



beAWARE

Enhancing decision support and management services in extreme weather
climate events

700475

D7.3

Integrated operational beAWARE platform

Dissemination level:	Public
Contractual date of delivery:	Month 12, 30 December 2017
Actual date of delivery:	Month 12, 22 December 2017
Workpackage:	WP7: System development, integration and evaluation
Task:	T7.2 - beAWARE platform tools integration
Type:	Report
Approval Status:	Final Draft
Version:	0.5
Number of pages:	37
Filename:	D7.3_beAWARE_Integrated operational beAWARE platform_2017-12-22_v0.5

Abstract

The main goal of this document is to describe the implementation of each component of the beAWARE platform and the integration stage of the operational platform as a whole. Additionally, a first simple operational scenario that has been implemented is also being described, in order to demonstrate the interoperability of the platform.

The information in this document reflects only the author's views and the European Community is not liable for any use that may be made of the information contained therein. The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information



This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

at its sole risk and liability.



Co-funded by the European Union



This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700475

History

Version	Date	Reason	Revised by
V0.1	01.12.2017	Initial version and assignments distribution	CERTH
V0.2	04.12. 2017	Incorporate information from all partners	All partners
V0.3 – 0.4	06.12.2017	Internal review	CERTH
V0.5	22.12.2017	Final Version	CERTH

Author list

Organisation	Name	Contact Information
IBM	Benny Mandler	MANDLER@il.ibm.com
CERTH	Anastasios Karakostas	akarakos@iti.gr
CERTH	Stefanos Vrochidis	stefanos@iti.gr
CERTH	Emmanouil Michail	michem@iti.gr
CERTH	Kostas Avgerinakis	koafgeri@iti.gr
UPF	Stamatia Dasiopoulou	stamatia.dasiopoulou@upf.edu
MSIL	Yaniv Mordechai	yaniv.mordecai@motorolasolutions.com
IOSB	Moßgraber Jürgen	juergen.mossgraber@iosb.fraunhofer.de
IOSB	Hylke van der Schaaf	hylke.vanderschaaf@iosb.fraunhofer.de
IOSB	Philipp Hertweck	philipp.hertweck@iosb.fraunhofer.de

Executive Summary

This deliverable presents the implementation of the operational prototype of the integrated beAWARE platform. The development status of each component is being presented here along with the integration approach and infrastructure that was used towards the implementation of the platform. The deliverable also presents the first simple operational use case scenario that has successfully been implemented in order to test the interoperability of the different integrated modules so far. Additionally, a logger demonstrator interface has been developed in order to monitor and demonstrate the interoperability of the platform and the messages that are being exchanged between components, according to this use case. Functional results and screenshots are presented in this deliverable.

Abbreviations and Acronyms

API	Application Programming Interface
ASR	Automatic Speech Recognition
CI	Continuous Integration
DTr	dynamic texture recognition
DTstL	Dynamic Texture spatio-temporal localization
GUI	Graphical User Interface
IoT	Internet of Things
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
K8s	Kubernetes
KB	Knowledge Base beAWARE component
M2M	Machine-to-machine
ObjD	object detection
OGC	The Open Geospatial Consortium
PaaS	Platform as a Service
PSAP	Public-safety answering point
SOA	Service-oriented architecture
STL	Spatio-Temporal localization
TLc	Traffic Level classification
UC	Use Case
WP	Work Package
MS	Milestone

Glossary

BlueMix – a public cloud hosting platform provided by IBM

Cloud Foundry – an open and extensible Platform as a Service offering

Consumable - Designing an easy to use product

Mongo DB – Scalable document oriented NoSQL data store

ElasticSearch - high-performance indexing and search system that can be integrated with the CouchBase scalable data back-end.

External applications – Independent applications that are developed and hosted outside the platform but internally make use of beAWARE components.

JSON - (JavaScript Object Notation) is a lightweight data-interchange format often used in web applications

Kafka – A messaging middleware mostly used to connect between different components

Kubernetes – Containers orchestration service: an open-source system for automating deployment, scaling, and managing containerized applications.

OASIS – standardization body: advancing open standards for the information society

OWL - The Web Ontology Language is a family of knowledge representation languages for authoring ontologies

Platform-as-a-Service - a cloud computing concept which offers the developer and deployer of cloud based applications the infrastructure, both HW and middleware, needed for creating and deploying successfully such applications on a cloud environment.

PSAP – Command and control centre to serve authorities, first responders, and citizens, mainly during crisis situations

RDF - A standard model for data interchange on the Web. Used in beAWARE for storing entities description in a searchable manner.

RESTful service – A web service adhering to the REST protocol for information exchange among services, specifying the operations and associated data payload.

SPARQL – an RDF query language.

WebSockets – Communication protocol providing full-duplex communications channels over a single TCP connection; used for bi-directional communication between the beAWARE platform and applications.

Table of Contents

1	INTRODUCTION.....	10
2	PROTOTYPE ARCHITECTURE	11
2.1	Global View.....	11
2.1.1	Architecture Layers.....	12
2.2	Components.....	14
2.2.1	Ingestion Layer	14
2.2.2	Business Layer	14
2.3	Internal Services.....	15
2.3.1	Communication Bus.....	15
2.3.2	Data management	15
2.4	External layer	15
2.4.1	End-users applications.....	15
3	INTEGRATION APPROACH	16
4	COMPONENTS AND INTEGRATION STATUS.....	17
5	INFRASTRUCTURE AND CODE ORGANIZATION.....	22
6	DEMONSTRATOR URLS AND INFORMATION	27
7	SUMMARY AND CONCLUSIONS	37

List of Figures

Figure 1: Architectural high-level view	12
Figure 2: beAWARE Kubernetes cluster	22
Figure 3: beAWARE cluster overview.....	22
Figure 4: Kubernetes cluster worker nodes	22
Figure 5: MessageHub cloud service.....	23
Figure 6: CI workflow.....	24
Figure 7: CI detailed description	24
Figure 8: beAWARE Kubernetes cluster	25
Figure 9: Cluster deployments	26
Figure 10: The beAWARE Logger Environment.....	27
Figure 11: Operational Prototype Demonstration: The triggers simulator’s drop-down menu that creates the incident messages.	28
Figure 12: Operational Prototype Demonstration: The structure of the new incident message (right screen) during the first phase (initial incident report).	29

Figure 13: Operational Prototype Demonstration: The messages that are sent to KB and consequently to PSAP (central screen) during the first phase (initial incident report).	29
Figure 14: Operational Prototype Demonstration: The updated incident report containing an image as an attachment, as can be seen on the right screen, during the second phase (image upload).	30
Figure 15: The messages (central screen) that are sent to KB, PSAP and Image Analysis module after the upload of an image, along with the corresponding information flow (left screen).	30
Figure 16: Operational Prototype Demonstration: The messages (central screen) that are exchanged after the image analysis, along with the corresponding information flow (left screen).	31
Figure 17: Operational Prototype Demonstration: The updated incident report containing a video as an attachment, as can be seen on the right screen, during the third phase (video upload).	31
Figure 18 Operational Prototype Demonstration: The messages (central screen) that are sent to KB, PSAP and Video Analysis module after the upload of the video with the corresponding information flow (left screen).	32
Figure 19: Operational Prototype Demonstration: The messages (central screen) that are exchanged after the video analysis, along with the corresponding information flow (left screen).	32
Figure 20: Operational Prototype Demonstration: The updated incident report containing an audio recording as an attachment, as can be seen on the right screen, during the fourth phase (audio upload).	33
Figure 21: Operational Prototype Demonstration: The messages (central screen) that are sent to KB, PSAP and Audio Analysis module after the upload of the audio recording with the corresponding information flow (left screen).	33
Figure 22: Operational Prototype Demonstration: The messages (central screen) that are exchanged after the audio transcription and the text analysis of the transcription, along with the corresponding information flow (left screen).	34
Figure 23: Operational Prototype Demonstration: The metric report (right screen) that is created by the Crisis Classification component.	34
Figure 24: Operational Prototype Demonstration: The messages (central screen) sent from the Crisis Classification component to KB and subsequently to PSAP, along with the corresponding information flow (left screen).	35
Figure 25: Operational Prototype Demonstration: The Twitter simulation tool (right screen) which simulates the Social Media Analysis Tool.	36
Figure 26: Operational Prototype Demonstration: The messages (central screen) sent after the identification and insertion of the most relevant tweets from the Social Media Analysis Tool, along with the corresponding information flow (left screen).	36

List of Tables

Table 1: Social Media Monitoring Tool status.....	17
Table 2: Media Hub Tool status	17
Table 3: Image Analysis Tool status	18
Table 4: Video Analysis Tool status	18
Table 5: Automatic Speech Recognition Tool status.....	19
Table 6: SCAPP/FRAPP status	19
Table 7: Knowledge Base status	20
Table 8: Sensor Analytics Tool status	20

1 Introduction

The beAWARE project aims to perform research leading to the development of a system that supports handling of weather related crisis events. The envisioned system covers areas of a crisis lifecycle taking place in the past, present, and future. Namely, there are provisions and components that help with the forecast of future events, there are components that help to provide a current accurate understanding of the crisis event at hand and help handling the event. Finally, there are components that accumulate knowledge of past events to help handling future events, as such beAWARE can be viewed as a learning platform which makes use of knowledge from previous incidents to help respond better to a current incident.

The technological roadmap deliverable (D7.1 – beaware technological roadmap) determined the main attributes and timelines for the development of the different components of the BeAWARE platform, as well as the integration steps of all the separate components into an operational platform. The system requirements deliverable (D7.2 – System Requirements and Architecture) further described the architecture of the proposed platform including main system requirements driving it and the way the proposed design will enable achieving the ambitious goals set out at the proposal, via the use cases.

The main goal of the current deliverable is to describe the progress that has been made towards the implementation of the operational BeAWARE platform and map it to the proposed timeline that has been determined in the technological roadmap. This document presents the implementation stage of each separate component and the operational platform as a whole, along with the infrastructure that is used, the code organization and the architecture that has been implemented. The deliverable also presents the first simple operational Use Case scenario that has successfully been implemented in order to test the interoperability of the different integrated modules so far.

According to the project's Description of Action and the technical roadmap in Del7.1, the second milestone (MS2), which has due date month 12 (December 2018), marks the completion of the setup of the operational infrastructure of the beAWARE system (with dummy services). Specifically, a dummy end-to-end architecture is expected, with all service dummies in place, operating on test/mock data. MS2 also includes: setup of the basic text analysis infrastructure; the initial social media monitoring module; web real time communication; (vi) network infrastructure.

As will be described in detail in the following sections, an operational Prototype of the beAWARE platform has been developed, according to the technical roadmap. Moreover, services for multimedia analysis, tweeter monitoring and text analysis have been developed along with semantic representation and reasoning modules, in accordance to the expected timeplan.

2 Prototype architecture

The forming architecture of the operational platform was determined with the ultimate goal of providing accuracy and functionality for decision support systems to be able to respond well to weather related disaster scenarios. This architecture has been described in detail in D7.2, thus only a short description of the main architectural parts of the Integration Prototype will follow in this chapter.

2.1 Global View

The architecture is roughly made up of the following layers:

- 1 **Ingestion layer**, containing mechanisms and channels through which data is brought into the platform;
- 2 **Internal services layer**, is comprised of a set of technical capabilities which are consumed by different system components. This layer includes services such as generic data repositories and communication services being used by the different components;
- 3 **Business layer**, containing the components that perform the actual platform-specific capabilities;
- 4 **External facing layer**, including the end-users' applications and PSAP (Public-safety answering point) modules, interacting with people and entities outside the platform (end-users of the platform).

As can be derived from **Figure 1**, the first step in a generic flow of the platform consists of new data being pushed into the platform. That step acts as a trigger to potentially many internal components which are interested in the newly acquired data, or further processing thereof. The new data can originate from specific beAWARE applications used by civilians (end-users) or by first responders. Moreover, incoming data to the platform can originate from other sources as well, such as IoT devices and social media posts grabbed by crawlers.

