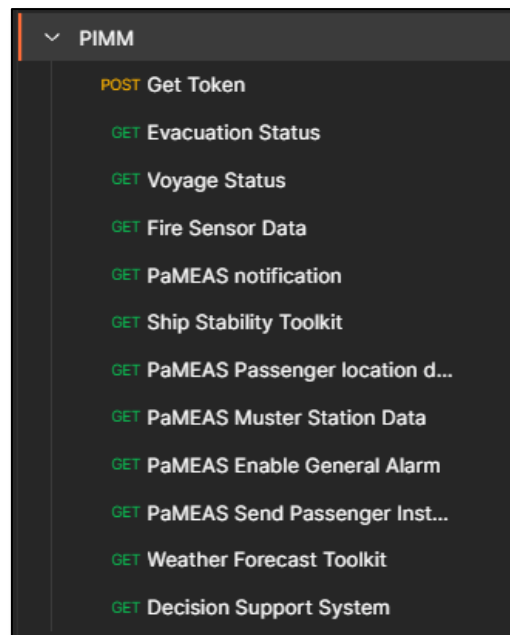


Figure 22. PIMM Postman collection - list of services

Finally, For a deeper description of this component and all its connections, the reader shall refer to D6.3 (PALAEMON Interfaces and HMIs toolkit) [34], D6.4 (Development of PALAEMON On-Board Decision Support System) [35] and D6.5 (PALAEMON Incident Management Module - PIMM) [21]. In addition, all the interactions with SRAP and PaMEAS will be reflected in WP8's deliverables, where real evacuation scenarios will be assessed.

#### 4.5 Grafana as system validator

Similarly to Kibana, Grafana [36] offers a visualization layer on top of raw information, hard to read by the non-expert eyes. Instead, it offers a plethora of widget and graphics that aggregate and present the information in a more friendly and interpretable manner. Moreover, Grafana supports direct connectors with Kafka, Elasticsearch, Prometheus, and a long etcetera. As for Elasticsearch, we have opted for Kibana, as we described in 4.1; on Kafka, we can see some of the main metrics in Figure 23.

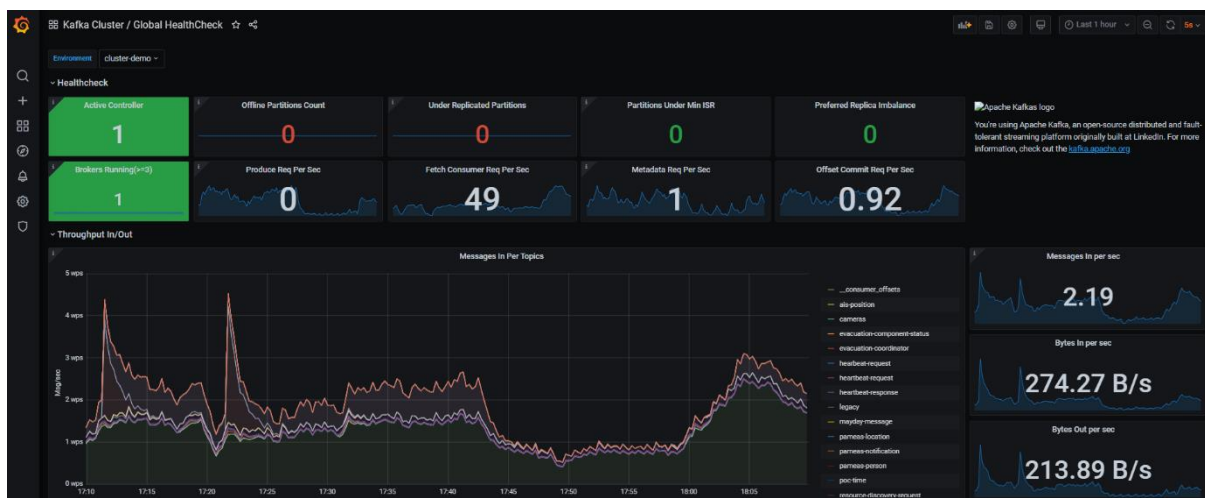


Figure 23. Grafana Kafka Monitoring Sample

A simple glance at the screenshot let us observe what parameters are monitored: broker status, traffic sent per topic – inbound and outbound, consumer requests per second, etc. The main outcome we can extract out of this figure (without delving into Kafka's technical particularities) is that there is a number of messages from many topics going across the PALAEMON platform.

#### 4.6 Voyage Report Generator as system validator

Finally, the last component to be “activated” during the evacuation process is the VRG. Even though it does not play an active role during the evacuation management, it is sensible to consider a part of it. In essence, it gathers all the information generated during a voyage (and during the evacuation) and prepares a thorough report, with all the data (raw from Elasticsearch and recorded videos from the Smart Cameras). Figure 24 spans the sequence diagram tackled by the VRG for this.

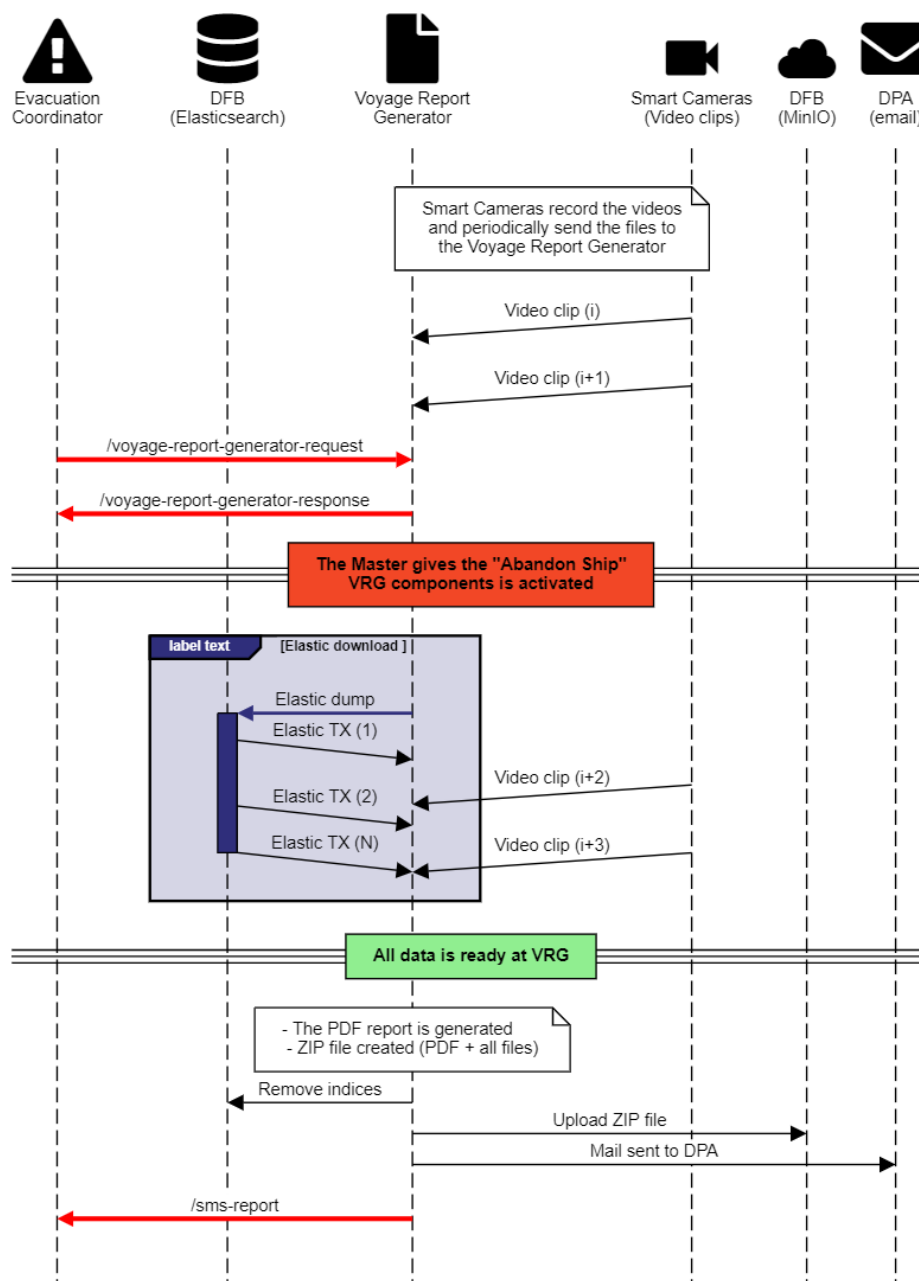


Figure 24. PALAEMON Voyage Report Generator sequence diagram



Despite the cue (for the VRG) officially starts when the PALAEMON Evacuation Coordinator sends the `/voyage-report-generator-request` notification (which is followed by an immediate `/voyage-report-generator-response` acknowledgement sent by the VRG), when the General Alarm is triggered (this corresponds to the “Passenger mustering” status switch), Smart Cameras proceed to record locally the video they are capturing<sup>13</sup>. Once a video clip is completed, they automatically transmit the file to the VRG.

Alongside this “background” operation, the VRG fetches all useful information from Elasticsearch (e.g., event logs, sensors data, passenger and crew lists – unanonymized, etc.). To do this, we have leaned on Elasticsearch dump [37], a command line tool that seamlessly grasps all documents stored from a list of indices. As the amount of information to dump may be large, this operation may take tens of seconds to finalize. In parallel, the component is still receiving video clips from the various smart cameras.

When all the data is available, the VRG takes all “displayable” data and generates a PDF report, where the information could be much more intuitively interpreted. The reader can refer to Annex V to check the visual information represented in the report. When the PDF is ready, a compressed (i.e., ZIP) file is created, including this document and all the data harvested in the process, that is, data from Elasticsearch, saved in a “json” file per index; moreover, video clips (.avi) are also attached. In order to deter the access from unexpected hands, the ZIP file is protected with a password. In real conditions, only the DPA shall be able to open and see this information.

As the information may be lost if we only keep it locally (e.g., a fire damages the hardware or the ship sinks), the VRG takes the ZIP file and sends it to two different destinations: 1- a MinIO instance deployed on PALAEMON-02 (shore infrastructure); 2- a digital copy is sent via mail to the predefined DPA.

Finally, the VRG sends a Kafka message under the `/sms-report` topic. This message notifies the Safety Management System that a new report is generated and can be found at the corresponding location in MinIO. As we will see in Section 5.5, the SMS automatically updates its internal database and appends a new incident report from the information received in this message. Needless to say, the SMS tool counts on its own access control list, and this information shall be only accessible by the same DPA, who should own a personal account on the platform.

Last but not least, it is worth highlighting that the whole VRG operation may take a couple of minutes, as the size of the full report may be in the order of Gigabytes. Considering that a real evacuation might last many hours, the storage of a number of video files from cameras, alongside information from sensors, etc. may end up in a massive amount of information.

Focusing on the technical validation per se, Code snippet 3 covers the messages exchanged between the PALAEMON Evacuation Coordinator and the Voyage Report Generator. We can see the format of the `/voyage-report-generator-request` and `/voyage-report-generator-response`, which basically include a timestamp, an originator identifier and a simple value that request the creation of the report and confirms it, respectively.

---

<sup>13</sup>Further versions may include multimedia data from more sources (AR glasses video and audio, messages from PaMEAS, video from ship’s Close Circuit Television, etc.).

```

=====
Received message (voyage-report-generator-request:0:158)
{'originator': 'Evacuation Coordinator', 'timestamp': '2022-11-
29T11:14:37.376981Z', 'action': 1}
=====
Received message (voyage-report-generator-response:0:209)
{'timestamp': '2022-11-29T11:14:37.666618', 'originator': 'VDR', 'status':
'OK'}
=====
Received message (sms-report:0:182)
{'Vessel': 'Elyros', 'UDE Type': 'Fire', 'UDE Description': 'Fire at Deck
8', 'Ocurrred On': '2022-11-29T11:16:17.128824', 'Port': 'Piraeus',
'objectName': 'Elyros_2022-11-29_1116_AHF7dNBDXWPBTND.zip', 'bucketName':
'palaemon-reports'}
=====

```

Code snippet 3. Voyage report generator message exchange

As for the third message, **/sms-report**, it carries more interesting information. We have adapted the format to the inputs the SMS tool needs to document an incident<sup>14</sup>. Namely, ID of the vessel, type of incident, timestamp, associated port, name of the MinIO Bucket and name of the file (ZIP).

We include below some screenshots that come to proof the correct operation of the component. Figure 25 shows the bucket (can be seen as a folder) dedicated to store all the VRG reports on MinIO (PALAEMON-02). We can see the huge size (>60 GB) of data stored).

