SAFECARE

Integrated cyber-physical security for health services

Grant Agreement Number: 787002

Specifications about Data Exchange Layer

Deliverable 6.2

Author: CSI

Deliverable classification: PU

Contributors: MS, CCS, PEN, ISMB, ENC, CNAM, BEIA, KUL, ENC

Co-funded by the Horizon 2020 Framework Programme of the European Union

Version Control Sheet

Title	*Deliverable of Data Exchange Layer*
Prepared By	*CSI-Piemonte*
Approved By	
Version Number	*5*
Contact	

Revision History:

Version	Date	Summary of Changes	Initials	Changes Marked
V1	11/07/2019	Initial draft of the structure.	CSI	
V2	02/10/2019	Added Section 4 – Role of Data Exchange Layer. Updated publish-subscribe proposal, according to global architecture. Updated json structure of incidents. Added json structures of availabilities and responses. Changed proposal about check of data format and content.	CSI	
V3	11/10/2019	Added list of figures, list of acronyms, references and conclusions. Updated sections 6,7 and 8.	CSI	
V4	15/10/2019	Minor changes in the chapter 6 “Store and extract data from the Central Database”.	CSI	
V5	24/10/2019	Changes to fit the notes and corrections from Enovacom and Philips.	CSI	

The research leading to these results has received funding from the European Union's Horizon 2020 Research and Innovation Programme, under Grant Agreement no 787002.

Contents

1 The SAFECARE Project 5
2 Executive Summary 5
3 Introduction 5
4 Role of Data Exchange Layer 7
4.1 A focus on static data 8
4.2 Information flows through DXL 8
5 Publish-subscribe mechanism 10
5.1 Description of the solution 10
5.2 Proposal of implementation 12
5.2.1 MQTT implementation 12
5.2.2 Message implementation 14
6 Store and extract data from the Central Database 23
7 Coherence of data format and data content 24
8 Extraction of information by other components 24
9 Conclusions 25
References 25

List of figures

FIGURE 1: FLOWS OF INFORMATION THROUGH DXL 7
FIGURE 2: SEQUENCE DIAGRAM OF DATA FLOW RELATING TO IPM. 9
FIGURE 3: SEQUENCE DIAGRAM OF DATA FLOW RELATING TO TRAS. 9
FIGURE 4: SEQUENCE DIAGRAM OF DATA FLOW RELATING TO HAMS 10
FIGURE 5: STRUCTURE OF A MQTT MESSAGE 12
FIGURE 6: DIAGRAM OF THE PUBLISH-SUBSCRIBE MECHANISM FOR SAFECARE 14

List of acronyms

API	Application Programming Interface
DXL	Data Exchange Layer
MQTT	Message Queue Telemetry Transport
BTMS	Building Threat Monitoring System
CTMS	Cyber Threat Monitoring System
CDB	Central Database
IPM	Impact Propagation Module
HAMS	Hospital Availability Management System
TRAS	Threat Response and Alert System
MAS	Monitoring Alerting System
EDSA	E-health devices security analysis
QoS	Quality of Service

1 The SAFECARE Project

Over the last decade, the European Union has faced numerous threats that quickly increased in their magnitude, changing the lives, the habits and the fears of hundreds of millions of citizens. The sources of these threats have been heterogeneous, as well as weapons to impact the population. As Europeans, we know now that we must increase our awareness against these attacks that can strike the places we rely upon the most and destabilize our institutions remotely. Today, the lines between physical and cyber worlds are increasingly blurred. Nearly everything is connected to the Internet and if not, physical intrusion might rub out the barriers. Threats cannot be analyzed solely as physical or cyber, and therefore it is critical to develop an integrated approach in order to fight against such combination of threats. Health services are at the same time among the most critical infrastructures and the most vulnerable ones. They are widely relying on information systems to optimize organization and costs, whereas ethics and privacy constraints severely restrict security controls and thus increase vulnerability. The aim of this proposal is to provide solutions that will improve physical and cyber security in a seamless and cost-effective way. It will promote new technologies and novel approaches to enhance threat prevention, threat detection, incident response and mitigation of impacts. The project will also participate in increasing the compliance between security tools and European regulations about ethics and privacy for health services. Finally, project pilots will take place in the hospitals of Marseille, Turin and Amsterdam, involving security and health practitioners, in order to simulate attack scenarios in near-real conditions. These pilot sites will serve as reference examples to disseminate the results and find customers across Europe.

2 Executive Summary

The challenge of SAFECARE is to bring together the most advanced technologies from the physical and cyber security spheres to achieve a global optimum for systemic security and for the management of combined cyber and physical threats and incidents, their interconnections and potential cascading effects. The project focuses on health service infrastructures and works towards the creation of a comprehensive protection system, which will cover threat prevention, detection, response and, in case of failure, mitigation of impacts across infrastructures, populations and environment.

Over a 36-month time frame, the SAFECARE Consortium will design, test, validate and demonstrate 13 innovative elements, developed in the Document of Actions, which will optimize the protection of critical infrastructures under operational conditions. These elements are interactive, cooperative and complementary, aiming at maximizing the potential use of each individual element. The consortium will also engage with leading hospitals, national public health agencies and security Stakeholders across Europe to ensure that SAFECARE's global solution is flexible, scalable and adaptable to the operational needs of various hospitals across Europe and meet the requirements of newly emerging technologies and standards.

According to the general architecture of the system, the role of data exchange layer is to implement publish-subscribe mechanisms in order to trigger notifications to the other components when new physical and cyber incident or new impacts are sent.

3 Introduction

The deliverable D6.2 of WP6 is titled "Specification about data exchange layer". We report here the specification of the task.