Once a new piece of data is successfully ingested into the system, the data is stored in a temporary raw data store and the availability of the new piece of data is broadcasted to all interested components using a specific topic of the messaging bus service. **There is** a separate topic for each kind of information flowing into the system, such that only components which are interested in that specific kind of data will be made aware of the arrival of a new relevant piece of data. All interested parties will receive the information, which includes a pointer enabling to access the data, and shall perform their specific analysis on the new data. Note, that the result of such an analysis may in turn create a new piece of data which is of interest to other components. The end results of the analysis can be added to the knowledge base, and if required will notify PSAP related components of the identification of a new state.

The PSAP in turn may use beAWARE components, including apps to manage and alert the different stakeholders (citizens, first responders, and authorities).

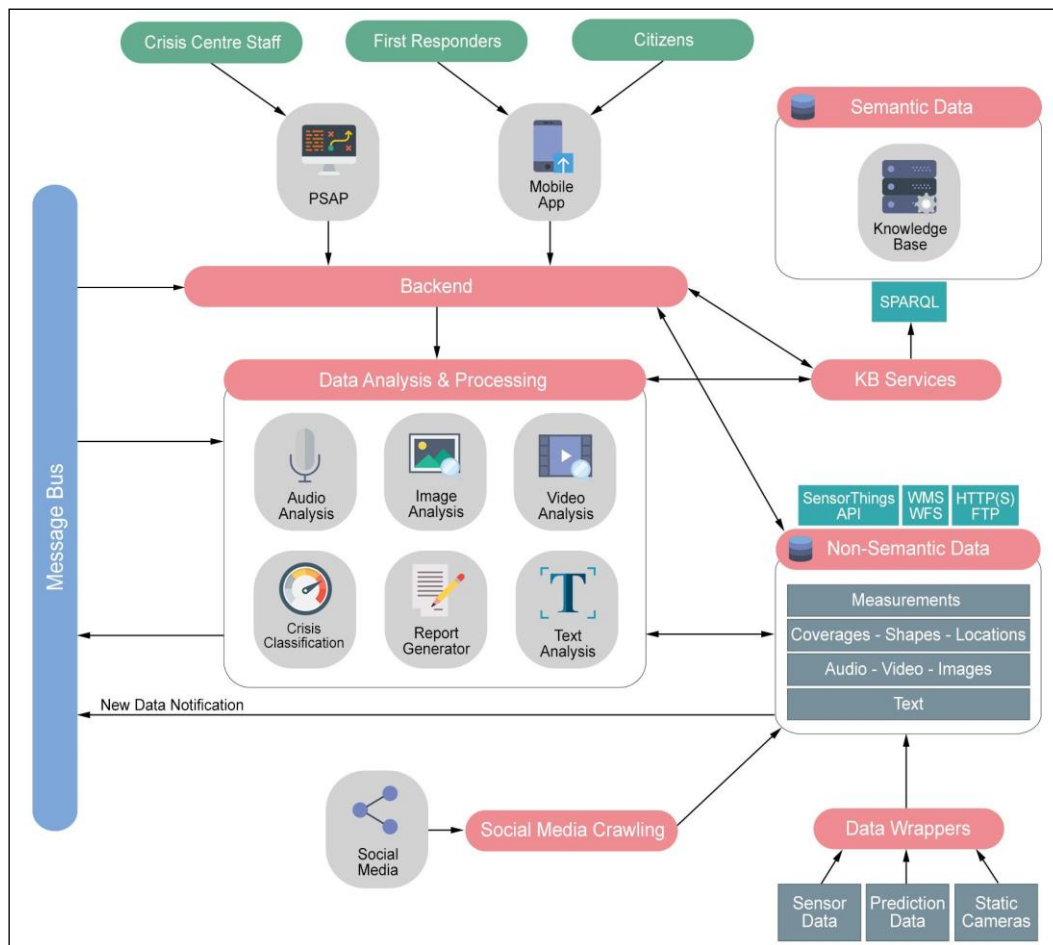


Figure 1: Architectural high-level view

2.1.1 Architecture Layers

As described above BeAware architecture consists of four main layers:

Ingestion layer – Serves as the input mechanism into the platform. Different kinds of information can serve as input to the system. For example, IoT devices and additional input mechanisms, such as dedicated applications, can produce different kinds of data such as measurements (time series), pictures, videos, audio, and more. An additional source of incoming information is weather related data. Typically, once a new piece of data has been ingested into the platform, it is stored temporarily in a raw storage system, and a proper notification is sent via the service bus to the interested parties.

Internal services – These services are used internally for the proper functioning of the capabilities provided by the various components. These are typical middleware services which are tailored for the specific use of the beAWARE system. These services are used mostly for data storage and communication. Some generic data analytics and processing services may be used as well. Examples of this layer include a central (raw) data repository, a central message bus, and a generic knowledge base. These services will be accessible to all platform components.

A particular kind of an internal service used throughout the system relates to handling semantic information. This includes extracting semantic information, storing it, and inferring new information based on accumulated data. This group of services includes:

- Knowledge base – component and service
- Semantic reasoners – component and service

Business layer – This layer encompasses the components that provide the actual platform specific capabilities. The bulk of the platform is concentrated at this layer, which interacts in turn with all additional layers.

A particular group of components in this layer tackles the analysis of different kinds of data flowing into the platform. A main aspect of the components in this category is the extraction of semantic information from various kinds of input data flowing into the platform from various different sources. Among this group we can find the following components:

- Social Media Analysis Services
- Image analysis
- Video analysis
- Automatic speech recognition - Audio analysis
- Sensor analysis

These components help to determine the current crisis classification and drive the detection of events which leads in turn to meaningful decision support.

A third group of related components deal mostly with text, both at the input and output aspects. Namely, incoming text messages, from social media, mobile applications and transcribed voice recordings, possibly in a variety of languages, are processed and analyzed, and outgoing text messages that need to be delivered by the platform are prepared:

- Multilingual Report Generator
- Multilingual Text Analyzer

External layer – This layer handles the interaction of the platform with external entities, both as input providers and output recipients. There are two main groups of components making up this layer, namely:

- Mobile applications
 - Civilians Mobile Application
 - First Responder Mobile Application
- Control centers
 - Public Safety Answering Point (PSAP)
 - Local Control Centers at the deployment sites

2.2 Components

The individual components of beAWARE have been described in detail in D7.2, thus a rough description follows here. As already mentioned, BeAware components can be categorized in the following layers.

2.2.1 Ingestion Layer

Social Media Monitoring module

The aim of the social media monitoring module is to collect posts from Twitter that appear to be relevant to the three main pilots, i.e. floods, fire, and heatwave, in their respective geographical locations. The crawling process needs to be real-time and effective, able to handle large streams of data, especially when keywords such as “fire” have multiple meanings and needs disambiguation. The module collects tweets in English, Greek, Italian and Spanish, that are published by any citizen, civil protection organization or online news website, aiming to provide relevant information about crisis events, clustered by location and delivered through a periodic Twitter report.

Monitoring machine sourcing information from IoT and M2M platforms module

Sensor data is crucial for tracking the onset and progress of a crisis. There is a large variation in sensors and an almost equally large variation in interfaces to access the data from those sensors. The aim of this module is to collect time-series sensor data from various on-line sensors and make this sensor data, and the sensor metadata, available through a standardised interface.

2.2.2 Business Layer

Semantic modelling, integration and aggregation

This component addresses the collection, aggregation and semantic integration of content relevant to emergency information in order to facilitate decision support and early warnings generation. This is accomplished through: (i) the social media monitoring module, (ii) the module responsible for monitoring machine sourcing information from IoT and M2M platforms, (iii) weather and hydrological forecasts, and (iv) the semantic representation and reasoning module.

The Semantic representation and reasoning module has overall responsibility for semantic extraction and storage capabilities in the system and consists of the following components: (a) the beAWARE Knowledge Base (KB), also referred to as “ontology”, (b) the KB Service, (c) the semantic enrichment and reasoning framework.

Early Warning

This component addresses the provisioning of the necessary technological solutions to enable the beAWARE framework to provide early warning and decision support to the PSAP. This is accomplished through: (i) the crisis classification module, (ii) the multilingual text analysis module, and (iii) the concept/event detection from multimedia and (iv) automatic speech recognition modules.

Multilingual report generation

This component addresses the generation of emergency reports and messages that should be communicated to the different types of stakeholders that beAWARE considers (authorities, first responders, citizens, etc.) during an emergency in order to provide them with tailored support. It accomplishes this through the following two modules.

PSAP

The objective of this component is to serve as a means for public safety answering points (PSAP) to obtain situational awareness and a common operational picture before and during an emergency, and to enable efficient emergency management based on a unified mechanism to receive and visualize field team positions, incident reports, media attachments, and status updates from multiple platforms and applications.

2.3 Internal Services

2.3.1 Communication Bus

The main purpose of this component is to provide generic communication capabilities among different beAWARE components and participants. It can be used to send messages and notification among components or to share information among various entities. The dominant paradigm shall be the publish/ subscribe pattern leading to event-based communication among collaborating partners by registering interest in particular events. Using this paradigm producers and consumers do not have to be aware of each other and need only to agree on the topic via which they are going to communicate, and the message format of the agreed upon topic.

2.3.2 Data management

The data management component deals with data ingestion, storage, and potentially some level of processing for shallow analysis. Moreover, data based notifications can be supported by connecting the data pipe to the cloud communication bus which serves as the messaging pipe.

2.4 External layer

2.4.1 End-users applications

There are two kinds of end-user applications envisioned for mobile devices (smartphones or tablets): one used by the first responders and another used by the general public. These mobile applications will communicate with a backend that will in turn connect to the rest of the beAWARE system.

3 Integration Approach

Due to the distributed approach of the design and implementation of the beAWARE platform, according to which different partners develop independently and maintain ownership over different components of the system, combined with the complexity of the platform, which is comprised of many different components, we opted to follow a micro-services approach to make the entire process of design, implementation, and integration more manageable, in a distributed manner.

Using this approach, we maintain the independence of the different components, and the integration focus is on the exposed capabilities of each component. The integration between components is being realized via two main mechanisms, first, inter-communication via the platform communication service bus, and second through exposed Representational state transfer (REST) interfaces of various components, potentially aided by a discovery service. In addition, data is being exchanged between components by using shared data stores, and having links to data items incorporated into messages exchanged between components.

For communicating through the communication bus, the components need only to agree on the topic via which they will be interacting, and the format of the messages published on that topic. For communicating via a REST interface, there needs to be a way for one micro-service to find another micro-service it wishes to invoke. For that purpose, a service discovery component may be deployed. Such a component serves as a central registry of available micro-services, responding to queries about the location of a certain micro-service.

The system development, as already described in Deliverable 7.2, follows an iterative approach, from an initial dummy prototype to the final version, and follows the above cycle:

1. Integration Prototype: Dummy end-to-end architecture; most service dummies in place, operating on test/mock data. This is expected by the end of the first year of the project as a part of MS2
2. First Prototype: Using real services from WP3-WP6; First Prototype milestone (MS3), expected at M18.
3. Second Prototype: Using real services from WP3-WP6; Second Prototype milestone (MS4), expected at M24.
4. Final System: Complete beAWARE platform; Final System milestone (MS5), expected by the end of the project (M36).

According to the expected timeline for MS2, an initial Integration Prototype has been implemented and communication between different components has been achieved, by implementing a simple use case, as will be described in the following sections. Moreover, apart from dummy services, some services already have some integrated functionality on them. The implemented services are described in detail in the next chapter.