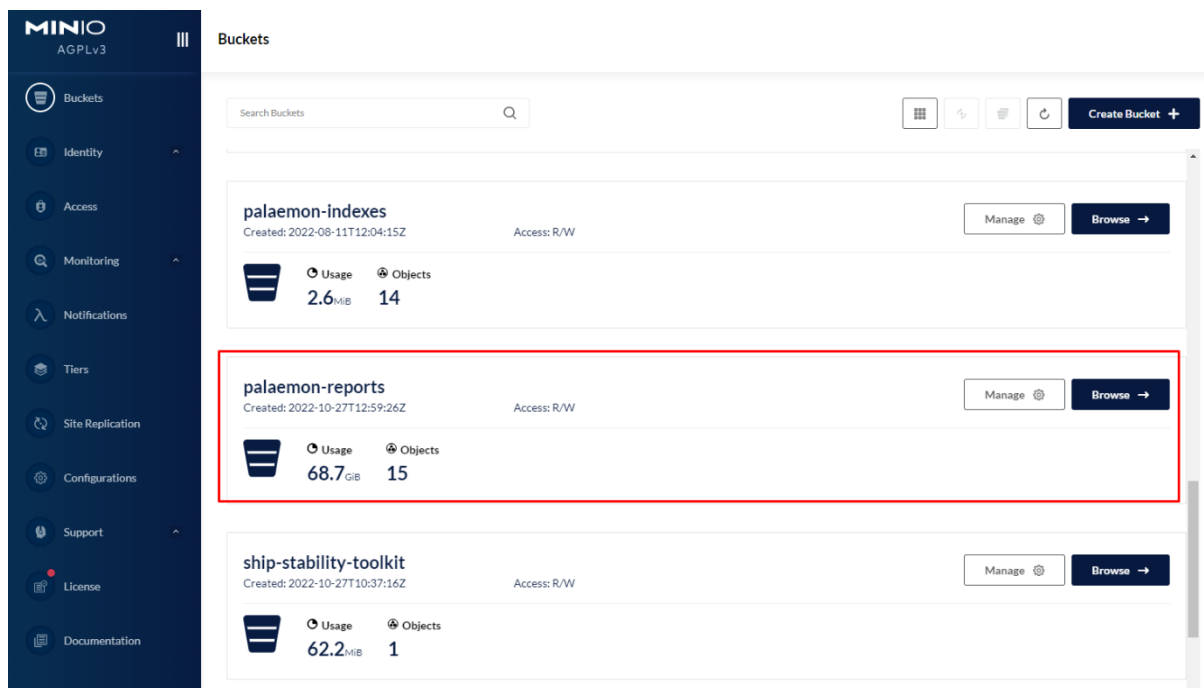


Figure 25. MinIO (part of DFB) repository in the cloud - PALAEMON buckets

<sup>14</sup> In a regular process, the SMS user has to manually input these parameters to register an incident.

Figure 26 “zooms into” the folder and shows some of the reports stored in the system. We can see framed in red the report we have used for this verification (we have captured the Kafka messages before).

Name	Last Modified	Size
VDR_Elyros_demo (1).zip	Thu Oct 27 2022 14:59:50 GMT+0200	4.1 MiB
Annual Overview of Marine Casualties and Incidents 2021.pdf	Thu Oct 27 2022 15:00:04 GMT+0200	5.8 MiB
Elyros_2022-10-27_1412_BDoEHhE8LkgEH.zip	Thu Oct 27 2022 16:12:11 GMT+0200	483.7 KIB
Elyros_2022-11-03_1630_AJo83UraXUhgZ3J.zip	Thu Nov 03 2022 17:31:07 GMT+0100	499.9 KIB
Elyros_2022-11-07_1702_AFDJuMLzemYSnXV.zip	Mon Nov 07 2022 18:02:34 GMT+0100	499.9 KIB
Elyros_2022-11-09_1158_8zxkgEp5rmbANqs.zip	Wed Nov 09 2022 12:58:54 GMT+0100	500.2 KIB
Elyros_2022-11-09_1328_6Smm9uVpS4ZFDr8.zip	Wed Nov 09 2022 14:28:27 GMT+0100	496.5 KIB
Elyros_2022-11-28_1531_422Xh3kFdwUyX8b.zip	Mon Nov 28 2022 16:31:56 GMT+0100	510.0 KIB
Elyros_2022-11-28_1706_AHSDqKhYeqx47K.zip	Mon Nov 28 2022 18:06:43 GMT+0100	510.0 KIB
Elyros_2022-11-29_0909_WPVGTfXnMzWbyie.zip	Tue Nov 29 2022 10:09:25 GMT+0100	510.2 KIB
Elyros_2022-11-29_1020_867doP7ZlZNXV6R.zip	Tue Nov 29 2022 11:20:37 GMT+0100	510.0 KIB
Elyros_2022-11-29_1116_AHF7dNBDXWPBTND.zip	Tue Nov 29 2022 12:16:55 GMT+0100	85.6 MiB

Figure 26. MinIO "palaemon-reports" bucket zoom

If we download the report and try to unzip the file, a popup windows demands us to type the password, as shown in Figure 27.

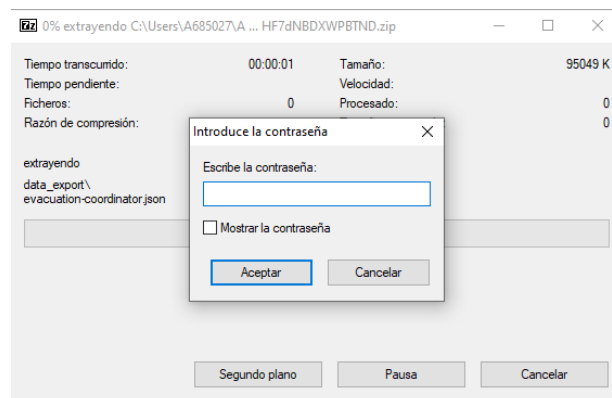


Figure 27. Voyage report zip file password request (Spanish Operative System)

Though nobody but DPAs should have access to these reports, for demonstration purposes, we have created mocked reports that do not contain any real or sensitive information. After uncompressing the file, Figure 28 reflects the content of the main folder (root path). In addition, we have a dedicated folder to dump all the videos transmitted by the different cameras (Figure 29). In this particular example, video clips are generated every 30 seconds (approximately). The reader could easily extrapolate, according to the size of an individual 30" clip (between 5 and 9 Megabytes), the overall space required to store hours of footage that come at the same time from a handful of different cameras.

Nombre	Estado	Fecha de modificación	Tipo	Tamaño
smart-cameras	✓	29/11/2022 11:16	Carpeta de archivos	
ais-position.json	✓	29/11/2022 11:16	Archivo JSON	875 KB
Elyros_2022-11-29_1116_AHF7dNBDXWP...	🔄	29/11/2022 11:16	PDF Document	344 KB
evacuation-coordinator.json	✓	29/11/2022 11:16	Archivo JSON	87 KB
example-passenger-list.json	✓	29/11/2022 11:16	Archivo JSON	13 KB
mayday-message.json	✓	29/11/2022 11:16	Archivo JSON	301 KB
pameas-passenger-list.json	✓	29/11/2022 11:16	Archivo JSON	11 KB
pameas-person.json	✓	29/11/2022 11:16	Archivo JSON	61 KB
ship_particulars.json	✓	29/11/2022 11:16	Archivo JSON	1 KB
shm-alarm.json	✓	29/11/2022 11:16	Archivo JSON	9 KB
smart-bracelet-alarm-2021.01.26.json	✓	29/11/2022 11:16	Archivo JSON	1 KB
smart-camera-alarm.json	✓	29/11/2022 11:16	Archivo JSON	1.030 KB
smart-safety-system.json	✓	29/11/2022 11:16	Archivo JSON	66 KB
srap.json	✓	29/11/2022 11:16	Archivo JSON	186 KB
test-poc-2022.02.08.json	✓	29/11/2022 11:16	Archivo JSON	2 KB
weather-service.json	✓	29/11/2022 11:16	Archivo JSON	37 KB

Figure 28. Voyage Data Report content (Spanish Operative System)

Nombre	Estado	Fecha	Tipo	Tamaño	Duración
camera1_20221129_121610.avi	✓	29/11/2022 11:16	Clip de víd...	7.628 KB	00:00:20
camera2_20221129_121518.avi	✓	29/11/2022 11:15	Clip de víd...	6.445 KB	00:00:20
camera2_20221129_121551.avi	✓	29/11/2022 11:15	Clip de víd...	6.792 KB	00:00:20
camera2_20221129_121625.avi	✓	29/11/2022 11:16	Clip de víd...	5.356 KB	00:00:20
camera3_20221129_121448.avi	✓	29/11/2022 11:14	Clip de víd...	5.166 KB	00:00:20
camera3_20221129_121520.avi	✓	29/11/2022 11:15	Clip de víd...	5.568 KB	00:00:20
camera3_20221129_121553.avi	✓	29/11/2022 11:15	Clip de víd...	5.526 KB	00:00:20
camera3_20221129_121627.avi	✓	29/11/2022 11:16	Clip de víd...	5.261 KB	00:00:20
camera4_20221129_121422.avi	✓	29/11/2022 11:14	Clip de víd...	8.151 KB	00:00:20
camera4_20221129_121454.avi	✓	29/11/2022 11:14	Clip de víd...	9.722 KB	00:00:20
camera4_20221129_121527.avi	✓	29/11/2022 11:15	Clip de víd...	8.154 KB	00:00:20
camera4_20221129_121603.avi	✓	29/11/2022 11:16	Clip de víd...	9.476 KB	00:00:20
camera4_20221129_121637.avi	✓	29/11/2022 11:16	Clip de víd...	8.795 KB	00:00:20

Figure 29. Voyage Data Report smart-cameras folder content (Spanish Operative System)

## 5 Individual validation

This section addresses individual verifications of the main software components. For the sake of simplicity, we stick to Kafka messages logging as the main check to demonstrate that components do what they are supposed to do.

*All these components have their own individual validation reporting reported in their own deliverables and/or will be complemented in WP8's reports (mainly PaMEAS, whose complexity will be addressed aside this document).*

### 5.1 Smart Bracelets

In terms of access, smart bracelets presented a technical limitation that led to seeking an alternative way to establish a communication with these devices. Due to hardware and battery restrictions, the utilization off SSL certificates (and all the operations that go alongside the authentication process) was ditched. Instead, we have deployed a MQTT broker as part of the DFB Access level. As for the security, there is a dedicated user and password protection.

This means that we must add a middleware that translate between Kafka and MQTT and vice versa; otherwise, the communication with the rest of the PALAEMON components would not be feasible. To deal with these on-the-fly Kafka  $\rightleftharpoons$  MQTT mapping (among other features), the DFB Access includes an instance of Apache NiFi [14]. Without digging into the technical details, this framework permits a straightforward “low-code” configuration of the flows, where we subscribe to MQTT or Kafka topics, transform the information if needed and publish on the other (if the subscription was on Kafka, it publishes on MQTT, and the other way around). Figure 30 shows a couple of examples that take bracelets' data and forwards to Kafka.

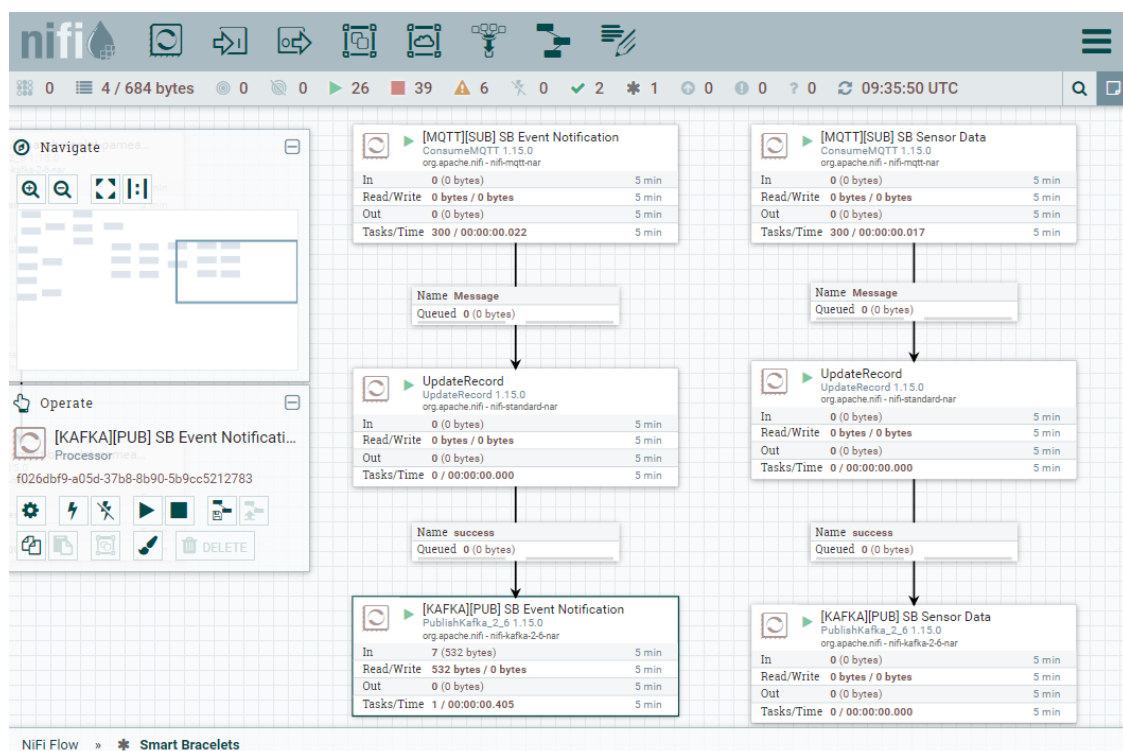


Figure 30. NiFi MQTT to Kafka flow sample (screenshot)

We have configured as many flows as needed ensuring to span all the topics that have to do with smart bracelets. For demonstration purposes, Figure 31 shows a MQTT message that

comes from the translation of the **/heartbeat-request** topic. As a matter of fact, we have used a Windows tool called MQTT.fx [38] to subscribe to all topics and visualize the data.