This task consists in providing a data exchange layer and interfaces between the central database and relevant other components (e.g. Task 4.10 building monitoring system, Task 5.4 cyber threat monitoring system, Task 6.4 impact propagation model and Task 6.5 threat response and alert system). The data exchange layer will implement publish-subscribe mechanisms in order to trigger notifications to subscribers when new physical and cyber incident (coming from D4.10 building monitoring system and Task 5.5 cyber monitoring system) or new impacts (coming from Task 6.4 impact propagation model) are sent. The data exchange layer will be developed in order to store and extract data from the central database (e.g. web services). It will dynamically check the data format and data content before storage by checking static referential of data (e.g. list of critical assets, scale of impacts, names of rooms and buildings). The data exchange layer will allow others project components to extract added-value information on demand from the central database. For instance, in case of a dynamic incident, a notification will be sent to the impact propagation model (Task 6.4) in order to evaluate the impacts. Once impacts will be returned, a second notification will be sent to the threat response and alert system (Task T6.5) in order to trigger relevant responses.

The specification of the Data Exchange Layer (**DXL** in the following) can be summarized by four main features:

- providing publish-subscribe mechanism,
- giving possibility to store and extract data from the central database,
- checking coherence of data format and data content with static data,
- other components should extract information from the central database.

In the next sections we expose the solutions proposed by CSI-Piemonte for these features.

The systems involved in the DXL are:

- BTMS: Building Threat Monitoring System (T4.10),
- CTMS: Cyber Threat Monitoring System (T5.4),
- CDB: Central Database (T6.3),
- IPM: Impact Propagation Model (T6.4),
- TRAS: Threat Response and Alert System (T6.5),
- HAMS: Hospital Availability management System (T6.6),
- MAS: Monitoring Alerting System (T4.4)
- EDSA: E-health Devices Security Analytics (T5.4)

BTMS's and CTMS's role is to elaborate local "events" and "alerts" to determine which of them can be classified as "incident", thus they send incidents to the DXL.

CDB contains all the static data about health facilities and assets and receive and store all the dynamic data passing through the DXL, keeping an historical trace of the happening and making available them for the other components.

IPM, TRAS and HAMS are "decisional modules" which generate a response to the incident.

MAS is a mobile application developed for the management of the physical alerts. It allows a human user to report to BTMS security alerts that he notices and it receives security plans from decisional modules.

EDSA is a component that can detect security related issues on medical devices.

4 Role of Data Exchange Layer

According to the specification of the global architecture of SAFECARE (D6.1), DXL should manage six kinds of messages:

- **Incidents**: generated by BTMS and CTMS and sent to IPM, HAMS and CDB.
  They provide information about a risk event (details provided in Section 5.2.2 and Documents D4.9 and D5.9).
- **Impacts**: generated by IPM and sent to MAS, TRAS, HAMS, EDSA, BTMS, CTMS and CDB.
  They inform about which assets are threatened by a single incident and with which degree of severity (details provided in Section 5.2.2 and Document D6.6).
- **Responses**: generated by TRAS and sent to CDB.
  They indicate an overall result of the reaction plan to an incident (i.e. calls / SMS, …) and the communication details (details provided in Section 5.2.2 and Document D6.8).
- **Notifications**: generated by TRAS and sent to MAS and CDB; and generated by MAS and sent to TRAS and CDB.
  They establish the communication between TRAS and MAS (details provided in Section 5.2.2 and Document D6.8).
- **Availabilities**: generated by HAMS and sent to CDB.
  They communicate the availability of assets involved in the incident (details provided in Section 5.2.2 and Document D6.10).
- **Static data**: stored in CDB and exposed to BTMS, CTMS, IPM, TRAS and HAMS.
  They are a different kind of data respect to the previous ones. They are not messages dynamically generated and sent but they are charged una tantum in the CDB. They contain information about the assets, locations and sensors present in the hospital, these data are available for analysis by other components and they are exposed through REST API (details provided in section 8 and document D6.4).

In short, the exchange of information between components through DXL can be summarize in this schema.

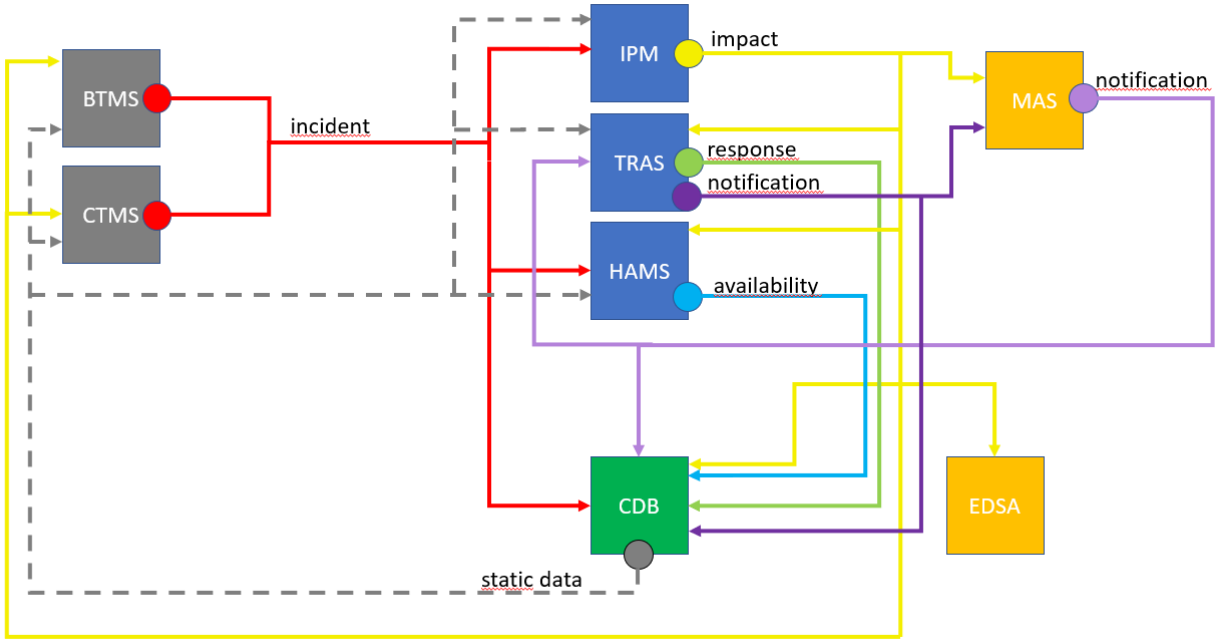

Figure 1: Flows of information through DXL

4.1 A focus on static data

Static data contain a census of all the medical devices, cyber and physical security controls, part of physical security, rooms, buildings, computers, networks, servers, etc. In short, everything contained in a hospital that can be relevantly involved in a security event. All these things are generically called “asset”.