4 Components and Integration status

Table 1: Social Media Monitoring Tool status

Service name	Social Media Monitoring Tool
CI cluster	social-media-analysis
Functional Description	Version 0.5, baseline version
Deployment status	Deployed
Integration status	Integrated
Integration issues / dependencies	All dependencies are included in pom.xml. Environment variables: SECRET_MH_API_KEY, SECRET_MH_BROKERS, SECRET_MONGO_URI
Next steps	- Send simulated tweets separated. - Replace dummy Twitter report.

Table 2: Media Hub Tool status

Service name	Central Hub to Assign Media Analysis
CI cluster	media-hub
Functional Description	Version 1.0, working version
Deployment status	Deployed
Integration status	Integrated
Integration issues / dependencies	All dependencies are included in pom.xml. Environment variables: SECRET_MH_API_KEY, SECRET_MH_BROKERS For a complete workflow, it requires CI clusters ASR, image-analysis, and video-analysis.
Next steps	

Table 3: Image Analysis Tool status

Service name	Image Analysis
CI cluster	beaware-project/image-analysis
Functional Description	Version 0.2, baseline version
Deployment status	Deployed
Integration status	Integrated
Integration issues / dependencies	Communicates with media-hub service Uses port 7788 Dependencies show up in the Dockerfile
Next steps	<ul style="list-style-type: none">• Integrate a working version of the flood-fire detection module• Optimize request handling• Optimize analysis algorithms

Table 4: Video Analysis Tool status

Service name	Video Analysis
CI cluster	beaware-project/video-analysis
Functional Description	Version 0.2, baseline version
Deployment status	Deployed
Integration status	Integrated
Integration issues / dependencies	Communicates with media-hub service Uses port 7777 Dependencies show up in the Dockerfile
Next steps	<ul style="list-style-type: none">• Integrate a working version of the object-detection and tracking module• Integrate a working version of the traffic management module• Integrate a working version of the dynamic texture detection module• Optimize request handling• Optimize analysis algorithms

Table 5: Automatic Speech Recognition Tool status

Service name	Automatic Speech Recognition
CI cluster	beaware-project/ASR
Functional Description	Version 1.0, working version
Development status	Implementation of first prototype. It includes language models for (Greek, Spanish, Italian and English). ASR module can handle several .wav formats
Deployment status	Deployed
Integration status	Integrated
Integration issues / dependencies	<p>All dependencies are included in pom.xml.</p> <p>Environment variables: SECRET_MONGO_URI</p> <p>It communicates with Media hub:</p> <ul style="list-style-type: none"> - Data in: audio file URL, recording timestamp, language - Data out: Mongo reference ID pointing to the transcription result
Next steps	<ul style="list-style-type: none"> - Increase recognition accuracy of all language models by further adapting the models as new data become available. - Expand compatibility with more audio formats - Include an automatic language identification module in order to automatically recognize the language of the speaker

Table 6: SCAPP/FRAPP status

Service name	SCAPP/FRAPP (Mobile Application)
CI cluster	Not on cluster, yet
Functional Description	End user application for mobile devices.
Development status	Implementation of first prototype. This includes a first version of the UI as well as basic communication functions.
Deployment status	Not deployed to cluster, yet.
Integration status	Communication interface nearly clarified with other partners. Basic communication function tested in coordination with other components.
Integration	No issues regarding the integration with other components.

issues / dependencies	Source code should not be available to public. Therefore not possible to participate in continuous integration process.
Next steps	Integrate deployment to K8 cluster. Finalize implementation of function to send incident report.

Table 7: Knowledge Base status

Service name	KB
CI cluster	Not on cluster, yet
Functional Description	Server and management APIs for semantic data.
Development status	Knowledge Base fully developed. First version of ontology deployed to Knowledge Base.
Deployment status	Not deployed to cluster, yet.
Integration status	Provided interfaces for other components to access/modify semantic data.
Integration issues / dependencies	No issues regarding the integration with other components. Source code should not be available to public. Therefore not possible to participate in continuous integration process.
Next steps	Integrate deployment to K8 cluster. Clarifying further requirements and changes to the ontology during development of other components.

Table 8: Sensor Analytics Tool status

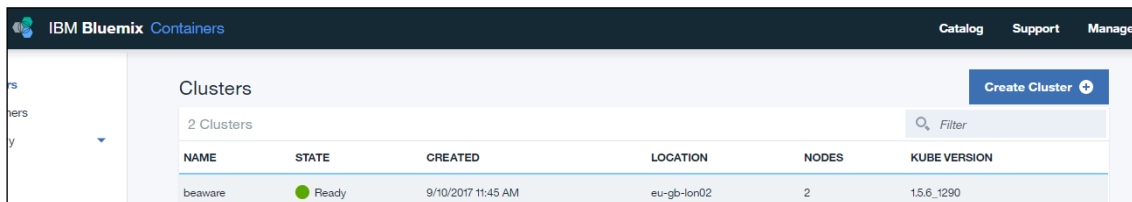
Service name	SENSAN
CI cluster	Deployed to cluster.
Functional Description	Server and management APIs for sensor data.
Development status	Server fully implemented. Data integration ongoing. Event detection in planning.
Deployment status	Server deployed to cluster. Data import and process scripts not yet.
Integration status	Data import ongoing. Data visualization integrated in UI.
Integration issues /	No issues regarding the integration. Data import not fully specified and clarifications ongoing.

dependencies	
Next steps	Deploy data import and processing to the cluster and specify the event detection requirements.

5 Infrastructure and Code Organization

After having defined the global architecture of the beAWARE platform, focusing on its high-level component design we turn to deployment and hosting issues, focusing on the provisioning and operation of the infrastructure, on which the beAWARE system will run. This includes the internal services, business services, associated repositories and external entities (mobile applications, control centers).

The bulk of the system component is hosted on a Kubernetes (K8s) cluster managed on BlueMix, IBM's public cloud, as can be seen in **Figure 2**, **Figure 3**, and **Figure 4**.



The screenshot shows the IBM Bluemix Containers interface. At the top, there's a navigation bar with 'Catalog', 'Support', and 'Manage'. Below it, the 'Clusters' section is active, showing a table with 2 clusters. The table has columns: NAME, STATE, CREATED, LOCATION, NODES, and KUBE VERSION. One cluster named 'beaware' is listed with a 'Ready' state, created on 9/10/2017 at 11:45 AM, located in 'eu-gb-lon02', with 2 nodes, and Kube version 1.5.6_1290. A 'Create Cluster' button is visible in the top right.

NAME	STATE	CREATED	LOCATION	NODES	KUBE VERSION
beaware	Ready	9/10/2017 11:45 AM	eu-gb-lon02	2	1.5.6_1290

Figure 2: beAWARE Kubernetes cluster

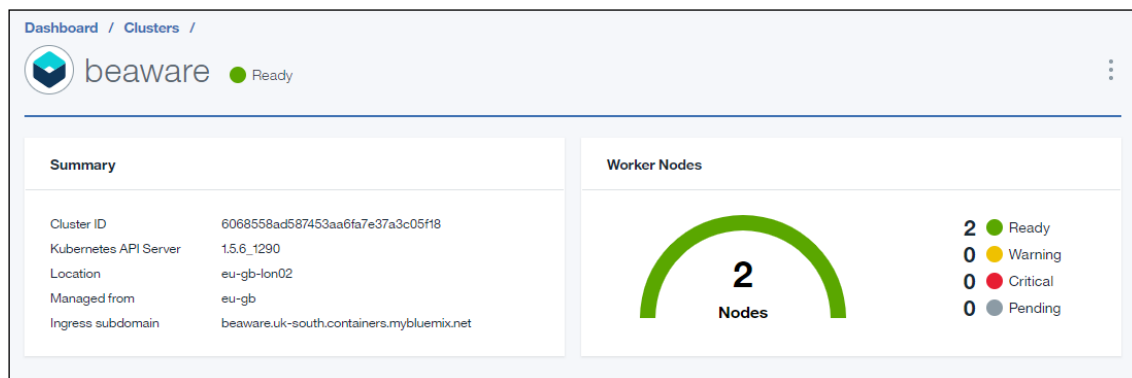
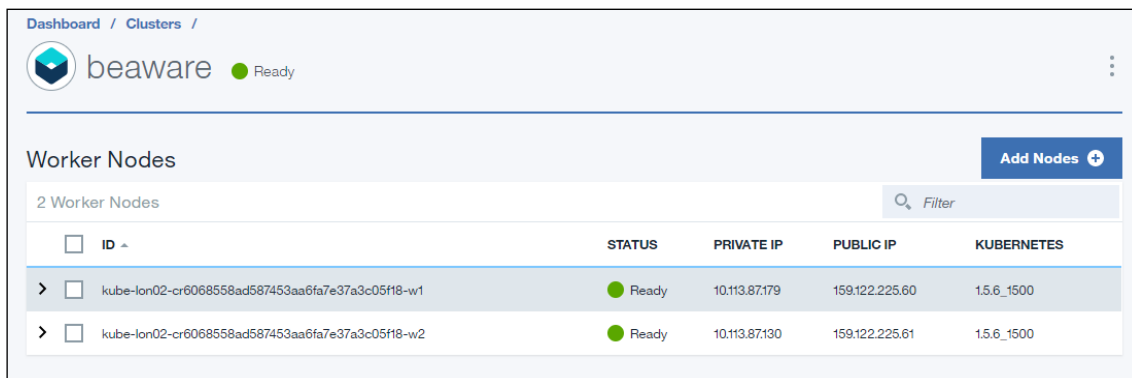


Figure 3: beAWARE cluster overview



The screenshot shows the 'Worker Nodes' section of the cluster overview. It displays a table with 2 worker nodes. The table has columns: ID, STATUS, PRIVATE IP, PUBLIC IP, and KUBERNETES. Both nodes are in a 'Ready' state. A legend on the right indicates the status of nodes: 2 Ready (green), 0 Warning (yellow), 0 Critical (red), and 0 Pending (grey).

ID	STATUS	PRIVATE IP	PUBLIC IP	KUBERNETES
kube-lon02-cr6068558ad587453aa6fa7e37a3c05f18-w1	Ready	10.113.87.179	159.122.225.60	1.5.6_1500
kube-lon02-cr6068558ad587453aa6fa7e37a3c05f18-w2	Ready	10.113.87.130	159.122.225.61	1.5.6_1500

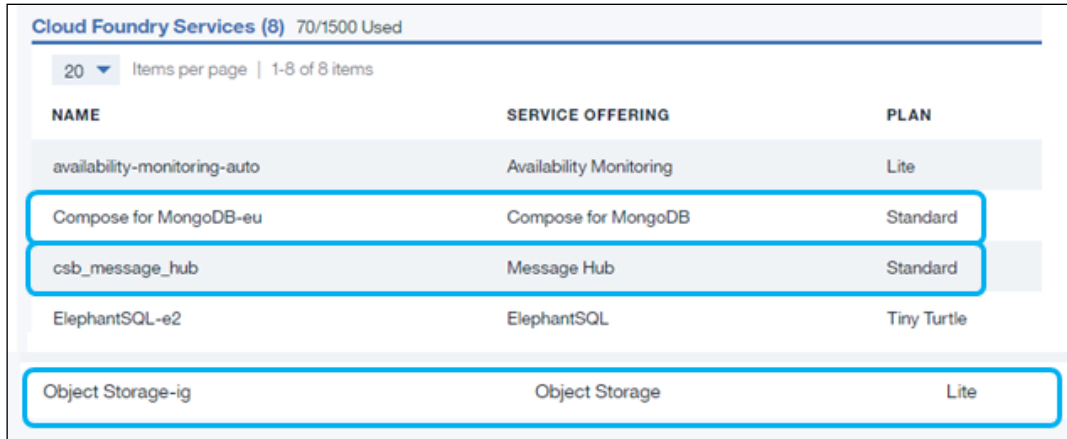
Figure 4: Kubernetes cluster worker nodes

Currently the cluster is composed of two worker nodes, and it may grow based on evolving system needs. The K8s cluster is divided into 3 namespaces:

- Default (Jenkins Master)
- Build (Jenkins Slave)

- Prod (Deployed Applications) – residing behind Ingress (a reverse Proxy acting as a gateway)

In addition, there are cloud services that are used by beAWARE components, such as the messaging bus and data repositories (**Figure 5**). These services are hosted on IBM's public cloud and offer binding capabilities to all beAWARE components.