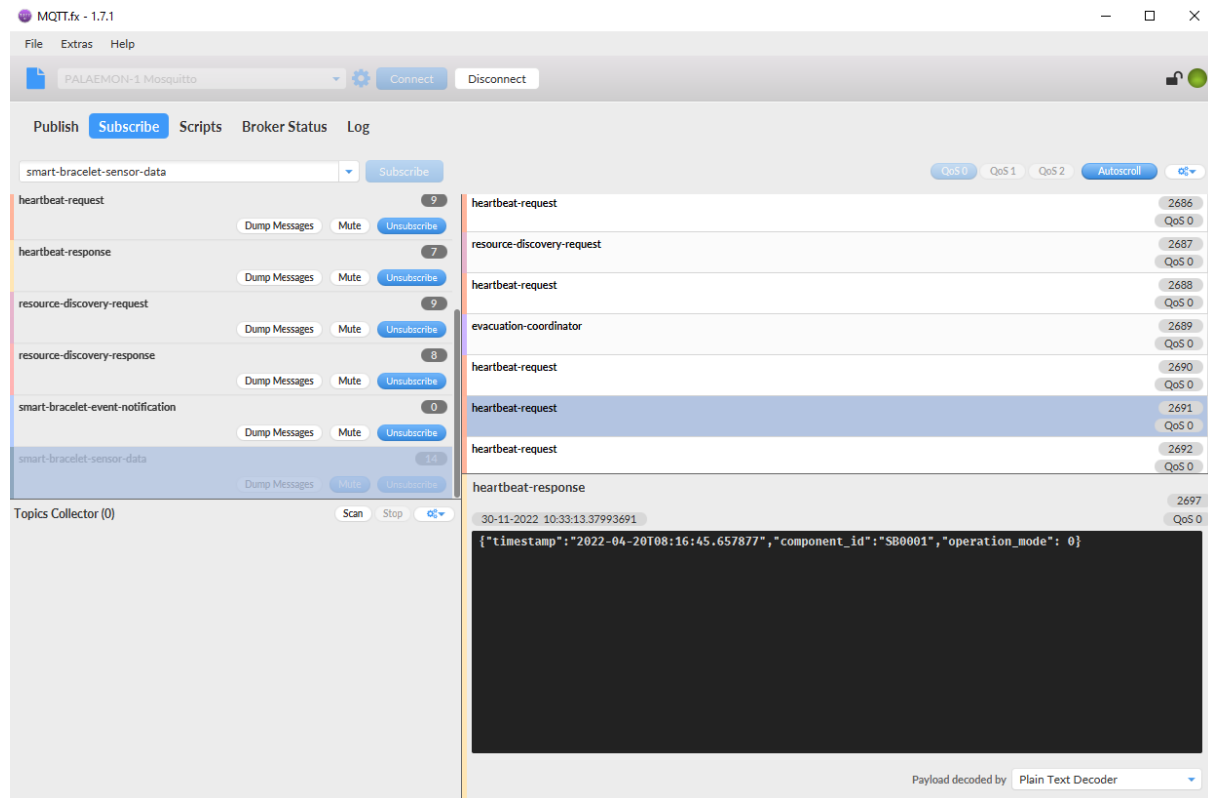


Figure 31. Smart Bracelets MQTT communication sample (/heartbeat-request)

Following this example, the pipeline would be as follows:

1. [Kafka] PALAEMON Evacuation Coordinator generate the **/heartbeat-request** message.
2. [MQTT] NiFi translates from Kafka and sends the same message on as **/heartbeat-request** topic.
3. [MQTT] One of the smart bracelets, “SB0001” receives the request and proceed to reply via **/heartbeat-response** topic.
4. [Kafka] NiFi intercepts the MQTT transmission and forwards the same message through Kafka (as shown in Code snippet 4).

```
=====
Received message (heartbeat-response:0:484215)
{'timestamp': '2022-04-20T08:16:45.725', 'component_id': 'SB0001',
'operation_mode': 0, '_topic': 'heartbeat-response', '_qos': 0, '_isDuplicate':
False, '_isRetained': False}
=====
```

Code snippet 4. Smart Bracelets Kafka sample (/heartbeat-response)



A complementary example is shown in Figure 32 and Code snippet 5. In this case, the original source is “SB0001”, which sends some biometrics through the `/smart-bracelet-sensor-data` topic (MQTT).

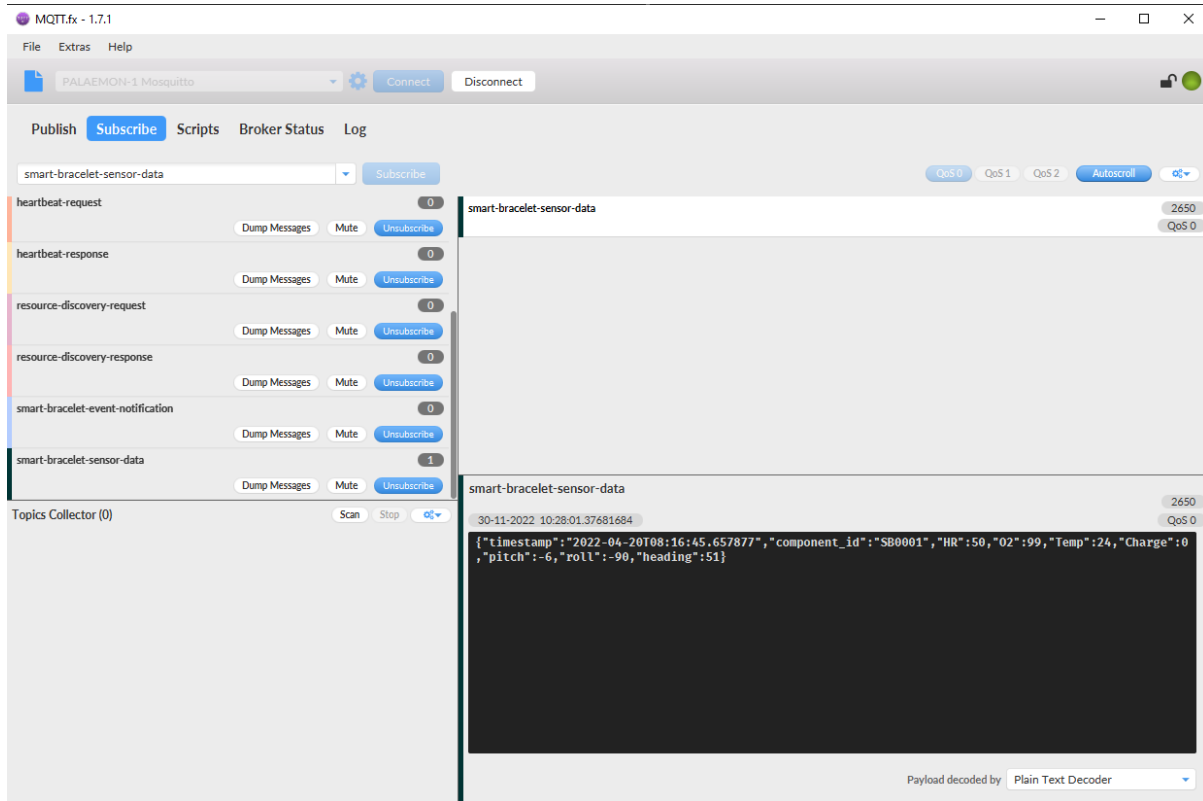


Figure 32. Smart Bracelets MQTT message sample (`/smart-bracelet-sensor-data`)

As the output of the pipeline, NiFi generates the final `/smart-bracelet-sensor-data` on Kafka, which reaches the DFB and is stored into Elasticsearch. In fact, we demonstrated the integration of this flow in Section 4.3, where we plotted the heartbeat and oxygen saturation of an arbitrary user on Kibana Canvas (Figure 17 in page 27).

```
=====
Received message (smart-bracelet-sensor-data:0:12395)
{'timestamp': '2022-04-20T08:16:45.657877', 'component_id': 'SB0001', 'HR': 50,
'02': 99, 'Temp': 24, 'Charge': 0, 'pitch': -6, 'roll': -90, 'heading': 51}
=====
```

Code snippet 5. Smart Bracelets Kafka sample (`/smart-bracelet-sensor-data`)

Besides this uplink message, smart bracelets also notify when an event is produced (either the device has detected an accidental fall or the user has pressed the button of the wristband for five seconds), using the `/smart-bracelet-event-notification` to that effect. In addition, PaMEAS can also produce simple commands to be printed on the bracelets' screens by means of the `/smart-bracelet-pameas-evac-msg` topic. Some visual examples of how these messages are translated into simple visual inputs can be found in Annex VI. As a matter of fact, these snapshots are taken on the v3 of the wristbands. WP8's deliverables will dig into PaMEAS' messaging and notification system.

## 5.2 Smart Cameras

Under normal conditions, smart cameras operate as a standalone component, where the cameras capture the video and the companion Processing nodes execute some computer-vision algorithms. In terms of integration with the PALAEMON Communications platform, cameras send (asynchronously) every time the number of people detected changes using the `/smart-camera` topic, represented in Code snippet 6.

```
=====
Received message (smart-camera:0:97432)
{'component_id': 'camera-02', 'people_count': 1, 'timestamp': '2022-11-
23T12:02:42.216186'}
=====
=====
Received message (smart-camera:0:97433)
{'component_id': 'camera-02', 'people_count': 0, 'timestamp': '2022-11-
23T12:02:44.607789'}
=====
```

*Code snippet 6. Smart Cameras Kafka sample (/smart-camera)*

Furthermore, as of the ship evacuation status switches to “Situation Assessment”, cameras modify their operation mode and add three additional features to their workflow.

1. Detected events – people trapped / stampede (via `/smart-camera-alarm` Kafka topic, as shown in Code snippet 7).
2. Real-time video streaming from the cameras is visible on PIMM’s “Video Streaming Center. In the example shown in Figure 33).
3. Processing nodes proceed to record the video and automatically transmit the files (.avi) to the VRG, as we described in Section 4.6.

```
=====
Received message (smart-camera-alarm:0:7713)
{'component_id': 'camera-02', 'event_code': 1, 'timestamp': '2022-11-
29T12:33:33.966009'}
=====
=====
Received message (smart-camera-alarm:0:7714)
{'component_id': 'camera-04', 'event_code': 3, 'timestamp': '2022-11-
29T12:35:46.550523'}
=====
```

*Code snippet 7. Smart Cameras Kafka sample (/smart-camera-alarm)*



## SMART VIDEOSURVEILLANCE SYSTEM

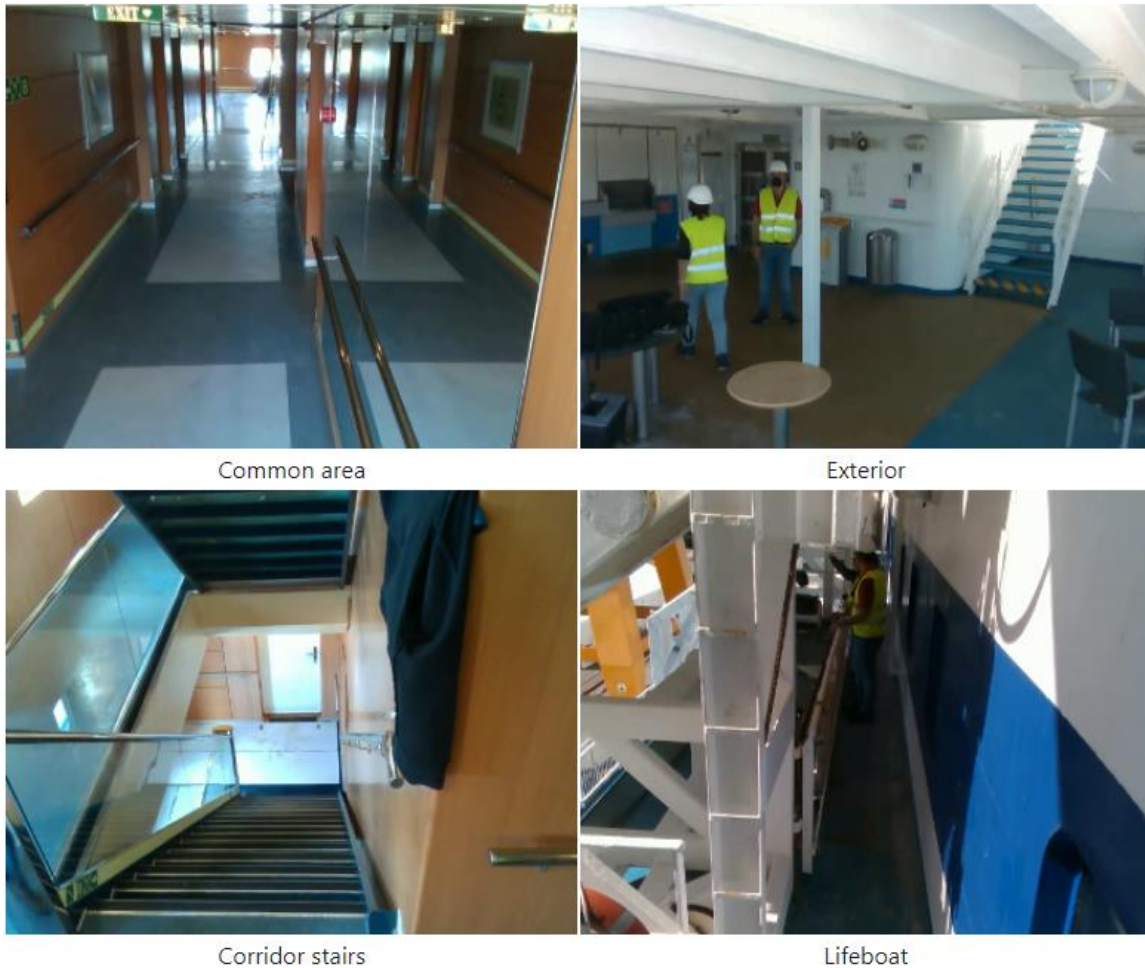


Figure 33. Smart Cameras main dashboard layout (PIMM's Video Streaming Center)

### 5.3 Smart Safety System

In the context of an evacuation, SSS offers a drag-and-drop graphical user interface that crew members can use (through e.g., a smartphone or tablet) to notify an event and pinpoint at their exact position (in terms of deck and position in X and Y coordinates). As we can see in Figure 34, this component presents three main areas: 1- the central part corresponds with the visualization of the blueprint of a deck, which we can select by clicking (or touching) 2- an option at the right part of the layout. 3- Finally, the bottom part displays some icons, through which we can select the type of incident. Following their order of appearance, from left to right, these are the notifications users can communicate:

- Officer required at a particular position
- Crew member (regular) required
- Hotel staff required
- Fire detected
- Smoke detected
- Flooded area
- High voltage

- Medical assistance needed
- Person with disability needs assistance
- New muster station (ad-hoc)
- Restricted area (no-go area)
- Checked area

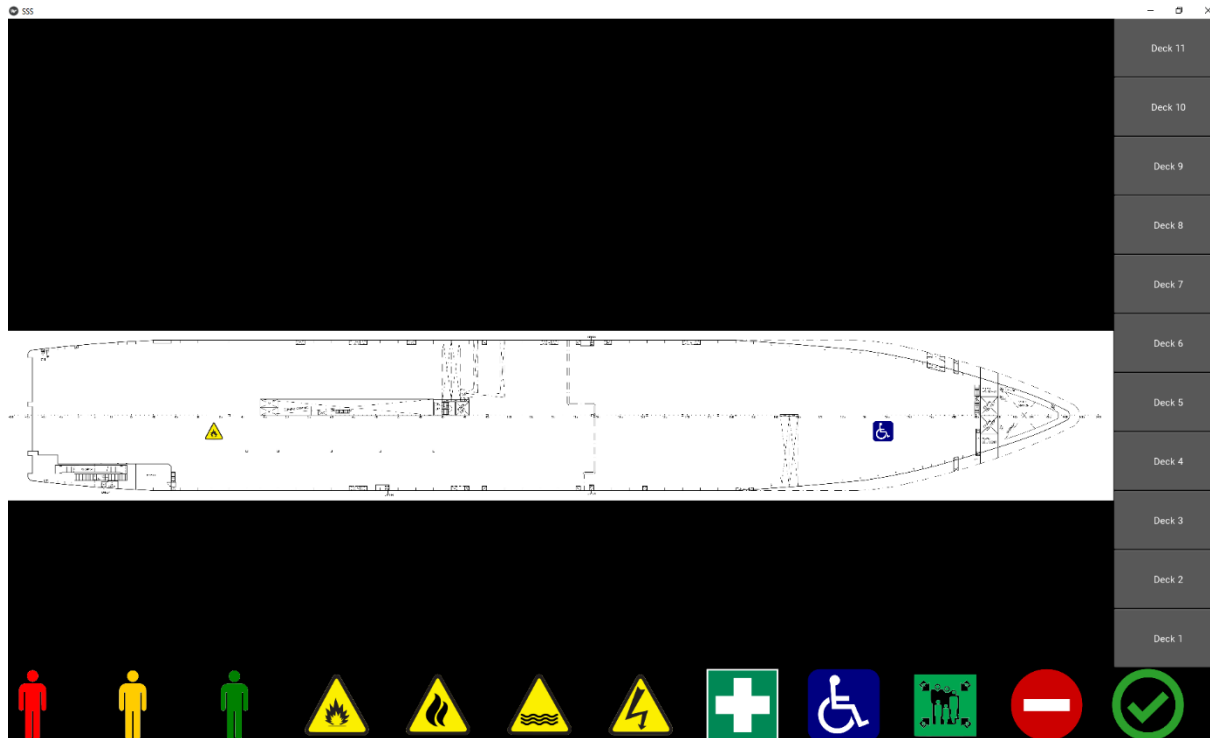


Figure 34. Smart Safety System Main layout

These notifications are transmitted across the `/smart-safety-system` topic (Kafka), thus subsequent smart services can gather the information and react accordingly. By taking a look at Code snippet 8, we can easily appreciate the structure of the event, where we observe the type (from the list given above), the timestamp, and the position, combination of deck number plus X and Y position.

```
=====
Received message (smart-safety-system:0:734)
{'type': 'Fire', 'timestamp': '2022-11-23T10:55:17.737907+00:00', 'deck': 4,
'position_x': 24.48, 'position_y': -2.81}
=====
```

Code snippet 8. Smart Safety System Kafka sample (`/smart-safety-system`)

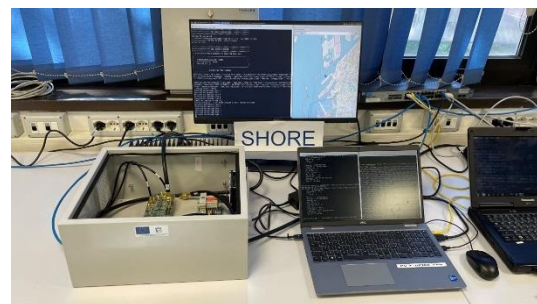
## 5.4 VDES transceiver and gateway

Nowadays we can state that VDES systems have deservedly become the spiritual successor of legacy AIS communications, as it presents a more efficient use of the radio spectrum, permitting to deliver more information (i.e., payload) using less bandwidth in ship-to-ship, ship-to-shore and even ship-to-satellite radio links<sup>15</sup>.

In PALAEMON, we have designed and manufactured up to three different VDES Transceivers (prototypes), including all the hardware (antennae, etc.), Software Defined Radio (SDN) and a middleware component, the VDES gateway, that bridges with the PALAEMON platform. Figure 35 shows two of the three transceivers that we have used for the validation tests. One of them plays the roles of a transceiver installed onboard Elyros; the second one to be part of an installation ashore. The third one was kept as a backup solution in case of major necessity.



a) VDES Transceiver (Ship)



b) VDES Transceiver (Shore)

Figure 35. VDES Transceiver Prototypes (ship & shore)

In terms of validation, Kafka streams come to demonstrate that the end-to-end VDES data flow works as expected, as they appear in both sides of the communication, acting as the intermediate between DFB and VDES. VDES transceivers have direct participation in three different scenarios:

For the first use case, the receiver overhears AIS transmissions from nearby transponders (vessels, port authorities). If we parse the identifier and match to that of the ship, we can extract the information of the vessel itself, thus legacy AIS data will be seamlessly integrated as part of the PALAEMON system. Code snippet 10 presents an arbitrary message sent through the `/ais-position` topic.

```
=====
Received message (ais-position:0:6126)
{"cog" : 241.8000030517578, "cog_available" : true, "latitude" :
43.55862808227539, "longitude" : 10.297853469848633, "manoeuvre_ind" : 0,
"nav_status" : 3, "pos_accuracy" : 0, "rate_turn_rotais" : 0, "sog" : 0.0,
"sog_available" : false, "th_available" : true, "timestamp" : "2022-11-
10T12:45:43.925180", "true_heading" : 50, "user_id" : 247343700}
=====
```

Code snippet 9. VDES Gateway Kafka sample (/ais-position)

<sup>15</sup> Ship-to-satellite communications is a feature not covered by PALAEMON's prototypes.

The second feature gathers weather information forecasts (a new prediction is generated every three hours) from an external service (accessible through an Internet connection). The data workflow is described with all details in Section 5.8. As output, the raw object appears as shown in Code snippet 10.

```
=====
Received message (weather-service:0:12783)
{'Timestamp': '2022-12-02 06:00:00', 'combDirectionDegrees': 86.61, 'combPeriod':
2.87, 'combSWHMeters': 0.3, 'currentsDirectionDegrees': 45.0, 'currentsSpeedKnots':
0.05832, 'hum%': 82.5, 'iceCover': 0.0, 'lat': 45, 'lon': 14, 'mslhPa': 1017.65,
'sea': true, 'swellDirectionMeters': 90.74, 'swellPeriod': 3.07, 'swellSWHMeters':
0.27, 'tempCelciusDegrees': 12.35, 'visKm': 24.135, 'wavesDirectionDegrees':
58.44, 'wavesPeriod': 1.68, 'wavesSWHMeters': 0.11, 'windDirectionDegrees': 56.33,
'windSpeedKnots': 14.71608}
=====
```

*Code snippet 10. VDES Gateway Kafka sample (/weather-service)*

The third and last scenario was briefly outlined in the section where we spoke about Kibana Canvas as system validator. When the General Alarm sounds, the PALAEMON platform periodically sends distress or mayday messages through the VDES radio interface. This way, the transmitted data might reach close-by vessels or port authorities, thus speeding up the search and rescue process, if needed.

```
=====
Received message (mayday-message:0:4111)
{"via" : "VDES", "ship_name" : "Ship for VDES Demo", "ship_position" :
[43.549484, 10.297393], "timestamp" : "2022-10-28T10:21:08.524562138Z",
"incident_type" : "Contact", "assistance_required" : "None",
"number_passengers" : 1234, "injured_people" : 56, "crew_on_board" : 789,
"weather_conditions" : "Unknown", "damage_extent" : "42%"}
=====
```

*Code snippet 11. VDES Gateway Kafka sample (/mayday-message)*

## 5.5 Safety Management System

When it comes to deal with safety procedures and according to the International Safety Management (ISM) shipping companies have to develop and maintain a platform to manage all safety/contingency plans, policies and strategies. In the scope of PALAEMON, SMS tool presents two instances (one onboard and another one ashore) that cope with the addition and updates of safety information, in the form of manuals or checklists, for example. The platform is presented to the user as a graphical user interface; any information query or addition could be done in a very intuitive manner. It goes without saying that only allowed users have access to the information.

As for the extent of the different instance, whereas that of onboard focuses on a single ship (i.e., Elyros in this project), ashore platform controls and manages the whole fleet. Said in other words, SMS behaves as a centralized system where the core (i.e., shore) is always synchronized with the information generated on the onboard instances (i.e., each ship should have its own SMS tool up-and-running).

In order to demonstrate the main breakthrough achieved in the project, we have come up with a link between this tool and the Voyage Report Generator. Thanks to this binding, the PALAEMON system yields a semi-automatic registration of incidents<sup>16</sup>, where the user intervention is not required. In a nutshell, when the VRG generates a report and uploads the final file (i.e., ZIP) to the repository in the cloud (i.e., MinIO), the underlying Kafka notification (i.e., `/sms-report` topic) warns the SMS tool that a complete report is available and can be retrieved. The content of the message (Code snippet 3 in page 34) contains some basic information that is used to automatically create the corresponding entry in the SMS database. The full VRG's report is automatically downloaded, as shown in Figure 36 and all files are inserted in the placeholder produced before.

```

ic| topic_list: ['sms-report']
kafka.consumer.group.KafkaConsumer object at 0x0f89f040>
=====
Received message (sms-report:0:3)
{'Vessel': 'Elyros', 'UDE Type': 'Fire', 'UDE Description': 'Fire at Deck 8', 'Occurred On': '2022-10-06T13:10:44.768434', 'Port': 'Piraeus', 'objectName': 'Elyros_2022-09-09_0833_50qXf3QPlak9ck', 'bucketName': 'palaemon-test', 'vslId': 'SPK', 'timestamp': '2022-09-08T13:43:00.888955', 'occured_on': '2022-09-08 13:43:00', 'acc_flag': '1'}
=====
Downloading MinIO File . . .
===== Bucket: palaemon-test | Created: 2022-06-23 06:34:39.335000+00:00 =====
Annual Overview of Marine Casualties and Incidents 2021.pdf
Elyros_2022-10-06_1548_5V2j3HqD227m0f.zip
Elyros_2022-10-06_1150_JM268HnyvEY6Gvp.zip
Elyros_2022-10-06_1310_6175CL4PcARiQKf.zip
VDR_Elyros_demo.zip
=====
Inserting into DB . . .
SPK_0000000659 Q:\ism\accidents\Elyros_2022-09-09_0833_50qXf3QPlak9ck.pdf
===== DONE !

```

Figure 36. Safety Management System - Report updated to cloud repo (console proof)

Figure 37 corroborates what we have seen with the previous command-line logging. We can observe in the main user interface that the report has been successfully uploaded onto the SMS tool (i.e., the unique ID of the file matches). Moreover, the tool offers a built-in PDF viewer to read the report created by the PDF.

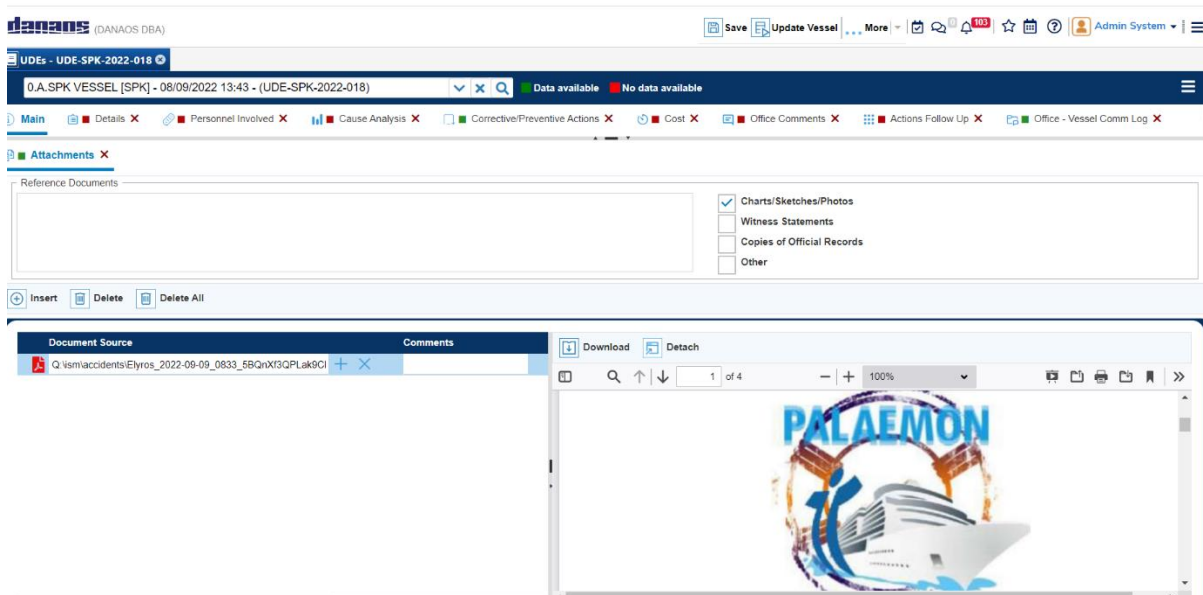


Figure 37. Safety Management System - ISM Dashboard visualization of report generated by the VRG

<sup>16</sup> Though to be used for a post-incident analysis carried out by an authorized DPA.