All the assets involved in the project will be registered in CDB, and for each of them is assigned an ID code valid in the whole SAFECARE system, this code is not related to the fabric ID of the asset nor with the ID assigned by the hospital. From the moment of storage in CDB, in all messages sent through DXL every asset will refer only to the “SAFECARE ID”. Components like BTMS or CTMS, that manage directly assets and sensors, should have conversion tables in order to compose the messages with the correct ID code.

In a nutshell, every component, before subscribing a message to DXL, must check that the message contains the ID code according to CDB.

Static data of this type are registered in CDB by the medical centre of reference with some information useful to identify them, such as type of asset, model, location and eventually IP address. Other data, like vulnerabilities and functionalities of the specific asset, cannot be managed by CDB, thus they are not present. Decisional modules, like IPM, that could need this kind of data should save them inside their own module to be able to access them and process messages.

Notice that the SAFECARE solution it is not a single system for all hospitals, but it will be deployed locally in three releases, one in Turin for ASLTO5, one in Marseille for AP-HM and one in Amsterdam for AMC. So static data charged on CDB are different depending on the release: CDB in Turin will manage static data charged from ASLTO5, and so on.

In the CDB there will be also another kind of static data, consisting in some decoding tables that define allowed values for some SAFECARE entities such as threats, types of risk, severities. These tables will not be managed by the medical centre of reference but they will be proposed in the development of CDB and validated from the SAFECARE partners.

4.2 Information flows through DXL

Information exchanged by DXL should follow a precise flow.

The ideal flow of data should be the following:

The generation of an incident by BTMS or CTMS start the flow: incident is sent to IPM and HAMS and it is stored in CDB.

IPM receives the incidents, requires static data from CDB, and, using its correlation rules, elaborates an impact, that is sent to all other components and stored in CDB.

TRAS receives the impact, and, using its logic rules, generates a response (stored in CDB) and a notification (sent to MAS). In the meantime HAMS, using data from the incident, the impact and the static data from CDB, elaborates an availability message, that is stored in the CDB.

To be more clear in the following we report the sequence diagrams relating to each of the decisional modules.

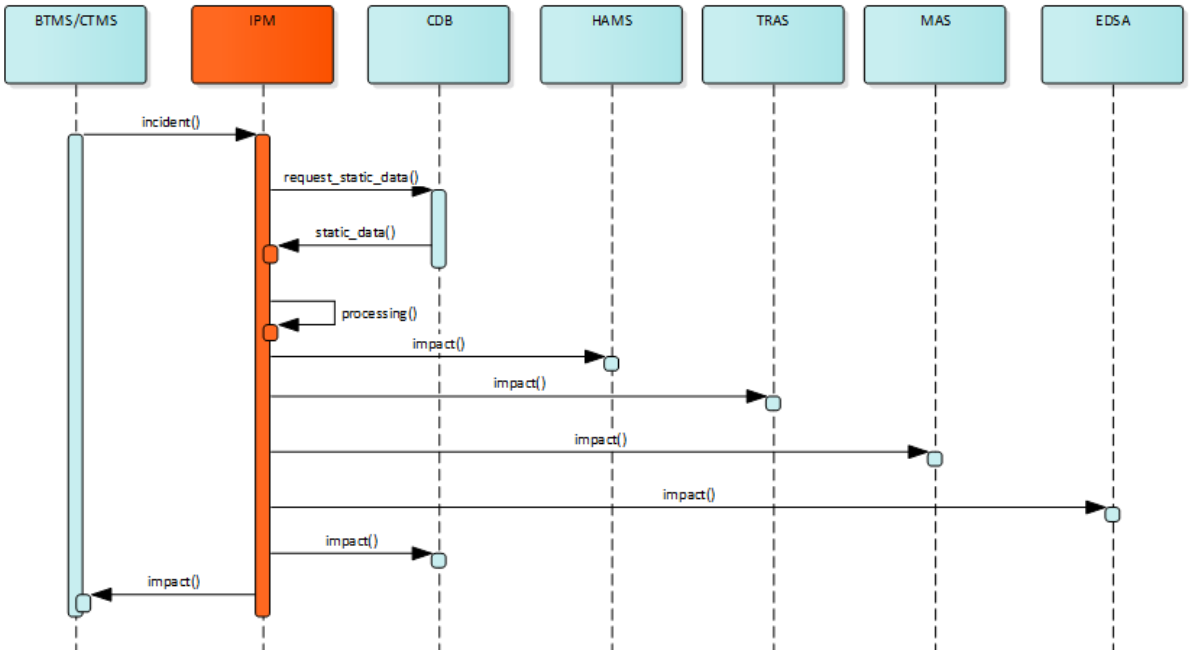

Figure 2: Sequence diagram of data flow relating to IPM.