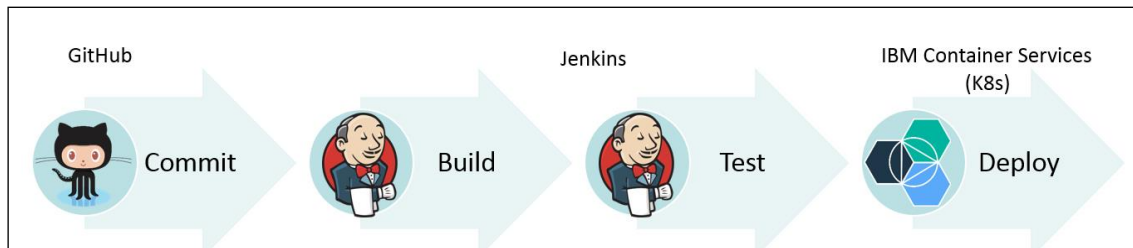
NAME	SERVICE OFFERING	PLAN
availability-monitoring-auto	Availability Monitoring	Lite
Compose for MongoDB-eu	Compose for MongoDB	Standard
csb_message_hub	Message Hub	Standard
ElephantSQL-e2	ElephantSQL	Tiny Turtle
Object Storage-ig	Object Storage	Lite

Figure 5: MessageHub cloud service

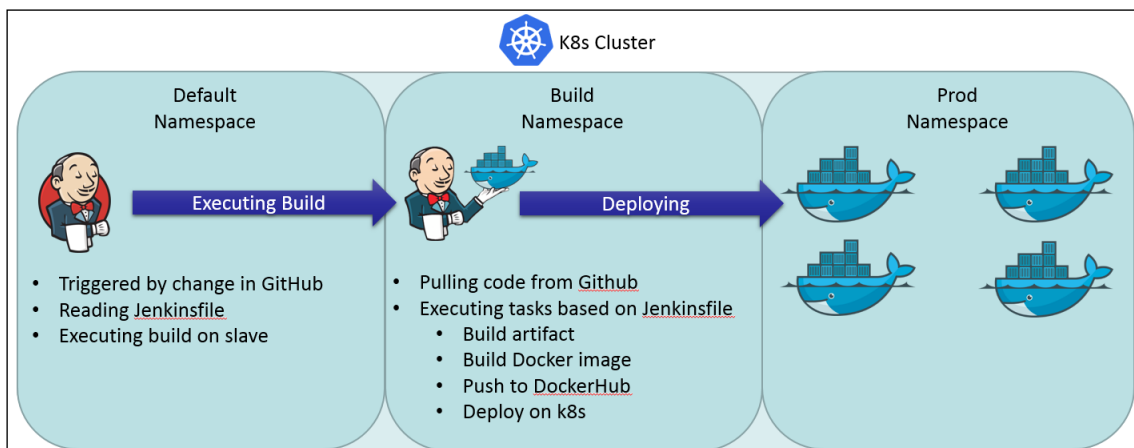
Moreover, there are components that are deployed external to the project K8s cluster, namely the PSAP and additional control center capabilities, as well as the end-user applications which will have their front-end deployed on mobile devices, while their backend may still reside within the project cluster.

The Continuous Integration (CI) environment is comprised of the following components, as depicted in **Figure 6**:

1. GitHub repository: all components should have a repository under the beAWARE project (<https://github.com/beAWARE-project>).
2. Docker – a docker image is created for each component.
 - Generally, requires a dockerFile for each component
3. Jenkins: build, test, and deploy
 - Requires a JenkinsFile for each component
 - Builds are executed in separate environment (namespace) in the project's K8s cluster Executing tasks based on Jenkinsfile
 - Pulling code from Github
 - Build artifact
 - Build Docker image
 - Push to DockerHub
 - Deploy on k8s
4. Kubernetes -IBM container services - managed cluster on which all components are deployed
 - Requires specific Kubernetes configuration for each component

**Figure 6: CI workflow**

The automated workflow kicks in upon a new commit to the master branch in the project github repository. Jenkins keeps track of such changes via web hooks, and initiates the procedure as specified in the JenkinsFile. The standard procedure is to build the component using the dockerFile, and if no errors reported, to deploy to the Kubernetes cluster, using the specified K8s configuration. This process happens for every repository for which there is an associated JenkinsFile. A more elaborated pictorial view of the CI workflow can be seen in **Figure 7**. Some details of the kubernetes cluster used by the platform can be seen in **Error! Reference source not found..**

**Figure 7: CI detailed description**

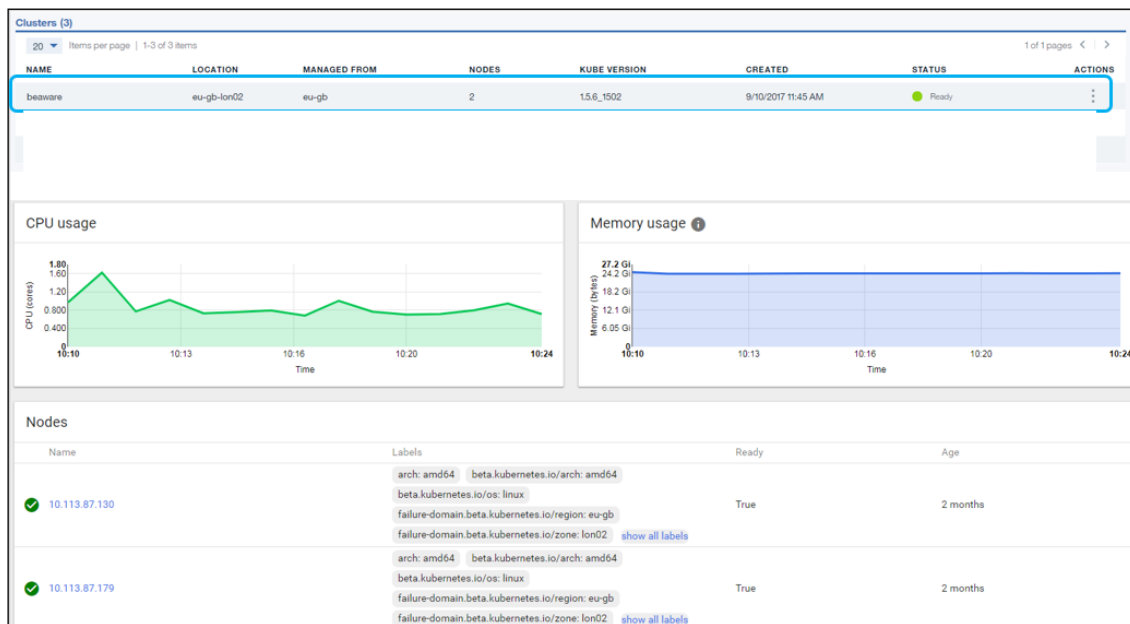


Figure 8: beAWARE Kubernetes cluster

The individual components deployed within the cluster can be seen in **Figure 9**. The beAWARE code on the GitHub repository is organised on a per-component basis. The root of the source tree is located at: <https://github.com/beAWARE-project>.

The code of the individual components can be found in the following links:

- **Text Analysis module:** Text analysis tools to extract information from tweets or other sources, Maven package (<https://github.com/beAWARE-project/text-analysis>)
- **Text analysis on ASR outputs:** Maven package (<https://github.com/beAWARE-project/text-analysis-asr>)
- **Automatic Speech Recognition tool:** for the transcription of audio recordings sent through the mobile app, Maven package (<https://github.com/beAWARE-project/ASR>)
- **Social Media Analysis tool:** A crawler that collects tweets and pushes the relevant ones to the bus, Maven package (<https://github.com/beAWARE-project/social-media-analysis>)
- **Social Media Annotation tool:** A web application to present crawled tweets and to annotate as (ir)relevant (<https://github.com/beAWARE-project/social-media-annotation-tool>)
- **Image Analysis tool:** Performs image analysis for the beAWARE project, Python (<https://github.com/beAWARE-project/image-analysis>)
- **Video Analysis tool:** Performs video analysis for the beAWARE project, Python (<https://github.com/beAWARE-project/video-analysis>)
- **Media Hub:** A central hub to receive any media and forward it to the correct component (audio/image/video), Maven package (<https://github.com/beAWARE-project/media-hub>)
- **Ontology:** the beAWARE Knowledge Base Ontology (<https://github.com/beAWARE-project/ontology>)

- **SensorThingsAPI server:** A Server API for monitoring measurements from weather forecasting and water sensors (<https://github.com/beAWARE-project/sensor-things-server>)
- **SensorThings tools:** Management API for monitoring measurements from weather forecasting and water sensors (<https://github.com/beAWARE-project/sensor-things-tools>)
- **Bus Producer API:** An API that offers a service for POST requests to produce messages to beAWARE Bus, Python (<https://github.com/beAWARE-project/bus-producer-api>)
- **Bus logger daemon:** Constantly monitors the beAWARE Bus and logs messages of certain topics to a database, Python (<https://github.com/beAWARE-project/bus-logger-daemon>)
- **Report Generation:** Generates new incident reports, Maven package (<https://github.com/beAWARE-project/report-generation>)
- **K8s:** BeAWARE Kubernetes applications (<https://github.com/beAWARE-project/k8s>)
- **Object Storage Application sample:** Applications for storing and retrieving from the data repository, Maven package (<https://github.com/beAWARE-project/object-storage-app-sample>)

Deployments				
Name	Labels	Pods	Age	Images
✓ image-analysis	app: image-analysis	1 / 1	18 hours	beaware/image-analysis:3
✓ media-hub	app: media-hub	1 / 1	9 days	beaware/media-hub:4
✓ my-nginx	run: my-nginx	2 / 2	2 months	nginx
✓ object-storage-app	app: object-storage-app	1 / 1	a month	beaware/object-storage-app
✓ sensor-things-server	app: sensor-things-server	1 / 1	29 days	beaware/sensor-things-server:13 million/postgis:latest
✓ social-media-analysis	app: social-media-analysis	1 / 1	a month	beaware/social-media-analysis:30
✓ social-media-analysis-mongo	app: social-media-analysis-mongo	1 / 1	a month	beaware/social-media-analysis-mongo
✓ social-media-annotation-tool	app: social-media-annotation-tool	1 / 1	a month	beaware/social-media-annotation-tool:11
✓ text-analysis	app: text-analysis	1 / 1	a month	beaware/text-analysis
✓ video-analysis	app: video-analysis	1 / 1	15 days	beaware/video-analysis:22

Figure 9: Cluster deployments

6 Demonstrator URLs and information

In order to demonstrate the communication flow between the different beAWARE components, a simple use case (UC) was implemented which will be described in detail in this section.

For the graphical representation and monitoring of the message exchange between the different components, a Graphical User Interface has been developed, serving as a message logger. As can be observed in **Figure 10**, the GUI consists of three panels, one (left screen) for the graphical representation of the live information flow between different components through the communication bus, one message logger (central screen) containing information about the messages that are published in the bus and one panel (right screen) which is used as a simulator of the message triggers from the Mobile Application, the Crisis Classification and the Social Media Analysis tool.