## 5.6 Ship Health Monitoring

This component is composed of a handful of motion sensors that monitor six degrees of freedom by means of built-in accelerometers, magnetometers and gyroscopes. These devices measure accelerations in the main three axes (i.e., X, Y and Z), and the corresponding three angular velocities between them. Though the SHM module mainly forward this raw information, other services may combine or aggregate the data in order to calculate more advanced phenomena, like deflection or torsion of the hull.

In terms of demonstration, the sensors generate a measurement every 5 seconds. SHM's processing node<sup>17</sup> gather the data and proceed to forward to DFB through the `/shm-report` topic, with the format presented in Code snippet 12. As a matter of fact, this information will be used as input in the next component: Ship Stability Toolkit.

```
=====
Received message (shm-report:0:3576)
{'component_id': 'shm', 'timestamp': '2022-11-30T08:41:03Z', 'heave': 0.0,
 'accelerometer_x': -0.10000000149011612, 'accelerometer_y': 0.0,
 'accelerometer_z': -9.800000190734863, 'yaw': 106.0, 'pitch': -
 0.30000001192092896, 'roll': -0.20000000298023224}
=====
```

*Code snippet 12. Ship Health Monitoring Kafka sample (/shm-report)*

Additionally, the Processing node can generate asynchronous alerts in case it detects a potentially dangerous situation, e.g., the ship is listing. In this case, a different message is immediately generated and, using the `/shm-notification` topic. On this particular example, PIMM grasps this notification and display a new message that identifies this as a type of incident that may unleash the “Situation Assessment” status. The format of the object is shown in Code snippet 13.

```
=====
Received message (shm-notification:0:30)
{'timestamp': '2022-11-30T08:42:26Z', 'component_id': 'shm', 'alarm_type':
 'Pitch>2', 'accelerometer_x': 0.0, 'accelerometer_y': 0.0, 'accelerometer_z':
 0.0, 'heave_velocity': 0.0, 'heave_acceleration': 0.0, 'heave_ship_motion': 0.0,
 'yaw': -46.70000076293945, 'pitch': 3.9000000953674316, 'roll': -9.5}
=====
```

*Code snippet 13. Ship Health Monitoring Kafka sample (/shm-notification)*

## 5.7 Ship Stability Toolkit

This component predicts the maximum motions likely to happen according to the current conditions (i.e., ship stability, speed, heading, weather). SST takes the information flows generated through the `/weather-service`, `/ais-position`, `/shm-report` and `/shm-notification` topics (this last one would reflect a hazardous situation).

<sup>17</sup> The motion sensors do not have the capacity to communicates with the PALAEMON platform by themselves. There is a so-called Processing Node (i.e., laptop) that centralizes the data and streams the information via Kafka messages.



But before that, we have created a model that forecasts the ship motions. Based on a 3D version of Elyros' hull, we have used the WAMIT simulation environment [39] to get the hydrodynamic Response-Amplitude-Operators (RAOs) properties. These values act as the input in a Matlab's toolchain to obtain the motion of the ship in different seaways. Due to the computational complexity behind these calculations, we have created a database that contains the pre-calculations. This repository is stored alongside the SST; hence the outputs are on hand and we can get the results with negligible latency.

The way to obtain the predictions is as follows: the SST aggregates the values obtained from the Kafka messages and queries the databases to fetch the closest values to the current conditions. An arbitrary example of the message output (`/stability-toolkit`) can be found in Code snippet 14.

```
=====
Received message (stability-toolkit:0:2927)
{'state': 0, 'timestamp': '2022-11-23T10:42:34.354107+00:00', 'fn': 0, 'Hs': 0,
'L0': 0, 'beta': 0, 'head': 0, 'Surge': 19.6153, 'Sway': 9.536, 'Heave': 5.6295,
'Roll': 15.5877, 'Pitch': 15.9844, 'Yaw': 7.7809}
=====
```

*Code snippet 14. Ship Stability Toolkit Kafka sample (/stability-toolkit)*

It is worth highlighting that this object contains predicted motion values, meaning that there will be different to those of generated by the SHM. Below we briefly describe each of them.

- **State** – Vessel initial state /heeling angle
- **fn** – Vessel speed over ground
- **t** – Wave peak period
- **Hs** – Significant wave height
- **L0** – Wave length
- **beta** – Wave direction
- **head** – Vessel's heading
- **surge** – Expected surge
- **sway** – Expected sway
- **heave** – Expected surge
- **roll** – expected roll
- **pitch** – Expected pitch
- **yaw** – Expected roll

The reader might refer to D3.4 (Ship Stability Toolkit – v2) [18] for a more complete description of the process.

## 5.8 Weather Service

It goes without discussion that weather plays a critical role during an incident and its subsequent evacuation. Alongside light (evacuations at night-time are extremely more complicated to handle than those that happen with direct sunlight), bad weather (e.g., harsh sea, storm, high waves, etc.) is another factor with utmost impact over evacuation procedures.

In PALAEMON we rely on an external source to get trustworthy weather information. Namely, we take the data from National Oceanic and Atmospheric Administration's (NOAA) API, which generates a forecast every three hours. As input, the API needs the location of the ship, which we extract from Kafka's `/ais-position` topic. We include in Figure 38 a screenshot that



embraces the request and response exchanged through a RESTful interface (i.e., HTTPS). We can see that, for the request, we need to include three parameters: timestamp (we have used “latest”, but we could have queried any past timestamp), latitude and longitude. As for the response, the object covers the main weather and sea phenomena.

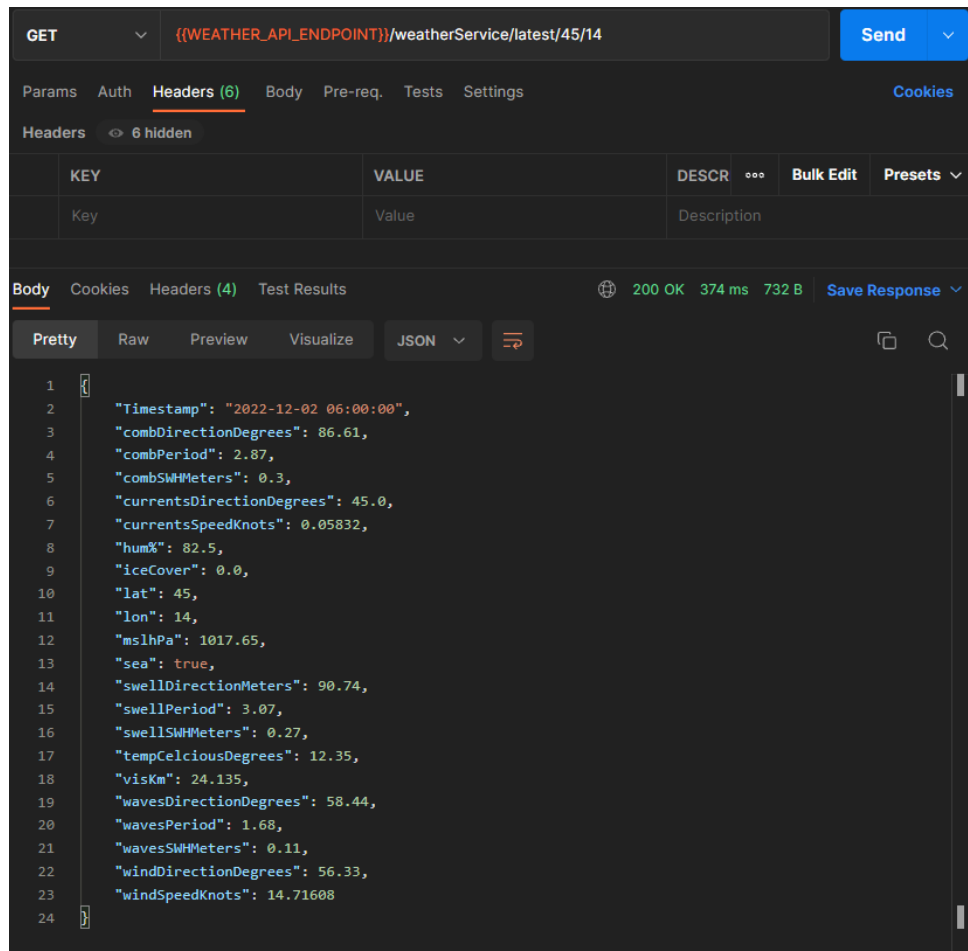


Figure 38. Weather service REST API request and response example

To emulate an end-to-end communication with NOAA’s service (on the Internet), we get the information at PALAEMON’s instance ashore (PALAEMON-02). As of this moment, the following pipeline is executed:

1. (Shore) VDES gateway sends the message to the VDES Transceiver via MQTT. This follows a similar process to that of the Smart Bracelets (HTTP → MQTT). However, this is an internal component task that goes out of the scope of this deliverable.
2. VDES TRX ashore broadcast the weather message via VDE channel. We invite the reader to get acquainted on this radio transmissions by reading D7.3 (Deployment of VDES – v2) [19].
3. (Ship) VDES TRX onboard receives the transmission and goes in the reverse direction: the physical-level message is translated into a JSON object, which is forwarded to the VDES Gateway (onboard) through a MQTT notification. Note: Shore and Ship’s MQTT infrastructures are completely independent and cannot communicate with each other.
4. VDES Gateway gathers the messages and sends to the PALAEMON Platform using the `/weather-service` topic, as illustrated in Code snippet 15.

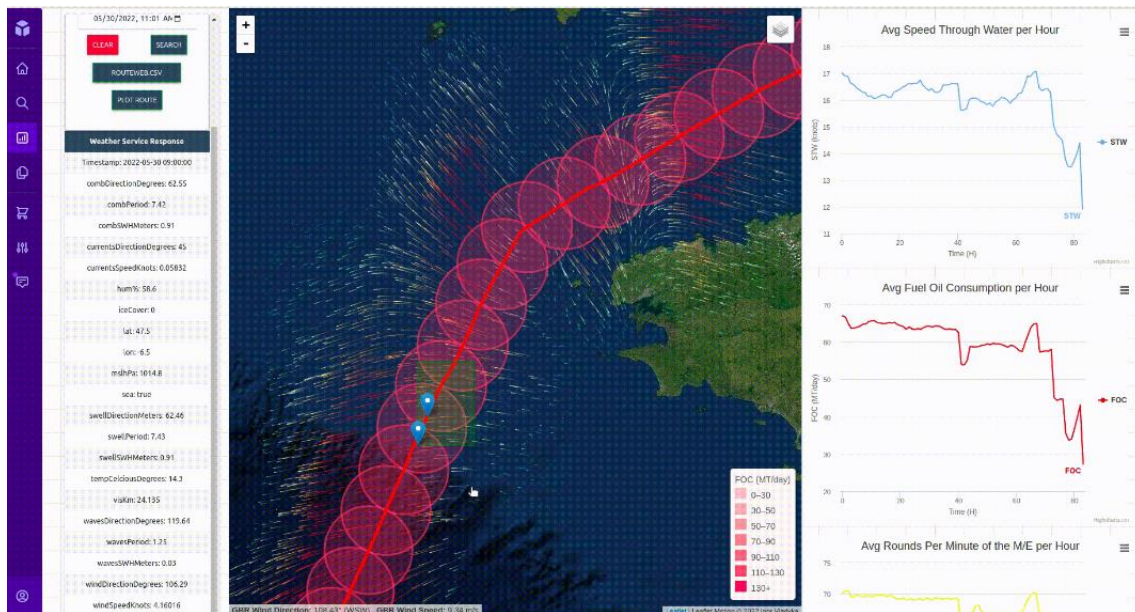
```

=====
Received message (weather-service:0:12783)
{'Timestamp': '2022-12-02 06:00:00', combDirectionDegrees': 86.61, combPeriod': 2.87,
combSWHMeters': 0.3, currentsDirectionDegrees': 45.0, currentsSpeedKnots': 0.05832,
hum%': 82.5, iceCover': 0.0, lat': 45, lon': 14, mslhPa': 1017.65, sea': true,
swellDirectionMeters': 90.74, swellPeriod': 3.07, swellSWHMeters': 0.27,
tempCelciusDegrees': 12.35, visKm': 24.135, wavesDirectionDegrees': 58.44,
wavesPeriod': 1.68, wavesSWHMeters': 0.11, windDirectionDegrees': 56.33,
windSpeedKnots': 14.71608}
=====

```

*Code snippet 15. Weather-service Kafka message sample (/weather-service)*

As a companion application to this weather service, DANAOS has developed a user interface (web page) that adds a visualization layer on top of the raw Weather Forecast Service API. Figure 39 illustrates the user interfaces and gives an idea of the main supported features.