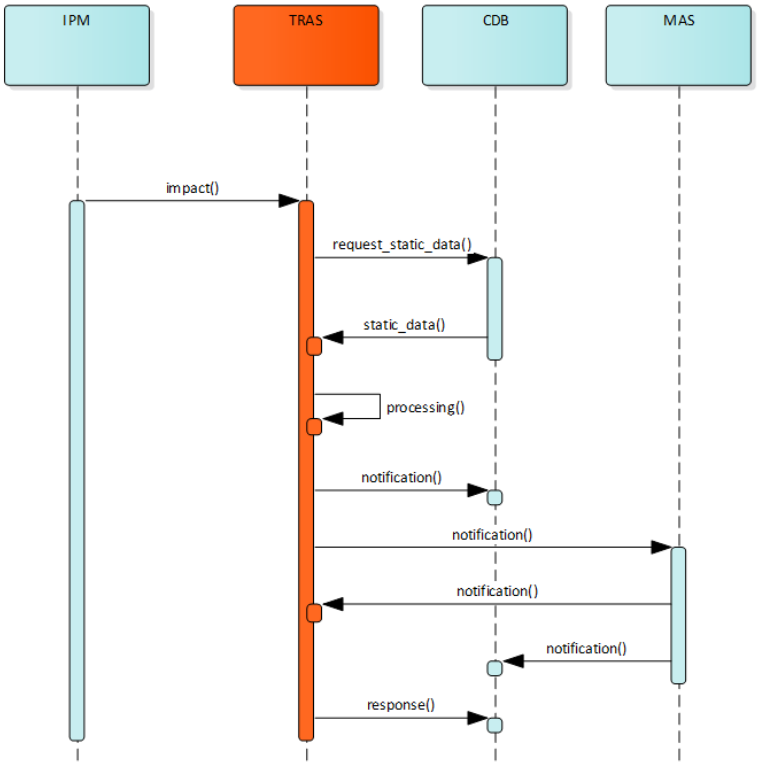

Figure 3: Sequence diagram of data flow relating to TRAS.

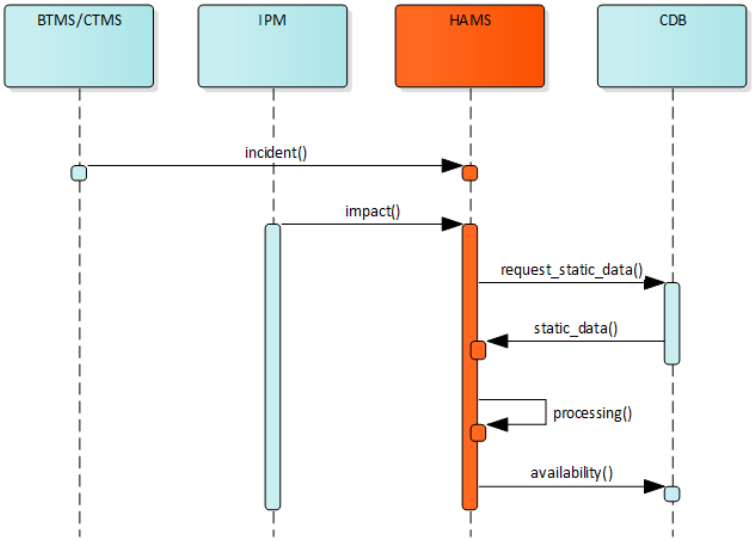

Figure 4: Sequence diagram of data flow relating to HAMS

5 Publish-subscribe mechanism

In this section we will present both aspects, theorical and which product and technological approach we propose for implementing the publish-subscribe mechanism of DXL.

5.1 Description of the solution

Considering that the DXL will provide a machine-to-machine communication mechanism, and it has to be of the kind "publish-subscribe" according to Feature 1, we propose to adopt one of the most well-known communication protocols: **MQTT.**

MQTT (Message Queue Telemetry Transport) became an ISO standard for publish-subscribe light messaging. It is positioned directly above the TCP/IP protocols and has been designed to solve all those situations where low impact may be required and data exchange may occur in the presence of limited bandwidth.

The MQTT protocol is based on the principle of publishing messages and subscribing to topics. Subscriber subscribes to particular topics which are relate to them and by that receive every message which are published to those topics. Alternatively, clients can publish messages to topics, thus making them available to all subscribers to those topics.

MQTT defines three levels of Quality of Service (QoS). Messages may be sent at any QoS level, and clients may attempt to subscribe to topics at any QoS level. This means that the client chooses the maximum QoS it will receive.

- QoS0, the message is sent at most once and it does not provide guarantee delivery of a message
- QoS1, the data is sent at least once and it is possible to deliver a message more than once by setting the value of duplicate flag by 1
- QoS2, the message is sent exactly once by using 4-way handshaking

In case of unexpected disconnection, client informs the server of holding will or message, which will be published to relevant topics. Useful in security scenarios in which system managers should know as early as possible when a sensor device lost connection with network.

A MQTT broker is used to manage the message queue and the subscriptions of the clients to the topic. There exist many MQTT broker, with different features.

The topic consists of one or more topic levels. Each topic level is separated by a forward slash (topic level separator). Each topic must contain at least 1 character and the topic string permits empty spaces. Topics are case-sensitive.

When a client subscribes to a topic, it can subscribe to the exact topic of a published message or it can use wildcards to subscribe to multiple topics simultaneously. A wildcard can only be used to subscribe to topics, not to publish a message.

A single-level wildcard replaces one topic level. The "+" symbol represents a single-level wildcard in a topic. Any topic matches a topic with single-level wildcard if it contains an arbitrary string instead of the wildcard.

For example the string:

```
myhome/groundfloor/+/temperature
```

matches the following topics:

```
myhome/groundfloor/livingroom/temperature
myhome/groundfloor/kitchen/temperature
```

but it doesn't match the following topics:

```
myhome/groundfloor/kitchen/brightness
myhome/firstfloor/kitchen/temperature
myhome/groundfloor/kitchen/fridge/temperature
```

The multi-level wildcard covers many topic levels. The "#" symbol represents the multi-level wild card in the topic. For the broker to determine which topics match, the multi-level wildcard must be placed as the last character in the topic and preceded by a forward slash. When a client subscribes to a topic with a multi-level wildcard, it receives all messages of a topic that begins with the pattern before the wildcard character, no matter how long or deep the topic is.