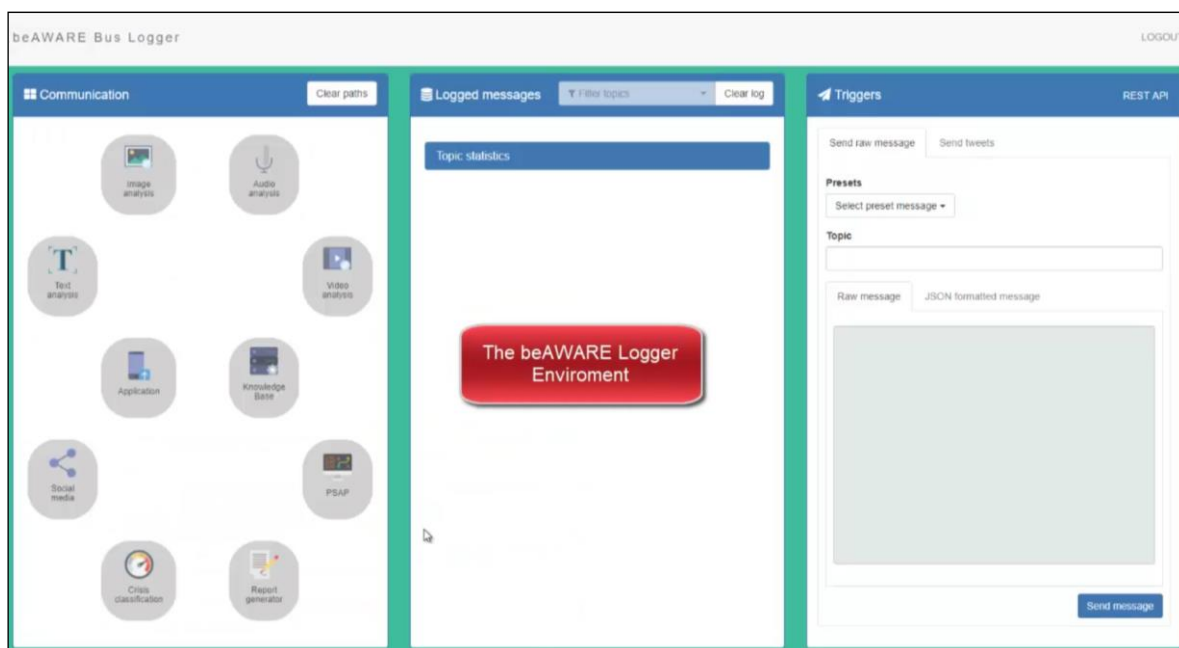


Figure 10: The beAWARE Logger Environment

In order to demonstrate the whole function of the operational prototype and the implementation of the simple UC, a demonstrator video has been created, capturing the whole function of the logger throughout the UC and it is available in the following link:

<http://beaware-project.eu/wp-content/uploads/2017/12/beAWAREOperationalPrototypeFinal.mp4>

Briefly, the demonstration has 6 phases:

1. A first responder uses the Mobile Application and informs the authority (PSAP) through the beAWARE for a new incident with its coordinates and initial description.
2. An image is uploaded from the Mobile Application to beAWARE and analyzed. The results update the incident and the PSAP is informed.
3. A video is uploaded from the Mobile Application to beAWARE and analyzed. The results update the incident and the PSAP is informed.

4. A voice recording is uploaded from the Mobile Application to beAWARE and analyzed. The results update the incident and the PSAP is informed.
5. The beAWARE Crisis Classification component inserts into the beAWARE measurements from the weather forecasting and the water sensors and then informs the PSAP.
6. The beAWARE Social Media Analysis tool selects all the relevant to the incident tweets.

The rest of this section contains a tutorial with descriptive screenshots in order to visualize the function of the demonstration logger during the aforementioned phases. Specifically, during the first phase of the UC, the trigger simulation panel is used in order to simulate the creation of the new incident from the Mobile Application. As can be seen in **Figure 11** and **Figure 12**, by selecting the “Alert” option from the drop-down menu in “Send raw message” tab, a new alert message is created with coordinates and description of the incident. No media have been uploaded yet. Then, by clicking the “Send message” button the message is sent to the Knowledge Base, which in turn sends only the location of the incident to the PSAP. The two corresponding messages are depicted in the central panel, as can be seen in **Figure 13**. By clicking on one of the messages, the body of the message is revealed. The flow of information between the components is also depicted in the left screen.

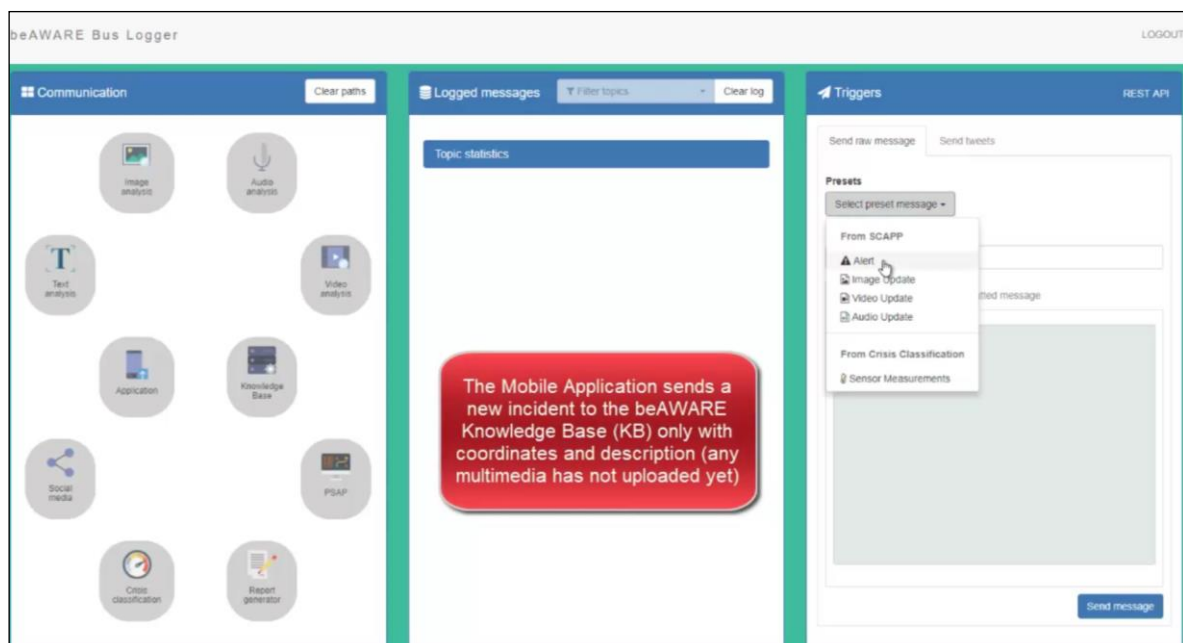


Figure 11: Operational Prototype Demonstration: The triggers simulator’s drop-down menu that creates the incident messages.

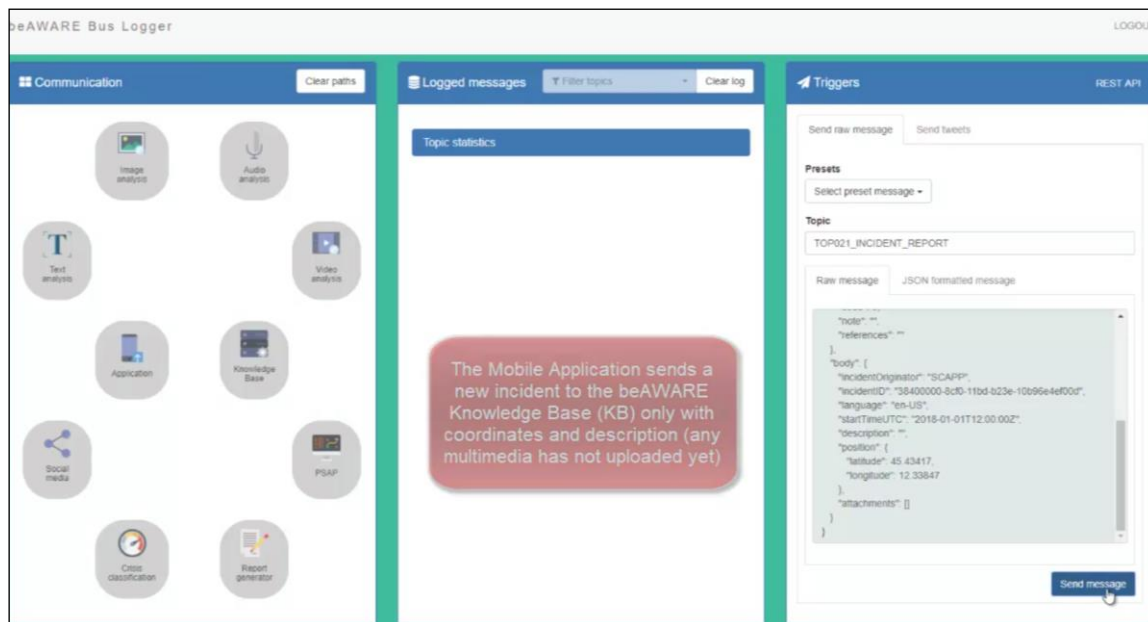


Figure 12: Operational Prototype Demonstration: The structure of the new incident message (right screen) during the first phase (initial incident report).

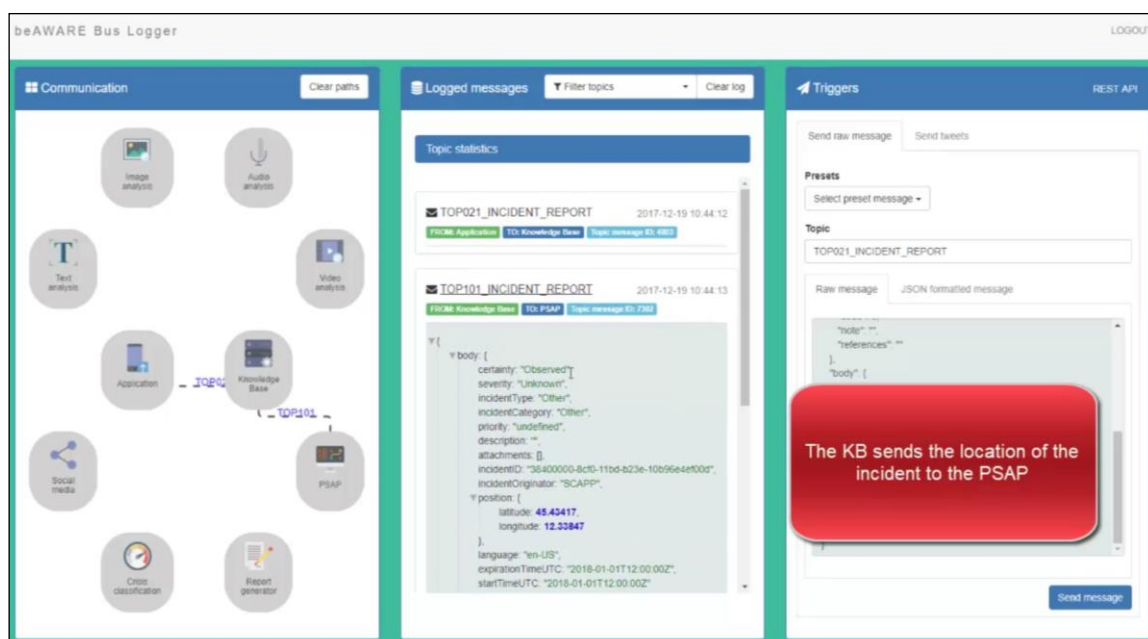


Figure 13: Operational Prototype Demonstration: The messages that are sent to KB and consequently to PSAP (central screen) during the first phase (initial incident report).

During the second phase, an image is uploaded from the Mobile Application to beAWARE and analyzed. In order to do so, from the drop-down menu of the trigger simulator we select the “image update” option and the incident report is updated by adding an image as an attachment and its corresponding metadata, as can be seen on the right screen of **Figure 14**. After we click the “Send message” button, the simulator sends the updated incident report message to the bus in order to inform the KB and Image Analysis module (TOPIC021_incident_report) for the existence of a new image. The KB in turn will inform PSAP about the new image (TOPIC101_incident_report). The relevant messages and

information flow can be seen in **Figure 15**. Consequently, the Image Analysis module will analyze the image in order to extract conceptual information regarding the incident. The analysis result is saved in the KB, which updates the incident severity, probability and certainty and informs PSAP about the incident update. Additionally, the KB requests new title and description from the Report Generator and updates the incident. **Figure 16** depicts the new messages that are exchanged after the analysis of the image.