*Figure 39. Weather Service Map user interface sample*

At the left side menu, users can either manually introduce a pair of coordinates (i.e., latitude and longitude) and a timestamp. They receive in turn the information we saw in Code snippet 15; the information is displayed on the left frame (under the inputs). Moreover, the application also allows to introduce a CSV (Comma Separated Values) file with multiple timestamps and coordinates, leading to a multi-query to NOAA's API. The map displays the route followed by the vessel, including some extra features as the wind direction or a circle that represents the valid area of the forecast (the precision of the API is 0.5 degrees). On the right panel, the application can show extra information, in form of time series curves. Some examples are:

- Average Speed through Water per Hour
- Average Fuel Oil Consumption per Hour
- Average Rounds per Minute of the M/E per Hour
- Average CO<sub>2</sub> emissions per Hour
- Average SOX emissions per Hour
- Average NOX emissions per Hour

- Average Wind Speed (m/s) per Hour

Though there is no actual data coming from the PALAEMON platform represented here, the application is ready and represents some metrics that we could extract from other different sources (e.g., shipboard legacy systems); we have included them for illustrative purposes. The reader can interpret this as a glimpse of the visualization material we could include as part of a, for example, a virtual IBS.

## 5.9 Smart Risk Assessment Platform

This component is responsible for analyzing the context during an incident management, increasing the awareness of Master and Bridge Command Team to lead to better and safer decisions. In essence, the SRAP receives the information of almost all sources (it is basically subscribed to all Kafka topics) and performs a dynamic risk analysis based on the aggregation of all this data. As a matter of fact, SRAP relies on Bayesian Networks to make the underlying calculations [40].

As for the different operative phases, SRAP divides the operation into three main modes/models:

1. **Risk Assessment:** this mode of operation is activated at the time the Master switches to the homologous evacuation status. As we can see in Code snippet 16, SRAP outputs a handful of general checks (e.g., vessel status, structural integrity, etc.), ending up in an overall recommendation to the Master of Sounding the General Alarm (or not). As a matter of fact, this final “Situation Assessment” field is the one that is displayed on PIMM’s Decision Support Center (Figure 19 in page 29).

```
=====
Received message (srap:0:790)
{'messageId': '81887a02-dec3-46ff-93fc-805bc66c5e08', 'timestamp': '2022-11-23T12:13:28.12496723', 'sender': 'SRAP', 'SRAP model': 'Situation Assessment', 'Effectiveness of mitigation measures': 'Not effective', 'Passengers proximity to hazards': 'Medium', 'Status of Passive containment': 'Not effective', 'Spreading': 'Not contained', 'Structural Integrity': 'Compromised', 'Stability': 'Sufficient', 'Hull status': 'Safe', 'Ability to communicate': 'Fully operational', 'Critical system status': 'Fully operational', 'Vessel Status': 'Safe', 'Pax vulnerability onboard': 'Moderate', 'Situation Assessment': 'Sound GA'}
```

*Code snippet 16. Smart Risk Assessment Platform Kafka message sample (/srap – Situation assessment)*

2. **Mustering assessment:** once the evacuation has started and passenger have received instruction to move to their respective closest/safest muster station, SRAP proceeds to break down the ship into different areas or geo-fences and evaluate the risk for each of them, as illustrated in Figure 20 (page 30). To simplify the complexity and speed up the reaction from the Master or Bridge Command Team, the “Low/Medium/High” possible values for each zone is displaying a simple and intuitive color code.

```
=====
Received message (srap:0:514)
{'messageId': '8fc2e9fc-c5f8-48ab-8ebd-2e42bc183adb', 'timestamp': '2022-11-
23T12:14:02.896301', 'sender': 'SRAP', 'SRAP model': 'Mustering Assessment',
'Individual status': {'2549': 'Assistance required', '2552': 'Assistance
required', '2553': 'Movement delayed', '2554': 'Assistance required', '2555':
'Movement delayed'}, 'Escape routes': {'Z1D9': 'Open', 'Z2D9': 'Disrupted',
'Z3D9': 'Open', 'Z4D9': 'Open', 'Z1D8': 'Closed', 'Z2D8': 'Closed', 'Z3D8':
'Opened', 'Z4D8': 'Disrupted', 'MSA': 'Opened', 'MSB': 'Closed', 'MSC':
'Opened', 'MSD': 'Opened'}, 'Group performance': {'Z1D9': 'Low', 'Z2D9':
'Medium', 'Z3D9': 'Low', 'Z4D9': 'High', 'Z1D8': 'Low', 'Z2D8': 'Medium', 'Z3D8':
'Low', 'Z4D8': 'High', 'MSA': 'Low', 'MSB': 'Low', 'MSC': 'High', 'MSD': 'Low'},
'Risk of delay': {'Z1D9': 'Low', 'Z2D9': 'Medium', 'Z3D9': 'Low', 'Z4D9':
'High', 'Z1D8': 'Low', 'Z2D8': 'Medium', 'Z3D8': 'Low', 'Z4D8': 'High', 'MSA':
'High', 'MSB': 'Low', 'MSC': 'High', 'MSD': 'Low'}}
=====
```

*Code snippet 17. Smart Risk Assessment Platform Kafka message sample (/srap – Mustering)*

### 3. Pre-abandonment assessment:

In this last phase, SRAP basically determines the urgency level (low/medium/high) for a potential abandonment, trying to support the Master to take this critical decision. Under the hood, the model is conceived to take inputs as the vessel status, in terms of floatability, stability structural integrity... how much safety teams may have contained the incident, etc. Nonetheless, this information belongs to the internal process, yielding a final JSON object in this phase like the one displayed in Code snippet 18.

```
=====
Received message (srap:0:318)
{'messageId': 'dce8ecca-9901-4697-9402-21e2d7ba37ca', 'timestamp': '2022-11-
23T12:17:48.072397', 'sender': 'SRAP', 'SRAP model': 'Pre-Abandonment
Assessment', 'Status': 'Abandon', 'Urgency for abandonment': 'Medium'}
=====
```

*Code snippet 18. Smart Risk Assessment Platform Kafka message sample (/srap – Preabandonment)*



## 6 Conclusions

In this report we have gathered the main results obtained after a thorough testing campaign, carried out over the PALAEMON Communications Platform. The aim behind this verification process is to assess that the performance, reliability and behaviour of the whole system fulfil the expectations (and requirements) pledged in the Grant Agreement.

Alongside the stakeholders' feedback and system requirements that were used to design the system, we created, at the beginning of the project, a methodology for coping with software development and system integration. Under a (to the extent possible) open-source philosophy, we used a private instance of GitLab, one of the most popular VCS platforms to keep track of the source code (and share it with the consortium or the community). Alongside this basic functionality, we harnessed other features, like the issue and branch management to report any bug or improvement found. Moreover, the CI/CD framework allowed an automatic and seamless re-deployment of modified components just after uploading a new version of the code, without any kind of human intervention.

Regarding the actual hardware running the different services, we have 2 different Virtual Machines in the cloud (i.e., PALAEMON-01 and PALAEMON-02) that emulate the infrastructure that shall be installed onboard Elyros and ashore, respectively. Software component will be deployed on these instances using a hybrid approach, with Kubernetes hosting the core services (Data Fusion Bus modules, PALAEMON Evacuation Coordinator, Voyage Report Generator, etc.), whereas those dedicated to properly manage the evacuation process are executed on Docker (Compose). Aside these, we cannot disregard the importance of all the standalone devices and technologies that funnel information to the system. Smart cameras, bracelets, drones, VDES transceivers, AR glasses, etc. bring a new level of hardware that help increase the awareness and safety during an evacuation.

We have followed two different approaches to validate the operation of the PALAEMON Communications platform, which, as the reader shall expect, hinges around the concept of smart evacuation management, concept in which we have delved into, defining its own state machine and communications protocol between the so-called PALAEMON Evacuation Coordinator and the rest of the main software components.

First, we cover a cross-validation model that presents transversal outputs or evidence from various either internal frameworks or even external tools, which do not have a direct participation during an incident. On the former group, Apache Kafka – actual key element of Data Fusion Bus, the PALAEMON Evacuation Coordinator and the Voyage Report Generator, which basically gathers all the information generated around a voyage. Belonging to these “extra tools”, we have presented Kibana as a data visualization framework over the main database (Elasticsearch) and Grafana, another monitoring and observability tool to see the current performance of Kafka's communications.

Aside this holistic demonstration, we also report the individual communication proofs of the main software components. For that, we have chiefly focused on the messages exchanged using Kafka, but there are special cases where the presence of graphical user interfaces foster and validate the correct interplay among components.

With these tests passed, the platform is ready to become an active part during a real evacuation scenario, framed under WP8 (PALAEMON Field Trials, Evaluation and Outcomes) activities.



## 7 References

- [1] "F/B Elyros - ANEK Lines (Official web page)." <https://www.anek.gr/en/vessel/fb-elyros/>.
- [2] "PALAEMON Deliverable D7.4 - Software Development and Integration Methodology," 2020.
- [3] "PALAEMON Deliverable D2.6 - PALAEMON Architecture v1," 2020.
- [4] "Production-Grade Container Orchestration - Kubernetes." <https://kubernetes.io/> (accessed Apr. 20, 2020).
- [5] "Docker. Accelerated, Containerized Application Development (homepage)." <https://www.docker.com/>.
- [6] "Apache Kafka." <https://kafka.apache.org/> (accessed Jun. 02, 2021).
- [7] "GitLab.org / GitLab · GitLab." <https://gitlab.com/gitlab-org/gitlab>.
- [8] "Sonatype Nexus Repository Manager (homepage)." <https://www.sonatype.com/products/nexus-repository>.
- [9] "PALAEMON Deliverable D2.7 - PALAEMON Architecture v2," 2021.
- [10] "PALAEMON Deliverable D7.5 - System Integration & Final PALAEMON Prototype (v2)," 2022.
- [11] "Overview of Docker Compose | Docker Documentation." <https://docs.docker.com/compose/>.
- [12] "Lens - The Kubernetes IDE (homepage)." <https://k8slens.dev/>.
- [13] "Kapow! GitHub repository." <https://github.com/BBVA/kapow>.
- [14] "Apache NiFi." <https://nifi.apache.org/> (accessed Jun. 02, 2021).
- [15] "Apache NiFi Registry (homepage)." <https://nifi.apache.org/registry.html>.
- [16] "Apache Zookeeper (homepage)." <https://zookeeper.apache.org/>.
- [17] "Keycloak - Open Source Identity and Access Management for Modern Applications and Services." <https://www.keycloak.org/>.
- [18] "PALAEMON Deliverable D3.4 - Development of Ship Stability Toolkit v2," 2021.
- [19] "PALAEMON Deliverable D7.3 - Deployment of VDES (v2)," 2022.
- [20] "PALAEMON Deliverable D3.10 - Development of Risk Assessment Platform v2," 2021.
- [21] "PALAEMON Deliverable D6.5 - PALAEMON Incident Management Module (PIMM)," 2022.
- [22] "MinIO | High Performance, Kubernetes Native Object Storage." <https://min.io/> (accessed Jun. 02, 2021).
- [23] "Traefik - The Cloud Native Application Proxy (homepage)." <https://traefik.io/traefik/>.
- [24] P. Consortium, "PALAEMON D7.5 PALAEMON Uniform Data Exchange Modules - Interoperability Layer," 2020.
- [25] "PALAEMON Deliverable D7.2 - Uniform Data Exchange Modules - Interoperability Layer (v2)," 2022.
- [26] "Elasticsearch: The Official Distributed Search & Analytics Engine | Elastic." <https://www.elastic.co/elasticsearch/> (accessed Jun. 18, 2021).
- [27] "Kibana: Explore, Visualize and Discover Data (homepage)." <https://www.elastic.co/kibana/>.



- [28] “Kibana Canvas: Create Live Infographic-Style Presentations,” [Online]. Available: <https://www.elastic.co/what-is/kibana-canvas>.
- [29] GDPR, “General Data Protection Regulation (GDPR) – Official Legal Text,” *General Data Protection Regulation*, 2016. <https://gdpr-info.eu/>.
- [30] European Commission, “General Data Protection Regulation - Recital 46: Vital Interests of the Data Subject,” 2018. [Online]. Available: <https://gdpr-info.eu/recitals/no-46/>.
- [31] F. C. C. (FCC), “Global Maritime Distress and Safety System (GMDSS) - Ship radio stations,” [Online]. Available: <https://www.fcc.gov/wireless/bureau-divisions/mobility-division/maritime-mobile/ship-radio-stations/global-maritime>.
- [32] A. L. Panagiotis Panagiotidis, Kyriakos Giannakis, Nikolaos Angelopoulos, “A MARINE ACCIDENTS DATASET,” 2021. doi: 10.5281/zenodo.5592999.
- [33] “The International Safety Management (ISM) Code.” <https://www.imo.org/en/OurWork/HumanElement/Pages/ISMCode.aspx> (accessed Jun. 21, 2021).
- [34] “PALAEMON Deliverable D6.3 - PALAEMON Interfaces and HMI toolkit,” 2022.
- [35] “PALAEMON Deliverable D6.4 - Development of PALAEMON On-board Decision Support System,” 2022.
- [36] “Grafana - The open observability platform (homepage).” <https://grafana.com/>.
- [37] “Elasticsearch Dump (GitHub repository).” <https://github.com/elasticsearch-dump/elasticsearch-dump>.
- [38] “Softblade - MQTT.fx (homepage).” <https://softblade.de/en/mqtt-fx/>.
- [39] I. WAMIT, “WAMIT Simulator (homepage).” <https://www.wamit.com/index.htm>.
- [40] A. G. Eleye-Datubo, A. Wall, A. Saajedi, and J. Wang, “Enabling a Powerful Marine and Offshore Decision-Support Solution Through Bayesian Network Technique,” *Risk Anal.*, vol. 26, no. 3, pp. 695–721, Jun. 2006, doi: 10.1111/j.1539-6924.2006.00775.x.
- [41] “Kubernetes Pod definition and documentation.” <https://kubernetes.io/docs/concepts/workloads/pods/>.