For example the string:

```
myhome/groundfloor/#
```

matches the following topics:

```
myhome/groundfloor/livingroom/temperature
myhome/groundfloor/kitchen/temperature
myhome/groundfloor/kitchen/brightness
```

but it doesn't match the following topic:

```
myhome/firstfloor/kitchen/temperature
```

A MQTT message has a structure illustrated in the following picture:

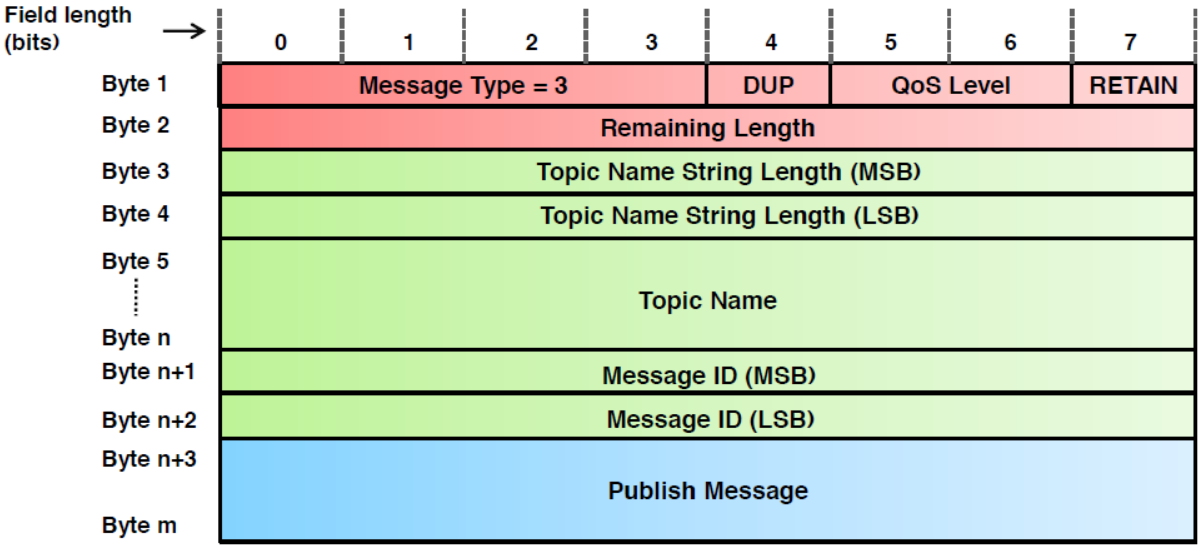

Figure 5: Structure of a MQTT message

There is a fixed header of two bytes that provides information about the characteristics of the message, we can find for example the QoS level. The "remaining length" contains the number of bytes that remain in the message (in this case *m-2*), this is used as a control to know where the message ends exactly.

In the rest of the message there are two bytes that indicate the string length of the topic name, then the space dedicated to the topic name, then two bytes for the message ID and the rest of the bytes contains the effective message (payload).

MQTT messages have a maximum size that corresponds to a payload of 256 MB. It is plenty sufficient to support messages involved in SAFECARE project so the system does not risk to overflow.

5.2 Proposal of implementation

5.2.1 MQTT implementation

The broker proposed for the implementation of MQTT system is Apache ActiveMQ, an open source, multi-protocol, Java-based messaging server. It can manage a MQTT message queue and it supports MQTT v3.1.1, so this will be the version of the protocol used in SAFECARE.

This broker will receive MQTT messages from all subscribers and it will submit them to subscribers.

MQTT messages are simple text string, without a format, but for the nature of the messages managed by the DXL the idea is to structure them as a **.json file**, since it will be easier for the other modules to interpret them.

We propose the following stratification of topics for the MQTT broker:

- `safecare`: this is the root of all the messages sent to the broker, no matter who is sending them or what they contain.
- `incident`: applied directly over the `safecare` topic, it identifies the incident messages, that are sent by BTMS and CTMS.
- `physical`, `cyber`: applied over `incident` topic, they specify the source and typology of the message

- `effect`: applied directly over the `safecare` topic, it identifies the messages of response given by the decisional modules IPM, TRAS and HAMS.
- `impact`, `response`, `availability`, `notification`: applied over `effect` topic, they specify the source and typology of the message.

So, a hypothesis of publish subscribe configuration should be the following:

BTMS:

- Publish: `safecare/incident/physical`
- Subscribe to: `safecare/effect/impact`

CTMS:

- Publish: `safecare/incident/cyber`
- Subscribe to: `safecare/effect/impact`

CDB:

- Publish:
- Subscribe to: `safecare/#`

IPM:

- Publish: `safecare/effect/impact`
- Subscribe to: `safecare/incident/#`

TRAS:

- Publish: `safecare/effect/response`
  `safecare/effect/notification`
- Subscribe to: `safecare/effect/impact`
  `safecare/effect/notification`

HAMS:

- Publish: `safecare/effect/availability`
- Subscribe to: `safecare/incident/#`
  `safecare/effect/impact`

MAS:

- Publish: `safecare/effect/notification`
- Subscribe to: `safecare/effect/impact`
  `safecare/effect/notification`

EDSA:

- Publish:
- Subscribe to: `safecare/effect/impact`

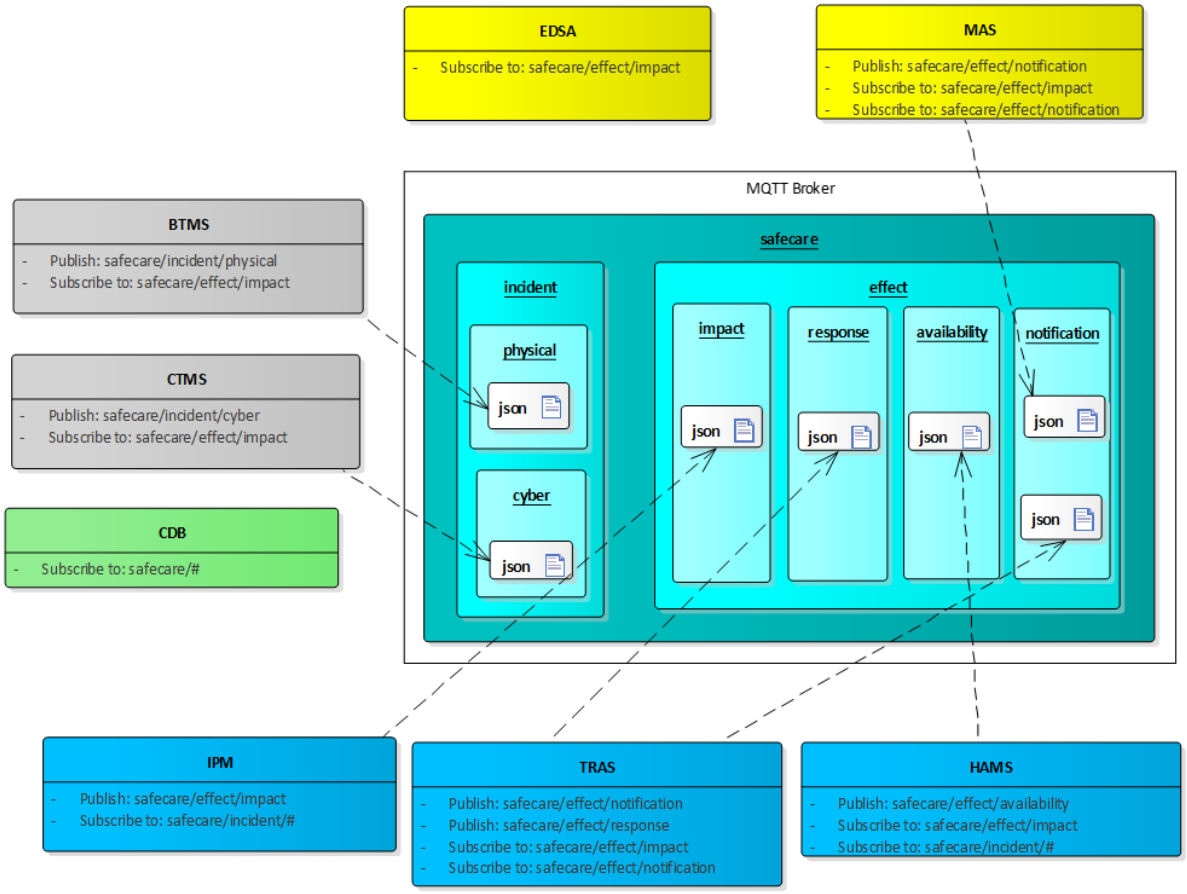

Figure 6: Diagram of the publish-subscribe mechanism for SAFECARE

5.2.2 Message implementation

Messages sent via data exchange layer with MQTT protocol have to follow a standard, this allows modules that receive them to analyse them easily. We illustrate here the proposal for standard composition of the different kind of messages.

Firstly, we consider that every message exchanged through DXL must have an ID code, that is unique for typology of message (there are not two incidents with the same ID, and the same for impacts and so on).

The generation of these IDs can be managed by the component that creates the message simply with a sequential string, because each message is generated from only one component, so there is not the risk of having two messages with the same ID. Separate speech is needed by incidents, that are generated both by BTMS and CTMS, but the problem can be avoided laying that the ID of incidents from BTMS must begin with a "P" (physical) and the ones from CTMS with a "C" (cyber).

Physical and cyber incident

Since SAFECARE modules involved in WP4 also use json format for data exchange inside BTMS logics, the json structure of the incident message is already proposed in document D4.3. Here we propose an alternative structure, that is almost the same, except for some data that are of course useful for BTMS communications but results redundant in a higher level, so they were removed in order to keep data clean and to adapt structure to fit not only physical incidents, but also cyber

ones. It consists in the following .json file (this example contains two alert and fields are filled with generical data, they do not refer to a real incident):


```
{
	"detector":"WP4",
	"severity":"HIGH",
	"date":"20190410T165514Z",
	"unique_identifier":"A#85647",
	"alerts": [
		{
			"id_alert_type":"XXX",
			"id_detector":"AAA",
			"date":"20190410T165514Z",
			"description":"Suspicious behaviour: loitering",
			"unique_identifier":"A#85648",
			"assets":{
				"id_asset":"XXX"
			},
			"sensor":{
				"id_asset":"XXX",
				"sensor_metadata":{
					"VideoAnalitics":{
						"NumberOfPeople": "2",
						"SecurityEvent" : "loitering",
						"TimeSpent": "180s",
						"mediaPhoto":{},
						"mediaVideo":{
							"uri":"rtsp://192/media.amp"
						}
					}
				}
			}
		},
		{
```

```
                "id_alert_type":"YYY",
                "id_detector":"BBB",
                "date":"20190410T165514Z",
                "description":"Fire detection system: Fire alarm",
                "unique_identifier":"A#45678",
                "assets":{
                      "id_asset":"YYY"
                },
                "sensor":{
                      "id_asset":"YYY",
                      "sensor_metadata":{
                            "VideoAnalitics":{
                                  "CameraId ": "a12",
                                  "mediaVideo":{
                                        "uri":"rtsp://192/media2.amp"
                                  }
                            }
                      }
                }
          }
    ]
}
```


"detector" (obligatory) indicates if the incident is coming from BTMS ("WP4") or CTMS ("WP5").