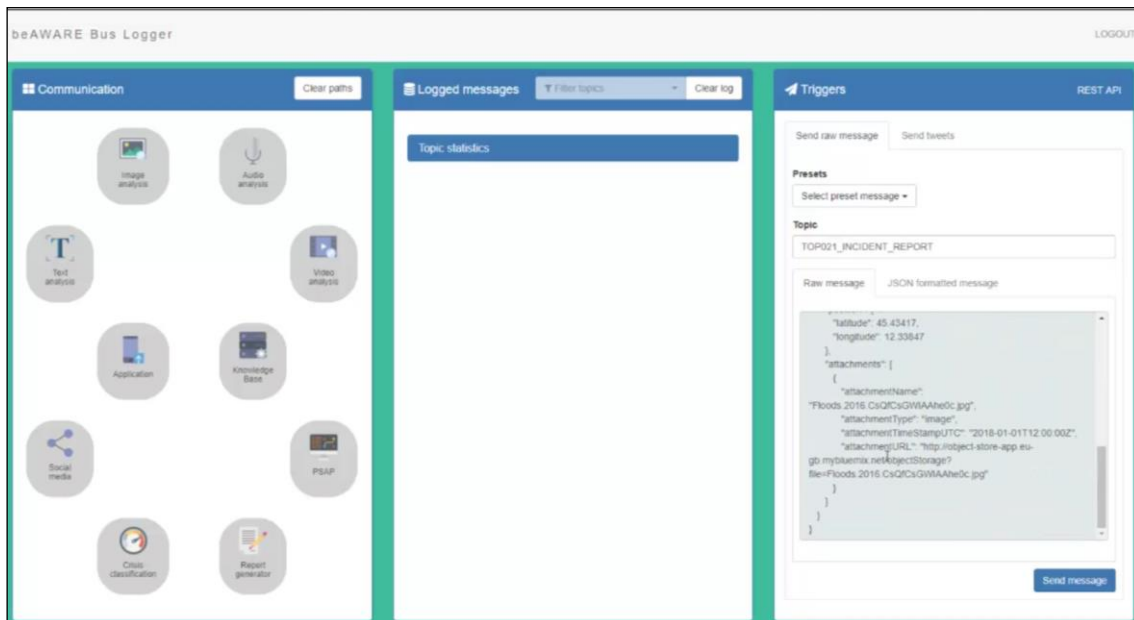


Figure 14: Operational Prototype Demonstration: The updated incident report containing an image as an attachment, as can be seen on the right screen, during the second phase (image upload).

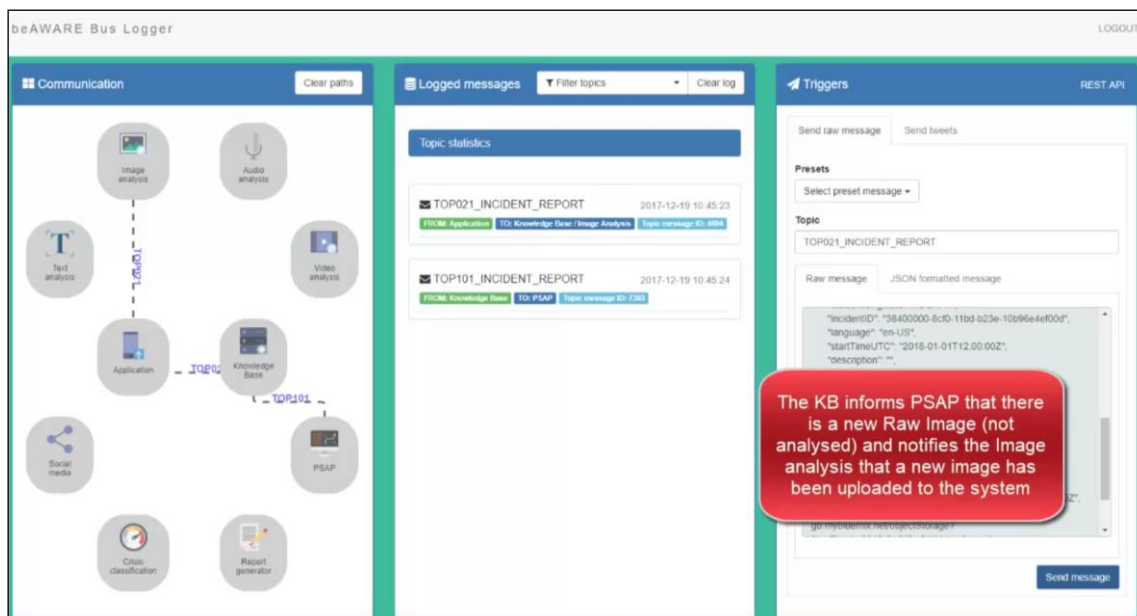


Figure 15: The messages (central screen) that are sent to KB, PSAP and Image Analysis module after the upload of an image, along with the corresponding information flow (left screen).

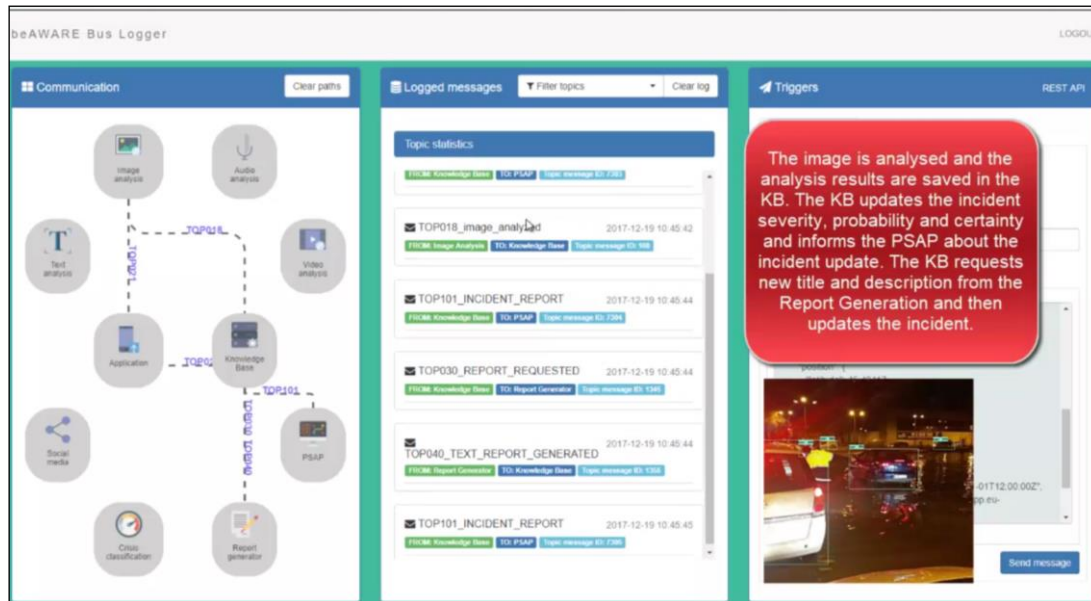


Figure 16: Operational Prototype Demonstration: The messages (central screen) that are exchanged after the image analysis, along with the corresponding information flow (left screen).

The same procedure as the one described in the second phase (image upload) is being followed for the third phase also, but instead of an image the user uploads a video from the incident this time. **Figure 17**, **Figure 18** and **Figure 19** depict the generation of the updated incident report from the simulator after the upload of the video and the messages exchanged before and after the analysis of the video.

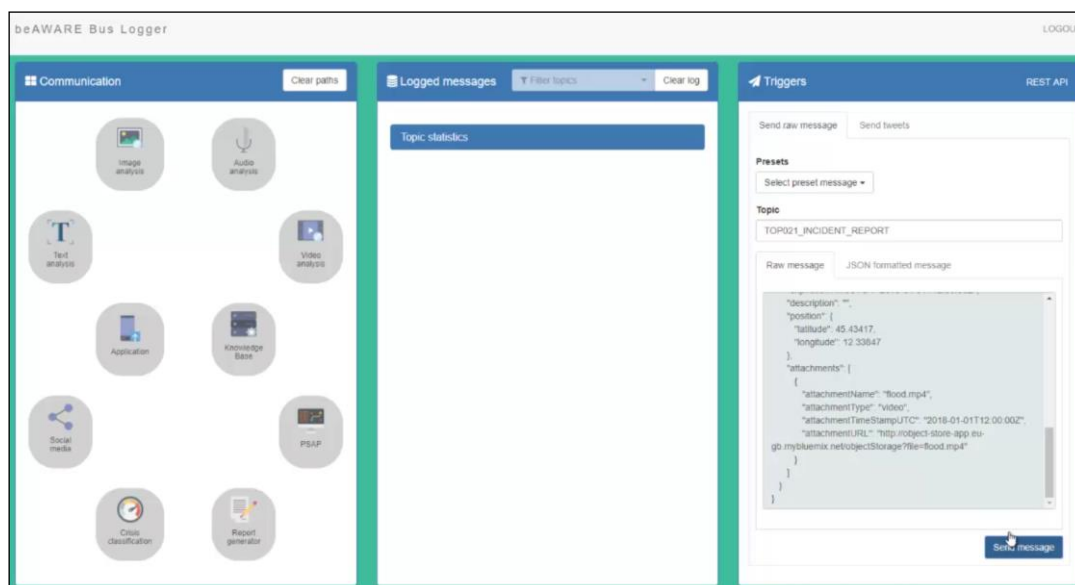


Figure 17: Operational Prototype Demonstration: The updated incident report containing a video as an attachment, as can be seen on the right screen, during the third phase (video upload).

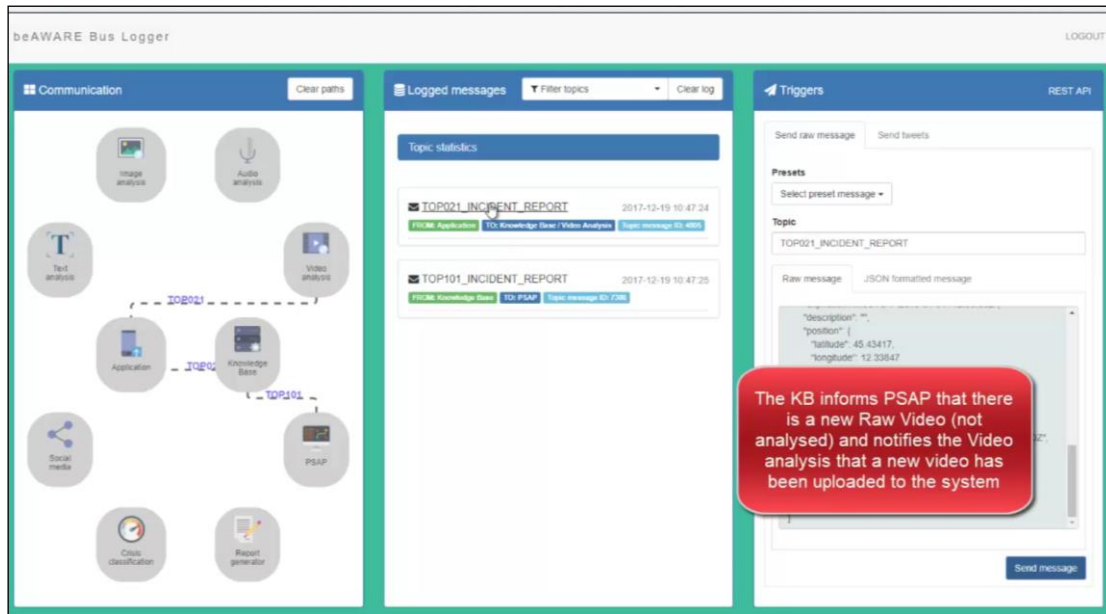


Figure 18 Operational Prototype Demonstration: The messages (central screen) that are sent to KB, PSAP and Video Analysis module after the upload of the video with the corresponding information flow (left screen).