## Annex I CI/CD Sample

Code snippet 19. Platform Deployment repository .gitlab-ci.yml file (CI/CD example)

```
image: docker:stable

variables:
  DOCKER_HOST: tcp://localhost:2375
  DOCKER_TLS_CERTDIR: ""
  VDR_IMAGE: voyage-report-generator
  KAPOW_IMAGE: kapow

services:
  - docker:stable-dind

before_script:
  - echo "===== Branch $CI_BUILD_REF_NAME ====="
  - echo $CI_BUILD_REF_NAME
stages:
  - build
  - build_master
  - image_clean
  - deploy

##### BUILD STEP #####
.build_template: &build_definition
  image:
    name: gcr.io/kaniko-project/executor:perf-debug
    entrypoint: [""]
  before_script:
    - mkdir -p /kaniko/.docker
    - echo "{\"auths\":{\"${NEXUS_CLI_HOST}\":{\"auth\":\"$(printf \"%s:%s\" \"${NEXUS_USER}\" \"${NEXUS_PASS}\" | base64 | tr -d '\\n')\"}}}" > /kaniko/.docker/config.json
  script:
    - >-
      /kaniko/executor
      --cache=true
      --snapshotMode=redo
      --context="${CONTEXT}"
      --dockerfile="${CONTEXT}/Dockerfile"
      ${DOCKER_EXTRA_ARGS}
      --destination="${NEXUS_CLI_HOST}/${DOCKER_IMAGE}:${CI_COMMIT_SHA}"
      --destination="${NEXUS_CLI_HOST}/${DOCKER_IMAGE}:latest"

build_kapow_job:
  <<: *build_definition # Merge the contents of the 'deploy_definition'
alias
stage: build
variables:
  CONTEXT: "./kapow"
  DOCKER_IMAGE: ${KAPOW_IMAGE}
only:
  changes:
    - kapow/**/*

build_vdr_job:
  <<: *build_definition # Merge the contents of the 'deploy_definition'
alias
stage: build
variables:
  CONTEXT: "./vdr"
  DOCKER_IMAGE: ${VDR_IMAGE}
only:
  changes:
```

```

- vdr/**/*

##### BUILD MASTER STEP #####
.build_master_template: &build_master_definition
  image:
    name: gcr.io/kaniko-project/executor:perf-debug
    entrypoint: [""]
  before_script:
    - mkdir -p /kaniko/.docker
    - echo "{\"auths\":{\"${NEXUS_CLI_HOST}\":{\"auth\":\"$(printf \"%s:%s\"
    \"${NEXUS_USER}\" \"${NEXUS_PASS}\" | base64 | tr -d '\\n')\"}}}" >
    /kaniko/.docker/config.json
  script:
    - echo "FROM ${NEXUS_CLI_HOST}/${DOCKER_IMAGE}:${CI_COMMIT_SHA}" | /kaniko/executor
    --context="${CONTEXT}" --dockerfile=/dev/stdin --
    destination="${NEXUS_CLI_HOST}/${DOCKER_IMAGE}:stable"

build_master_kapow_job:
  <<: *build_master_definition          # Merge the contents of the
  'deploy_definition' alias
  stage: build_master
  variables:
    CONTEXT: "./kapow"
    DOCKER_IMAGE: "${KAPOW_IMAGE}"
  only:
  refs:
    - master
  changes:
    - kapow/**/*

build_master_vdr_job:
  <<: *build_master_definition          # Merge the contents of the
  'deploy_definition' alias
  stage: build_master
  variables:
    CONTEXT: "./vdr"
    DOCKER_IMAGE: "${VDR_IMAGE}"
  only:
  refs:
    - master
  changes:
    - vdr/**/*

##### CLEAN TEMPORARY IMAGES STEP #####

.clean_temporary_images_template: &clean_temporary_images_definition # Hidden key that
defines an anchor named 'deploy_clean_definition'
  image: debian:bullseye-slim
  variables:
    IMAGE: ""
  before_script:
    - apt-get update && apt install -y python3 python3-dev python3-pip gcc musl-dev
    libffi-dev curl cargo
    - curl https://sh.rustup.rs -sSf | sh -s -- -y
    - source $HOME/.cargo/env
    - pip3 install nexus3-cli==3.0.0
    - nexus3 login --url $NEXUS_WEB_HOST --username $NEXUS_USER --password $NEXUS_PASS -
    -x509_verify
  script:
    - echo "Cleaning $NEXUS_REPO $IMAGE"
    - nexus3 delete $NEXUS_REPO/v2/$IMAGE/manifests/$CI_COMMIT_SHA
  when: always

clean_vdr_image_job:
  <<: *clean_temporary_images_definition

```



```
stage: image_clean
variables:
  IMAGE: $VDR_IMAGE
only:
  changes:
    - vdr/**/*

clean_kapow_image_job:
  <<: *clean_temporary_images_definition
  stage: image_clean
  variables:
    IMAGE: $KAPOW_IMAGE
  only:
    changes:
      - kapow/**/*

##### TRIGGER PIPELINE #####
trigger_pipeline_job:
  stage: deploy
  script:
    - apk update && apk add curl
    - "curl -X POST --fail -F token=$TRIGGER_TOKEN -F ref=master -F
variables[CI_COMMIT_SHA]={CI_COMMIT_SHA}
https://scm.atosresearch.eu/api/v4/projects/758/trigger/pipeline"
  only:
    refs:
      - master
  changes:
    - kapow/**/*
    - vdr/**/*
```

## Annex II Evacuation Coordinator message exchange with dependent components

```

=====
Received message (resource-discovery-request:0:119)
{'originator': 'Evacuation Coordinator', 'timestamp': '2022-11-23T11:32:37.640964Z',
 'evacuation-status': 0}
=====
Received message (resource-discovery-response:0:553)
{'component_id': 'SSS', 'operation_mode': 0, 'timestamp': '2022-11-
23T11:32:37.914780+00:00'}
=====
Received message (resource-discovery-response:0:554)
{'component_id': 'SST', 'operation_mode': 0, 'timestamp': '2022-11-
23T11:32:34.824838+00:00'}
=====
Received message (resource-discovery-response:0:555)
{'timestamp': '2022-11-23 11:32:17.359', 'component_id': 'PaMEAS-Location',
 'operation_mode': 1}
=====
Received message (resource-discovery-response:0:556)
{'component_id': 'VDES App', 'operation_mode': 1, 'timestamp': '2022-11-
23T11:32:37.954067301Z'}
=====
Received message (resource-discovery-response:0:557)
{'component_id': 'PIMM+DSS+WFT', 'operation_mode': 1, 'timestamp': '2022-11-
23T11:32:38.123562'}
=====
Received message (resource-discovery-response:0:558)
{'component_id': 'camera-02', 'operation_mode': 3, 'timestamp': '2022-11-
23T12:32:37.962965'}
=====
Received message (resource-discovery-response:0:559)
{'component_id': 'camera-01', 'operation_mode': 3, 'timestamp': '2022-11-
23T12:32:37.967202'}
=====
Received message (resource-discovery-response:0:560)
{'component_id': 'camera-04', 'operation_mode': 3, 'timestamp': '2022-11-
23T12:32:37.967381'}
=====
Received message (resource-discovery-response:0:561)
{'component_id': 'camera-03', 'operation_mode': 3, 'timestamp': '2022-11-
23T12:32:37.967423'}
=====
Received message (resource-discovery-response:0:562)
{'timestamp': '2022-11-23T11:32:40.515490', 'component_id': 'VDR', 'operation_mode': 0}
=====

```

*Code snippet 20. Evacuation Coordinator - /resource-discovery-request and /resource-discovery-response (sample)*



```

=====
Received message (heartbeat-request:0:85248)
{'originator': 'Evacuation Coordinator', 'timestamp': '2022-11-23T11:15:56.310966Z',
'evacuation-status': 3}
=====
Received message (heartbeat-response:0:378077)
{'timestamp': '2022-11-23T11:15:56.565955', 'component_id': 'VDR', 'operation_mode': 1}
=====
Received message (heartbeat-response:0:378078)
{'component_id': 'VDES App', 'operation_mode': 1, 'timestamp': '2022-11-
23T11:15:56.582800976Z'}
=====
Received message (heartbeat-response:0:378079)
{'component_id': 'SST', 'operation_mode': 3, 'timestamp': '2022-11-
23T11:15:53.481089+00:00'}
=====
Received message (heartbeat-response:0:378080)
{'timestamp': '2022-11-23 11:15:35.984', 'component_id': 'PaMEAS-Location',
'operation_mode': 1}
=====
Received message (heartbeat-response:0:378081)
{'component_id': 'PIMM+DSS+WFT', 'operation_mode': 1, 'timestamp': '2022-11-
23T11:15:56.758547'}
=====
Received message (heartbeat-response:0:378082)
{'component_id': 'camera-03', 'operation_mode': 3, 'timestamp': '2022-11-
23T12:15:56.593514'}
=====
Received message (heartbeat-response:0:378083)
{'component_id': 'camera-01', 'operation_mode': 3, 'timestamp': '2022-11-
23T12:15:56.593794'}
=====
Received message (heartbeat-response:0:378084)
{'component_id': 'camera-02', 'operation_mode': 3, 'timestamp': '2022-11-
23T12:15:56.602854'}
=====
Received message (heartbeat-response:0:378085)
{'component_id': 'camera-04', 'operation_mode': 3, 'timestamp': '2022-11-
23T12:15:56.609201'}
=====

```

*Code snippet 21. Evacuation Coordinator - /heartbeat-request and /heartbeat-response (sample)*

```

# Evacuation Component Status
=====
Received message (evacuation-coordinator:0:3750)
{'originator': 'Evacuation Coordinator', 'timestamp': '2022-11-23T11:13:37.851704Z',
 'evacuation-status': 3}
=====
Received message (evacuation-component-status:0:10927)
{'timestamp': '2022-11-23T11:13:38.102700', 'component_id': 'VDR', 'operation_mode': 1,
 'operation_info': 'description'}
=====
Received message (evacuation-component-status:0:10928)
{'component_id': 'VDES App', 'operation_mode': 1, 'timestamp': '2022-11-
23T11:13:38.114200341Z'}
=====
Received message (evacuation-component-status:0:10929)
{'component_id': 'SST', 'operation_mode': 3, 'timestamp': '2022-11-
23T11:13:35.024860+00:00'}
=====
Received message (evacuation-component-status:0:10930)
{'component_id': 'camera-04', 'operation_mode': 3, 'timestamp': '2022-11-
23T12:13:38.125351'}
=====
Received message (evacuation-component-status:0:10931)
{'component_id': 'camera-03', 'operation_mode': 3, 'timestamp': '2022-11-
23T12:13:38.134947'}
=====
Received message (evacuation-component-status:0:10932)
{'component_id': 'camera-02', 'operation_mode': 3, 'timestamp': '2022-11-
23T12:13:38.124869'}
=====
Received message (evacuation-component-status:0:10933)
{'component_id': 'camera-01', 'operation_mode': 3, 'timestamp': '2022-11-
23T12:13:38.156348'}
=====

```

Code snippet 22. Evacuation Coordinator - /evacuation-coordinator and /evacuation-component-status (sample)

### Annex III List of Kafka Topics

Table 1. Kafka Topics compilation table

Topic	Producer	Target consumer(s)	Elasticsearch	Description
/evacuation-coordinator	Evacuation Coordinator	ALL	evacuation-coordinator	Ship evacuation status (0 – Normal status... 5- MEV launching)
/evacuation-component-status	ALL	Evacuation coordinator	evacuation-component-status	Components reply to evacuation status change – may modify their internal operation mode
/resource-discovery-request	Evacuation Coordinator	ALL	resource-discovery-request	At voyage start, the System wants to know all available and connected devices
/resource-discovery-response	ALL	Evacuation coordinator	resource-discovery-response	Initialization message request (to know all active devices)
/heartbeat-request	Evacuation Coordinator	ALL	heartbeat-request	Periodic check of components health (e.g., every minute). Sent by the coordinator
/heartbeat-response	ALL	Evacuation coordinator	heartbeat-response	Periodic check of components health (e.g., every minute). Reply generated by the components
/smart-safety-system	SSS	DFB	smart-safety-system	Transcription of Drag'n Drop events created on the SSS dashboard
/smart-camera	ADV	DFB	smart-camera	Message sent when the cameras detect a variation in the number of people detected
/smart-camera-alarm	ADV	DFB	smart-camera-alarm	People trapped, blocked corridor, stampede detected, etc.
/shm-report	ESI	DFB	shm-report	Ship health monitoring stability report (sent every 20 seconds in the default configuration).