"severity" (optional) indicates a first estimate for the severity of the incident, provided by who validated the event as incident.

"date" (obligatory) refers to the moment of generation of the incident.

"unique_identifier" (obligatory) is an ID code of the incident useful to refer to it without misunderstanding. The code is unique for all the incidents. . ID code of physical incidents must begin with a "P" character, ID code of cyber incidents must begin with a "C" character.

"alerts" (obligatory) contains an array with the description of all the alerts that determined the incident.

"id_alert_type" (obligatory) chooses from a finite list of possible kinds of incident the one which matches the most what is happening.

"id_detector" (obligatory) indicates which component of SAFECARE detected the alert.

"date" (obligatory) refers to the moment of the detection of the alert.

"description" (optional) contains a free text in charge of a human user who wants to provide a specific description of the incident.

"unique_identifier" (obligatory) is an ID code of the alert useful to refer to it without misunderstanding.

"assets" (obligatory) contains a string with the assets involved in the alert.

"sensor" (obligatory) contains information about the instrument that detected the alert (for example a camera, the fire detection system, a human through the mobile app, an antivirus or a firewall).

"sensor_metadata" (optional) contains additional details about sensor.

The differences with .json file proposed in Document D4.3 are the following:

- Fields "created_timestamp", "confirmed_timestamp" and "start_date" are replaced with a single field "date".
- Field "type" are removed because in a incident message the only "type" accepted is "INCIDENT", messages of the type "ALERT" and "EVENT" are not published on DXL.
- Field "events" is changed in "alerts", because an incident can be composed only by alerts, other events are not of interest of decisional modules. For the same reason field "type" of the event is removed.
- "category" and "name" of the field "assets" are replaced simply by "id_asset", all the other information about assets are available as static data in CDB. For the same reason field "location" is removed.
- In field "sensor" there are not information about type but only the "id_asset" (other information about sensors are available as static data in CDB) and a free "sensor_metadata", that collects additional data about the detection.

Impact

An impact consists of a list of assets threatened by a single incident. The list is compiled by the IPM every time that it receives an incident from the DXL. As an output of the decisional algorithm it sends back a .json file called "impact".

```
{
      "impact_id":"XXXXXXX",
      "incident_id":"AAAAAAA",
      "assets":[
            {
                  "asset_id":"AAAAAAA",
                  "risk_type":"Fire",
                  "impact_score":1
            },
            {
```

```
                "asset_id":"AAAAAAB",
                "risk_type":"Fire",
                "impact_score":0.8
        },
        {
                "asset_id":"AAAAAAC",
                "risk_type":"Data leak",
                "impact_score":0.6
        }
    ]
}
```


In the message is reported the ID of the incident that generated the impact and the list of assets involved, for each of them is specified the risk type (chosen from a list of possible types) and an impact score between 0 (the asset is not involved) and 1 (the asset is totally and surely involved with high potential damage).

Response

A response message informs how to react to an incident, in particular which are the emergency communication to perform. It reports the ID code of the incident that triggered itself and the list of communications, with their status and all details. Its schema is the following.


```
{
    "response_id":"0000000",
    "originate_impact_id":"XXXXXXX",
    "timestamp":1567432951,
    "response_status":"executed and accepted",
    "communication_details": {
            "template": {
              "name": "call",
              "id": "984a79bd635fcb032d3bf0",
              "config": {
                "recipients": {
                  "source": "field",
                  "value": "@recipients",
```

```
        "medias": [
          "personal",
          "professional",
          "mobile"
        ]
      },
      "quorum_policy": {
        "method": "rate",
        "value": 100
      },
      "retry_policy": {
        "maximum": 3,
        "delay": 60,
        "feedbacks": [
          "unknown"
        ]
      },
      "blocking": true,
      "template": {
        "id": "df160101-710c-4be6-9a18-0b4fc782e476",
        "draft": false
      }
    },
    "timeout": 10800,
    "topics": [],
    "title": "Campagne d'appels"
  },
  "id": "9c53d2d5-cabf-4bf0-ae8d-28622427ef0e",
  "start": "2019-10-17T09:35:50.125000+00:00",
  "end": "2019-10-17T09:36:42.648000+00:00",
  "state": "finished",
  "reporting": {
    "quorum": {
```

```
    "reached": false,
    "expected": 3,
    "progress": 1
  },
  "feedbacks": {
    "expected": 3,
    "progress": 3,
    "received": {
      "positive": 1,
      "negative": 2,
      "unknown": 0
    },
    "failed": 0
  },
  "contacts": [
    {
      "uid": "73hFaL0OODJAforMuJzEFJ",
      "external": false,
      "display_name": "Jean-Pierre Facque",
      "start": "2019-10-17T09:35:50.125119+00:00",
      "end": "2019-10-17T09:36:33.381040+00:00",
      "feedback": "negative",
      "attempts": 1,
      "calls": [
        {
          "uid": "702bcc10",
          "media": "mobile",
          "number": "+33xxxxxxxx",
          "attempt": 1,
          "feedback": "negative",
          "status": "completed",
          "workflow": "f6ab92eb-f34e-4550-87ad-261104a75f59",
          "start": "2019-10-17T09:35:50.326957+00:00",
```

```
        "end": "2019-10-17T09:36:33.380436+00:00"
      }
    ]
  },
  {
    "uid": "2FQsFeXBCBTtC1y49ypu7W",
    "external": false,
    "display_name": "David Fermet",
    "start": "2019-10-17T09:35:50.125119+00:00",
    "end": "2019-10-17T09:36:42.648143+00:00",
    "feedback": "negative",
    "attempts": 1,
    "calls": [
      {
        "uid": "c4e24de0",
        "media": "mobile",
        "number": "+33xxxxxxxx",
        "attempt": 1,
        "feedback": "negative",
        "status": "completed",
        "workflow": "c8221599-27e2-4efb-b31f-1eb58281c476",
        "start": "2019-10-17T09:35:50.457911+00:00",
        "end": "2019-10-17T09:36:42.647646+00:00"
      }
    ]
  },
  {
    "uid": "4Fo52KcW0zb4CucyS2et2e",
    "external": false,
    "display_name": "Christophe Le Dantec",
    "start": "2019-10-17T09:35:50.125119+00:00",
    "end": "2019-10-17T09:36:24.880801+00:00",
    "feedback": "positive",
```

```
                    "attempts": 1,
                    "calls": [
                      {
                        "uid": "dca7508a",
                        "media": "professional",
                        "number": "+33xxxxxxxx",
                        "attempt": 1,
                        "feedback": "positive",
                        "status": "completed",
                        "workflow": "06ae4551-a2a0-4892-9d5d-c729e45bd0f8",
                        "start": "2019-10-17T09:35:50.671281+00:00",
                        "end": "2019-10-17T09:36:24.880080+00:00"
                      }
                    ]
                  }
                ]
              }
            }
}
```