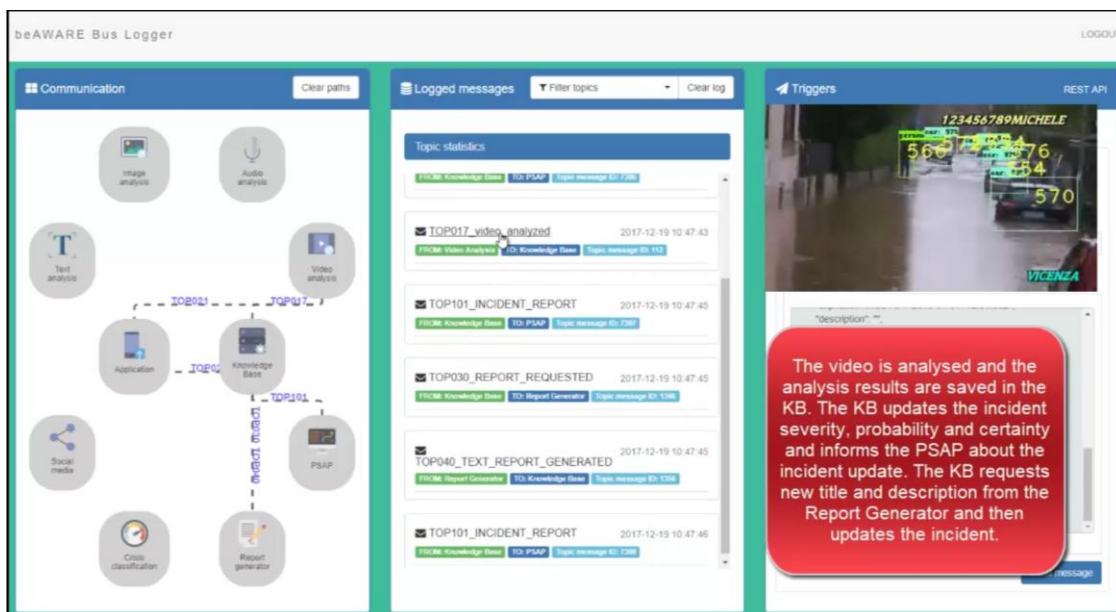


Figure 19: Operational Prototype Demonstration: The messages (central screen) that are exchanged after the video analysis, along with the corresponding information flow (left screen).

During the fourth phase of the UC, an audio recording is being uploaded in order to be analyzed and included in the same incident. The simulator updates the incident report by selecting “Audio update” from the drop-down menu and sends the relevant message to KB and Audio Analysis module through the message bus, by pressing the “Send message” button as depicted in **Figure 20**. Consequently, the KB informs PSAP that there is a new audio recording. The relevant messages (TOPIC021_INCIDENT_REPORT and TOPIC101_INCIDENT_REPORT) appear in the message panel, as can be seen in **Figure 21**.

Finally, the audio is transcribed by the Audio Analysis module and the transcription is sent to the Text Analysis component. The results of the Text Analysis are stored in the KB and the Report Generator creates a new report to inform the PSAP.

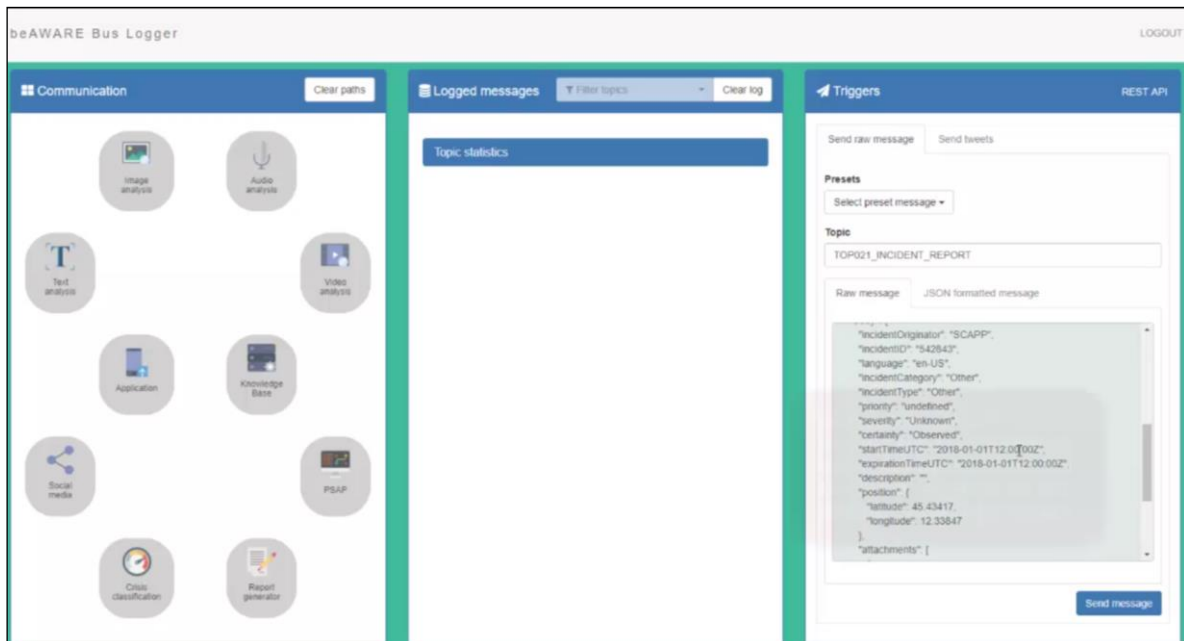


Figure 20: Operational Prototype Demonstration: The updated incident report containing an audio recording as an attachment, as can be seen on the right screen, during the fourth phase (audio upload).

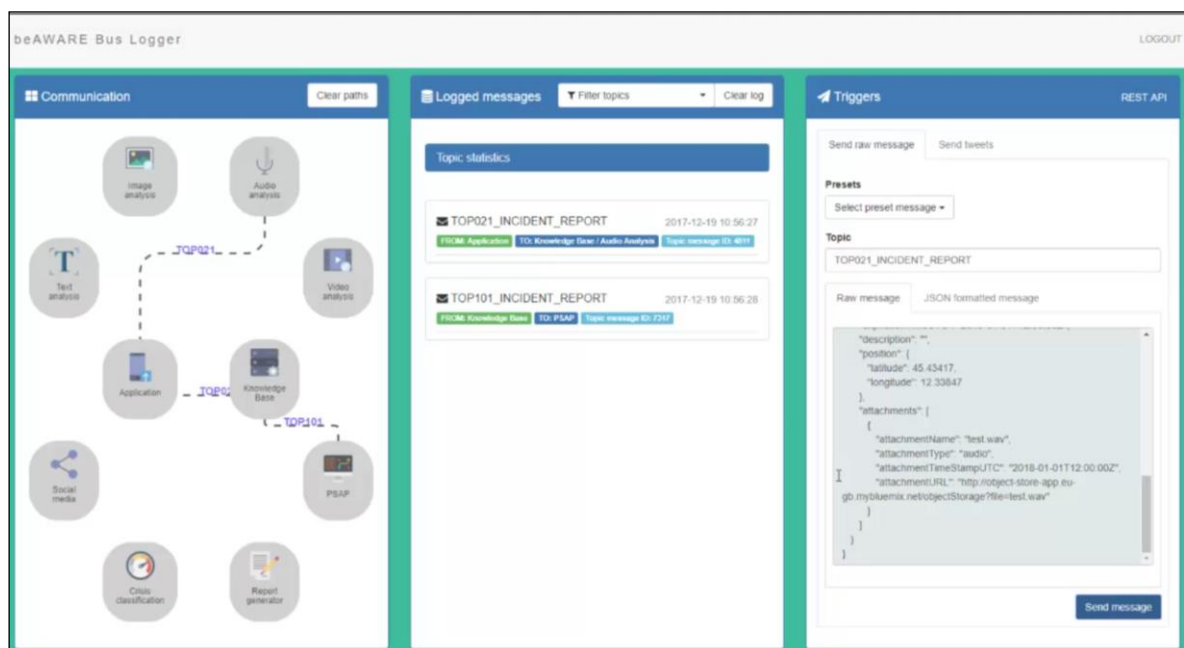


Figure 21: Operational Prototype Demonstration: The messages (central screen) that are sent to KB, PSAP and Audio Analysis module after the upload of the audio recording with the corresponding information flow (left screen).

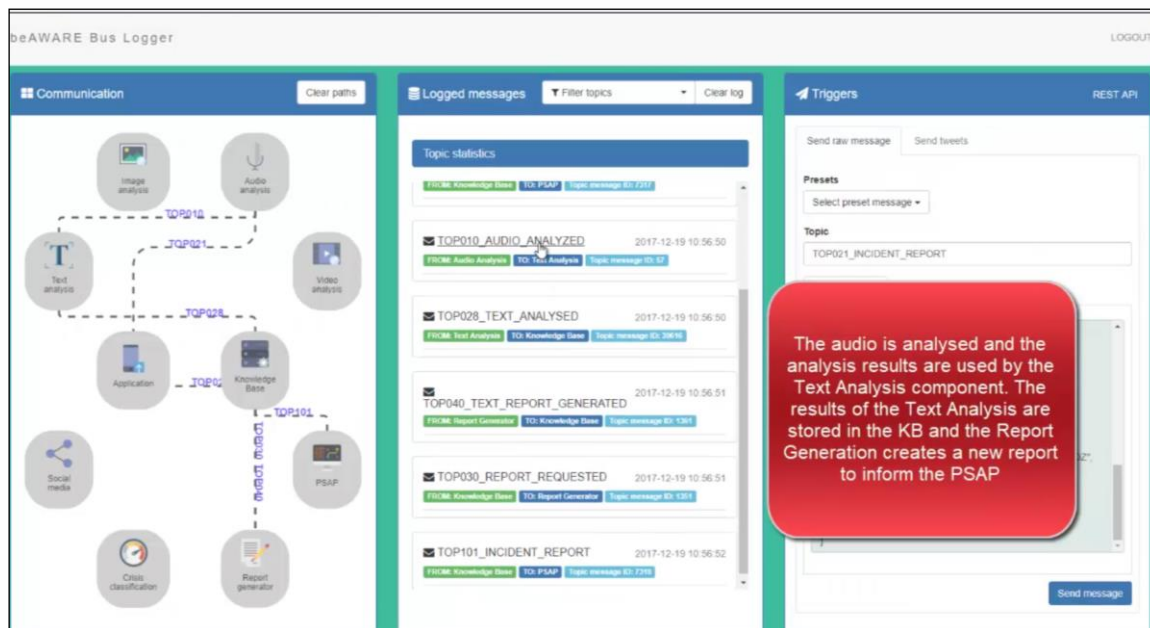


Figure 22: Operational Prototype Demonstration: The messages (central screen) that are exchanged after the audio transcription and the text analysis of the transcription, along with the corresponding information flow (left screen).

The fifth phase of the UC involves the Sensor-thing Server (water sensors) and Crisis Classification component. By selecting “Sensor Measurements” from the drop-down menu the Crisis Classification component receives the measurements from the Sensor-thing Server and creates a metric report (TOPIC104_METRIC_REPORT), as can be seen in **Figure 23**. After pressing the “Send message” button, the metric report is sent to the KB and KB sends the relevant information to PSAP (TOPIC101_INCIDENT_REPORT), as depicted in **Figure 24**.