/shm-notification	ESI	DFB	shm-alarm	Alarm triggered when the sensors data have passed a threshold (e.g., listing situation detected).
/smart-bracelet-event-notification	ADV	DFB	smart-bracelet-event-notification	Asynchronous event generated by the bracelets when e.g., a passenger falls down or pushes the alarm button
/smart-bracelet-sensor-data	ADV	DFB	smart-bracelet-sensor-data	Biomedical information of the bracelet wearer: heartbeat, oxygen (O <sup>2</sup> ) saturation, body temperature, accelerometer information (pitch, roll, angle)
/smart-bracelet-pameas-evac-msg	UAEG	DFB	smart-bracelet-pameas-evac-msg	Asynchronous msg from event generated by PaMEAS to be displayed on the smart bracelets' screens
/decision-support-system	KT	DFB	decision-support-system	List of actions taken from SOLAS and ISM manuals, according to the evacuation status and the stakeholder (Master, crew, etc.)
/mayday-message	ATOS, THALES	DFB	mayday-message	Distress signal to be transmitted via VDES wireless radio link
/srap	NTUA	DSS	srap	Output of the Smart Risk Assessment Platform
/pameas-location	UAEG	SRAP	<a href="#">pameas-location</a>	Anonymized location data of a passenger
/pameas-notification	UEAG	SRAP	<a href="#">pameas-notification</a>	Instructions exchanged between PaMEAS and other components (PIMM, crew, etc.)
/pameas-person	UAEG	DFB	<a href="#">pameas-person</a>	Record of person (passenger/crew member) registered to the system
/weather-service	WISER	Weather API	<a href="#">weather-api</a>	Weather forecast generated every 3 hours. Data captured and aggregated from an external service
/ais-position	ATOS	Dummy position notification (AIS-like)	<a href="#">ais-position</a>	Overheard information from the ship's Automatic Identification System (AIS)

/stability-toolkit	SST	DFB	<a href="#">stability-toolkit</a>	Ship motion predictions
/sms-report	ATOS	DANAOS	N/A	When the Voyager Data Report is ready, the component sends a message to SMS sharing the endpoint (location) of the report on DFB's cloud repository (i.e., MinIO)
/voyage-report-generator-request	ATOS	ATOS	voyage-report-generator-request	Activate/deactivate report. Notification from Evacuation Coordinator to VDR
/voyage-report-generator-response	ATOS	ATOS	voyage-report-generator-response	Acknowledgment
/legacy	UAEG	KT	legacy	Emulation of Shipboard Legacy Systems
/smoke-detector	UAEG	KT	smoke-detector	Emulation of Smoke Detector Alarms

## Annex IV PIMM API Assessment

This annex contains, as example, some of the services supported by the PIMM. They return data generated and collected in the PALAEMON System, either from Kafka or Elasticsearch. In order not to expose sensitive information (e.g., endpoints, passwords, tokens, etc.), these values will be substituted and will be displayed embraced between double curly brackets (i.e., `{{value}}`).

Table 2. PIMM Get Token Request

HTTP message type	POST	
Header	Content-Type	application/json
Body (only for POST messages)	<pre>{   "username": {{pimmuser}},   "password": {{pimm-password}} }</pre>	
Request (URL)	<code>{{pimm-host}}/token</code>	
Response	<pre>{   "refresh": {{refresh-token}}   "access": {{token}} }</pre>	

Table 3. PIMM Get Evacuation status

HTTP message type	GET	
Header	Authorization	Bearer <code>{{token}}</code>
Request (URL)	<code>{{pimm-host}}/evac_coordinator</code>	
Response	<pre>{   "level": 5 }</pre>	

Table 4. PIMM Get Voyage Status

HTTP message type	GET	
Header	Authorization	Bearer <code>{{token}}</code>
Request (URL)	<code>{{pimm-host}}/voyage_status</code>	
Response	<pre>{   "voyage_status": 0 }</pre>	

Table 5. PIMM Get Fire Sensor data

HTTP message type	GET	
Header	Authorization	Bearer <code>{{token}}</code>
Request (URL)	<code>{{pimm-host}}/fire-sensor</code>	



Response	<pre>[   {     "id": 1,     "label": "Fire Detector 1 on Garage",     "slug": "Fire Detector 1 on Garage",     "location": "Garage",     "triggered": false   },   {     "id": 2,     "label": "Fire Detector 2 on Garage",     "slug": "Fire Detector 2 on Garage",     "location": "Garage",     "triggered": false   },   {     "id": 3,     "label": "Fire Detector 3 on Garage",     "slug": "Fire Detector 3 on Garage",     "location": "Garage",     "triggered": false   },   {     "id": 4,     "label": "Smoke Detector 1 on Muster Station D",     "slug": "Smoke Detector 1 on Muster Station D",     "location": "Station D",     "triggered": false   },   {     "id": 5,     "label": "Smoke Detector 2 on Muster Station D",     "slug": "Smoke Detector 2 on Muster Station D",     "location": "Station D",     "triggered": false   } ]</pre>
----------	---

Table 6. PIMM Get Weather Forecast Toolkit

HTTP message type	GET	
Header	Authoriz ation	Bearer <b>{{token}}</b>
Request (URL)	<b>{{pimm-host}}</b> /wft	
Response	<pre>[   {     "id": 1,     "significant_wave_height": "1.00",     "wind_direction": "169.00",     "wind_speed": "4.00",     "air_temp": "5.00",     "timestamp": "2022-12-07T15:54:02.487259Z",     "action_plan_follow": "{\\"actions\\": [\\"The crew's actions to extinguish of the fire were effective\\", \\"Similarity\\": \\"100%\\", \\"Similar_Case\\": \\"Gunde Maersk\\""]}",     "action_plan_not_follow": "{\\"actions\\": [\\"The fire was spreading quickly\\", \\"Similarity\\": \\"99%\\", \\"Similar_Case\\": \\"Aframax River\\""]}",     "wind_charts": "{\\"chart\\": {\\"type\\": \\"bar\\"}, \\"title\\": {\\"text\\": \\"Distribution of deaths and injuries regarding the wind speed conditions\\"}, \\"xAxis\\": {\\"categories\\": \\"(0,34"</pre>	

```

] Knots\", \"title\": {\"text\": \"null\"}}, \"yAxis\": {\"min\"
: 0, \"title\": {\"text\": \"Frequency\", \"align\": \"high\"},
\"labels\": {\"overflow\": \"justify\"}}, \"tooltip\": {\"valueS
uffix\": \"\"}, \"plotOptions\": {\"bar\": {\"dataLabels\": {\"e
nabled\": \"true\"}}}, \"legend\": {\"layout\": \"vertical\", \"
align\": \"right\", \"verticalAlign\": \"top\", \"x\": -
40, \"y\": 80, \"floating\": \"true\", \"borderWidth\": 1, \"bac
groundColor\": {\"Highcharts.defaultOptions.legend.backgroundCo
lor\": \"#FFFFFF\", \"shadow\": \"true\"}}, \"credits\": {\"enab
led\": \"false\"}, \"series\": [{\"name\": \"Deaths\", \"data\":
93}, {\"name\": \"Injuries\", \"data\": 572}]]\",
  \"visibility_charts\": \"{\\\"chart\\\": {\\\"type\\\": \"bar\\\", \\
\"title\\\": {\\\"text\\\": \"Distribution of deaths and injuries regar
ding the visibility conditions\\\", \\\"xAxis\\\": {\\\"categories\\\": \\
\">5 Miles\\\", \\\"title\\\": {\\\"text\\\": \"null\\\"}}, \\\"yAxis\\\": {\\\"min
\\\": 0, \\\"title\\\": {\\\"text\\\": \"Frequency\\\", \\\"align\\\": \"high\\\"}
, \\\"labels\\\": {\\\"overflow\\\": \"justify\\\"}}, \\\"tooltip\\\": {\\\"valu
eSuffix\\\": \"\"}, \\\"plotOptions\\\": {\\\"bar\\\": {\\\"dataLabels\\\": {\\
\"enabled\\\": \"true\\\"}}}, \\\"legend\\\": {\\\"layout\\\": \"vertical\\\",
\\\"align\\\": \"right\\\", \\\"verticalAlign\\\": \"top\\\", \\\"x\\\": -
40, \\\"y\\\": 80, \\\"floating\\\": \"true\\\", \\\"borderWidth\\\": 1, \\\"bac
groundColor\\\": {\\\"Highcharts.defaultOptions.legend.backgroundCo
lor\\\": \"#FFFFFF\", \\\"shadow\\\": \"true\\\"}}, \\\"credits\\\": {\\\"enab
led\\\": \"false\\\"}, \\\"series\\\": [{\\\"name\\\": \"Deaths\\\", \\\"data\\\":
43}, {\\\"name\\\": \"Injuries\\\", \\\"data\\\": 175}]]\"
  }
]

```

## Annex V VDR PDF Report Sample (draft)

*It is worth highlighting that the implementation of this component is a Proof-of-Concepts and only poses the possibilities that we could explore in further iterations (after PALAEMON's lifetime).*

In this annex we have included some of the main representations covered in the PDF version of the Voyage Report. This document complements the raw data and multimedia files (i.e., .json and .avi) and graphically displays all the “illustrative” information.

### SHIP PARTICULARS

Vessel ID	Elyros
Flag	Greece
Classification Society	N/A
IMO Number	9178599
Ship Type	Passenger/Ro-Ro Cargo Ship
Owner	ANEK Lines SA
Year of Construction	1998
Length Overall	192
Gross Tonnage	33635
Departure Port	Piraeus
Arrival Port	Souda

### VOYAGE PARTICULARS

Port of departure	Piraeus
Port of arrival	Souda
Type of voyage	Coastal
Cargo information	In ballast
Manning	6
Number of passengers	1200

Figure 40. Voyage Report Example - Ship & Voyage Particulars

### Passenger information

ID	name	surname	dateOfBirth	gender
5oPyeN7bdRMLYVb	Oleta	Yost	61	female
4q4Vx2nWns9DN5k	Alycia	Howe	4	female
6yvD8s2xZyZUEpE	Brendon	Bashirian	38	male
3i8fAKX8Ads8SeL	Aurore	Simonis	24	female
57zzGm6GPerx668	Rae	Pollich	35	female

Figure 41. Voyage Report Example - Passenger list

### EVACUATION AND ALARM INFORMATION

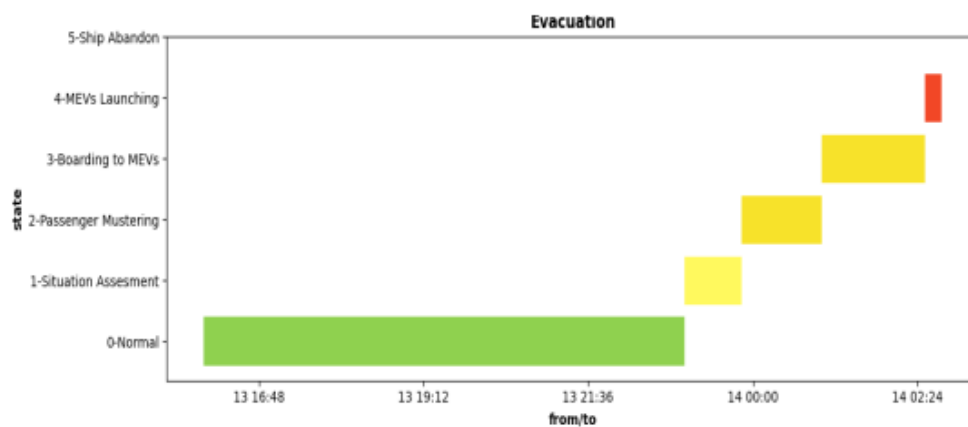


Figure 42. Voyage Report Example - Ship Evacuation Status Timeline

### TRAJECTORY INFORMATION

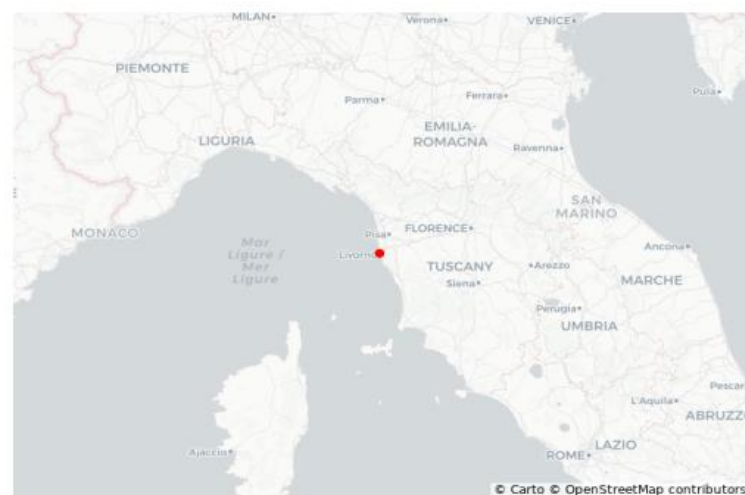


Figure 43. Voyage Report Example - Ship position & trajectory



Figure 44. Voyage Report Example - Smart Cameras alarms timeline

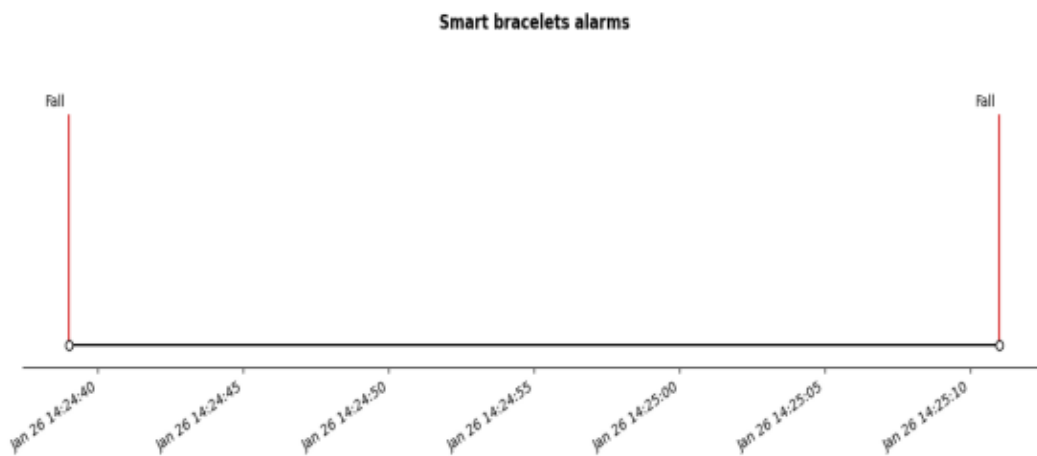


Figure 45. Voyage Report Example - Smart Bracelets alarms (i.e., fall detection) timeline



Figure 46. Voyage Report Example - Ship Health Monitoring alarms timeline

## Annex VI Smart Bracelets Evacuation Support Messages



a) Biometrics data



b) General Alarm



c) Go to Muster Station A



d) Go Downstairs



e) Go straight ahead



f) Turn left



g) Stay there (at muster station)



h) Embark to lifeboat (MEV)

Figure 47. Smart Bracelets Evacuation Support Messages