This sample output may be subject to changes.

Notification

Notification informs MAS about which emergency communications have to be performed according to the response plan. MAS then send another notification to TRAS informing of the status of the communication. An example of the .json file is the following.

```
{
      some communication IDs and references,
      timeout (max time to answer): some value,
      recipient : {
            “user1” : “user address 1”,
            “delivery_method”: “confirmation required”,
            “delivery status” : “message received”,
```

```
            “message” : “an example of alert message”
      },
}
```

Availability

An availability message contains a list of “resources” identified by an “id”. For each resource is indicated a status that can be “available” or “not available” and for some kinds of resource can be reported additional information. An example of the .json file is the following.

```
{
      “RequestID”: “abcd1234”,
      “timestamp”: 1567432951,
      “resources”: [
            {
                  “id”: “0000001”,
                  “type”: “asset”,
                  “status”: “available”
            },
            {
                  “id”: “0000005”,
                  “type”: “department”,
                  “status”: “available”,
                  “bed_availability”: 10,
                  “staff_availability”: 5
            }
      ]
}
```

6 Store and extract data from the Central Database

For the management of the static data in the Central Database, CSI has evaluated if to take advantage also of a specific software named openMAINT. Nevertheless, we should “wait and see” the availability of the data coming from the local hospital structures, in order to validate the adoption of such software, together with the central database: an open source software, thought for property and facility management, in particular for the management of buildings, installations and assets. It provides a ready-to-use configurable structure, where it is possible to insert all the physical asset that the project intends to manage.

This solution would represent part of the static part of the central database allowing to store and extract static data. Details about the structure of openMAINT are available in Document D6.4.

For the remaining part of the static data, mainly decoding tables of threats, severities, adversaries and other SAFECARE entities, specific tables will be designed and managed through one-shot data ingestion or exposed through REST API. In general access to the CDB will be mediated using REST API.

The update of the static data will not be performed by DXL.

OpenMAINT will empower the end-users to manage the lifecycle of the assets. The REST APIs will allow other components to query any updates of CDB.

Dynamic data coming from DXL will be represented as json file and stored in CDB for historical purposes, since all the actors will already get this information from DXL.

7 Coherence of data format and data content

MQTT does not force to adopt any specific format, but messages are managed simply as a string of bytes. Publisher and subscribers will have the responsibility to agree on a common format conforming to a json schema, as described in Section 5.2.2.

The specification of the task in Section 3 suggests a control of the coherence of data format and data content; since the DXL is just a carrier for the data, its validation has to be done by the publisher. Because of the nature of the publish-subscribe mechanism, that sends messages to all subscribers at the same time. Adding a control in the middle of the publish-subscribe would mean expecting an additional “object” inserted in the DXL that receives all messages before others, performs the control and then re-publishes all messages to forward them to other components. This solution would compromise the real-time exchange of information and the efficiency of the entire system.

For this reason these validations should be performed before the input of messages in the DXL system, delegating them to the components that generate messages. Data format should be checked by the system that compose the structure of the message, as the coherence of data content.

8 Extraction of information by other components

The software we talked about in Section 6, openMAINT, has the possibility to expose some APIs (Application Programming Interfaces), that allow other components to access some data in the Central Database without querying directly on the tables. Using these APIs “decisional modules” will be able to extract structured static data from the CDB.

The CDB will not be accessible for direct queries, SAFECARE components will get data only through REST APIs.

We will provide all details about the design APIs and their use in future release of this document.

For the moment we consider that APIs could access only static data in the CDB, dynamic data are stored but they will be kept only for historical purposes since there is no evidences that any components required to access them.

9 Conclusions

Data Exchange Layer is a central component of the SAFECARE project, that will interconnect almost all the other modules. For this reason it must work efficiently and it has to match all the needs of other partners. The architecture wants to be as simple as possible but at the same time flexible in order to manage different kinds of inputs from different components.

A close collaboration between partners allowed the realization of this document and, since specification of some other components is not totally available yet, it is possible that some formal details have to be modified to fix needs of the global prototype.

This document assumes its full meaning if coupled with Document D6.4, the specification of Central Database, in particular for what concerns management, reading and updating of static data.

References

1. **Egli, Peter R.** [Online] https://www.slideshare.net/PeterREgli/mq-telemetry-transport.

2. **http://mqtt.org/. [Online]**

3. CSI. ***D6.4 Specification of the central database.*** **s.l. : SAFECARE document.**

4. F. Lubrano, G. Varavallo. ***MQTT description.*** **s.l. : SAFECARE document.**

5. LINKS. ***D6.10 Specification of the HAMS.*** **s.l. : SAFECARE document.**

6. [Online] http://www.openmaint.org/it.

7. MS. ***D4.3 Specification of the intrusion detection system.*** **s.l. : SAFECARE document.**