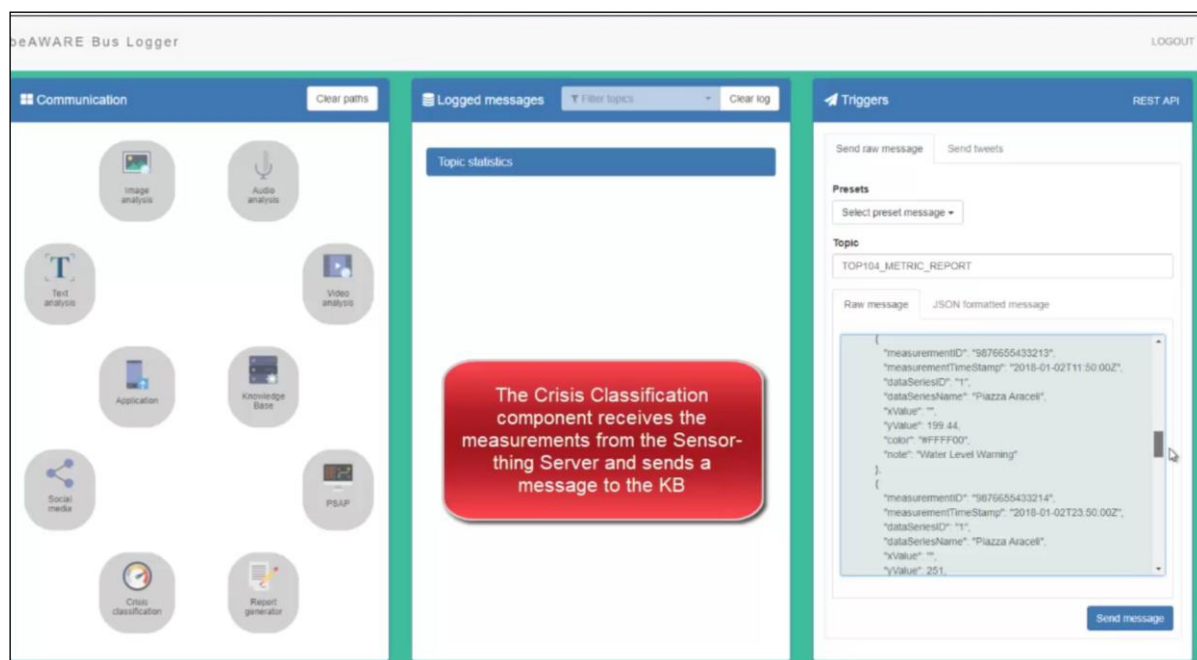


Figure 23: Operational Prototype Demonstration: The metric report (right screen) that is created by the Crisis Classification component.

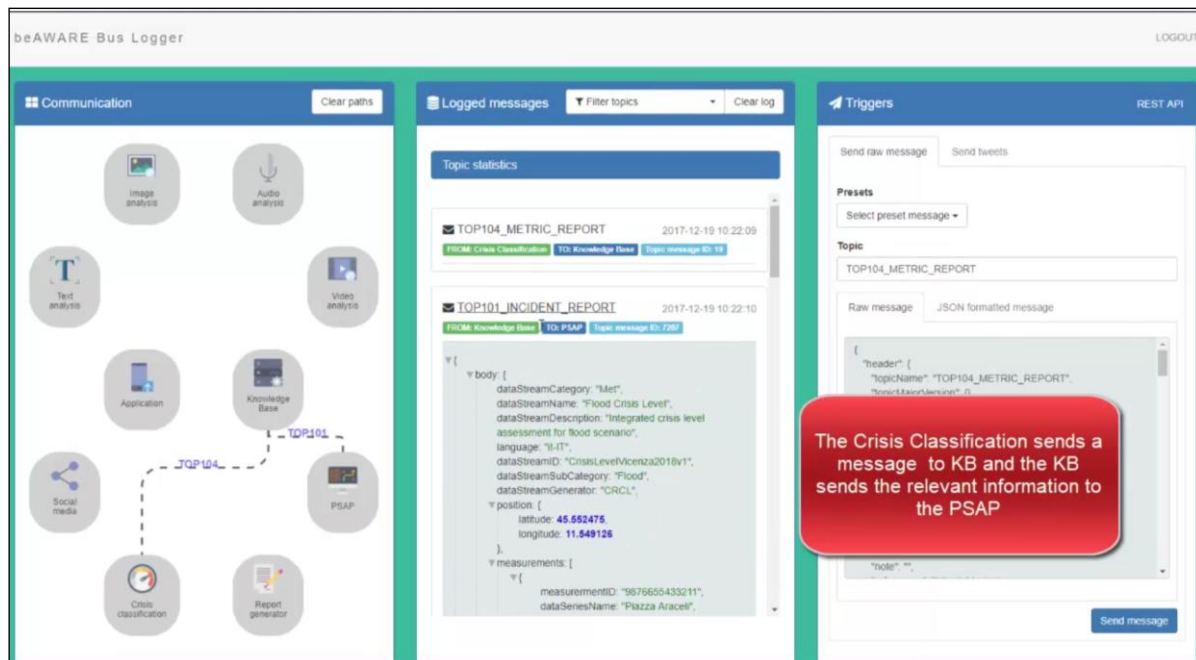


Figure 24: Operational Prototype Demonstration: The messages (central screen) sent from the Crisis Classification component to KB and subsequently to PSAP, along with the corresponding information flow (left screen).

During the sixth phase of the UC, the beAWARE Social Media Analysis tool identifies all the relevant tweets and sends them to the Text Analysis module in order to be analyzed. As depicted in **Figure 25**, by using the Twitter simulation tool ("Send tweets" tab) a number of new tweets are identified from the Social Media Analysis Tool. By pressing the add button, the Social Media Analysis Tool inserts only the relevant tweets into beAWARE (second column in the right screen in **Figure 26**). Subsequently, the Text Analysis module analyzes each relevant tweet, stores the information in the KB and a new report is generated by the Report Generator and finally the PSAP is informed by the KB.

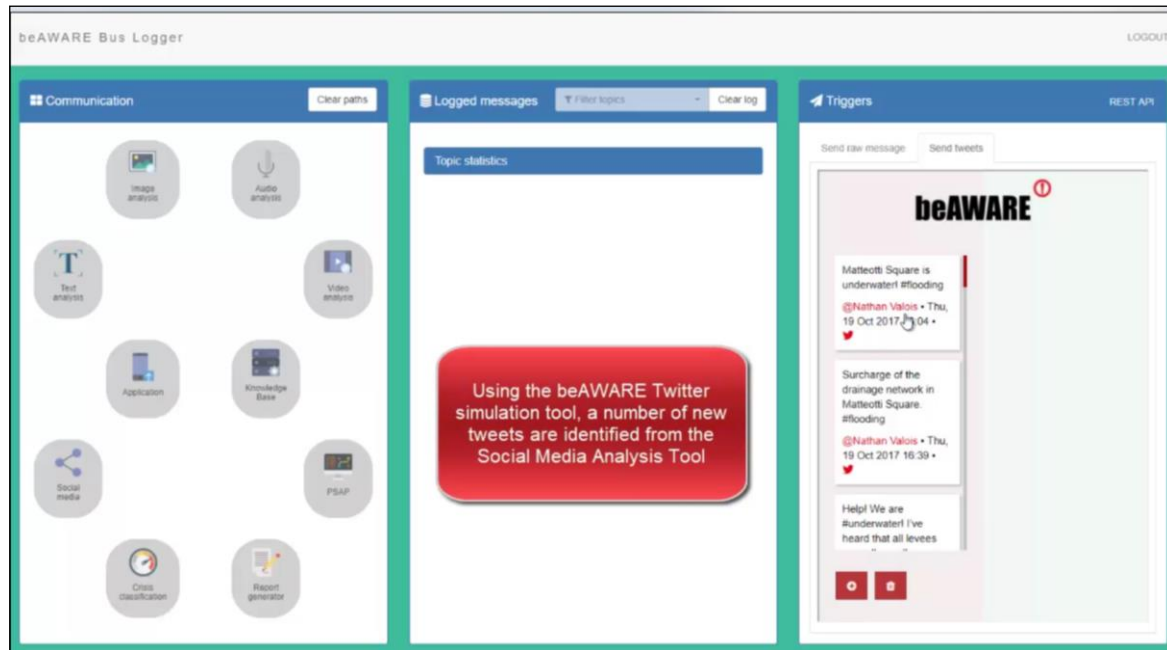


Figure 25: Operational Prototype Demonstration: The Twitter simulation tool (right screen) which simulates the Social Media Analysis Tool.

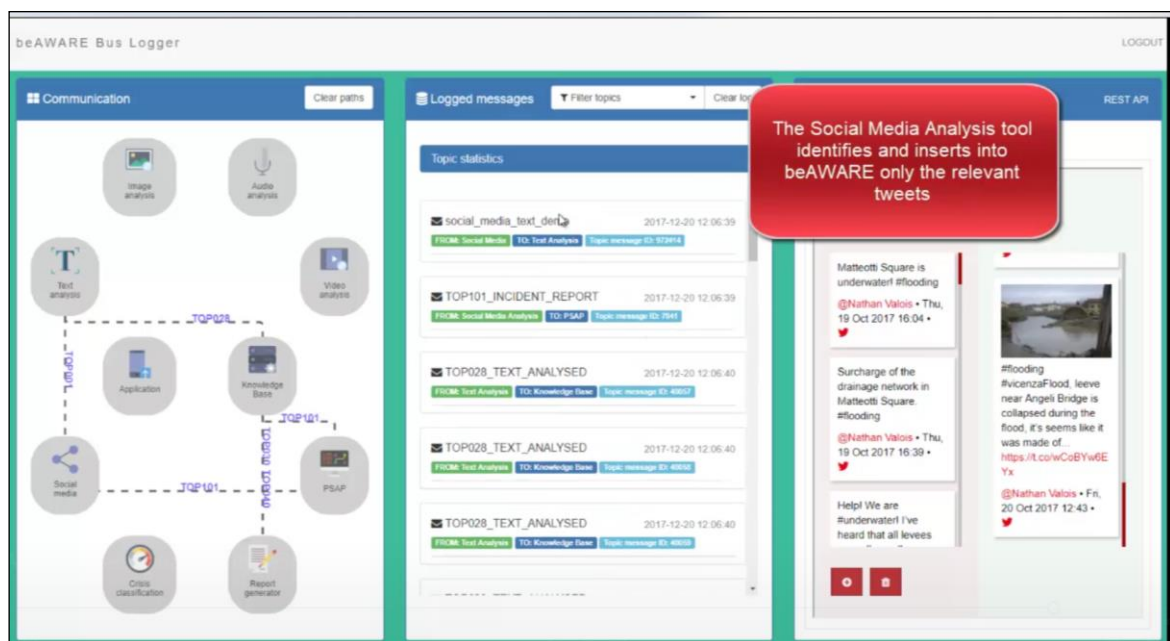


Figure 26: Operational Prototype Demonstration: The messages (central screen) sent after the identification and insertion of the most relevant tweets from the Social Media Analysis Tool, along with the corresponding information flow (left screen).

7 Summary and Conclusions

This deliverable is a demonstrator of the initial integration operational prototype of the beAWARE platform that was realized during MS2. This document presents the development status of each component, the integration approach and the infrastructure used for the development of the operational platform. The deliverable also presents a first simple operational use case scenario and a logger interface that were implemented in order to demonstrate the functionality of the platform and the different modules. According to the defined roadmap, the first prototype was expected to contain dummy services and working on mock data. However, most of the services have already some implemented functionality on them. In general, the implementation of the first prototype revealed the basic problems that had to be overcome and helped to come to conclusions regarding the architecture of the platform and the structure of the messages that will enable the communication between the modules. During MS3, a first prototype that will use real services from WP3-WP6 is expected and will be described in D7.4.