This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 607881

Petty cRiminality diminution through sEarch and Analysis in multi-source video Capturing and archiving plaTform

P-REACT

Instrument:	Research and Innovation Action
Thematic Priority:	FP7- SEC - 2013.7.2-1
Grant Agreement:	607881

P-REACT Conceptual Architecture

Deliverable Number	D2.3
Title	P-REACT Conceptual Architecture
Version	3.0
Date	20th July 2015
Status	Revision
Dissemination Level	PU (Public)
Nature	Report

EC Distribution

Project Partners:		Vicomtech-IK4 (VICOM); Kinesense (KS); Aditess (ADI); Future Intelligence (FINT); Center for Research and Technology Hellas (CERTH); Center for Security Studies (KEMEA); Societa Reti e Mobilita (SRM)
Contributors:	Main:	Georgios Stavropoulos – CERTH
	Others:	Dimitrios Tzovaras - CERTH Marcos Nieto – VICOMTECH Anargyros Sideris – FINT Nectarios Efstathiou – ADITESS Georgios Kioumourtzis – KEMEA Prokopios Drogaris - KEMEA Daniel Keenaghan – KS Liza Charalambous – ADITESS Nikolaos Koutras – ADITESS Dimosthenis Ioannidis - CERTH Nikolaos Porfyriou - CERTH

Every effort has been made to ensure that all statements and information contained herein are accurate; however the Partners accept no liability for any error or omission in the same.

Document Control

Version	Date	Author	Modifications
0.1	16/09/2014	Georgios Stavropoulos	Initial Version - TOC
0.2	3/11/2014	Anargyros Sideris	FINT Contribution
0.3	11/11/2014	Marcos Nieto	VICOM Contribution
0.4	5/12/2014	Liza Charalambous, Nectarios Efstathiou, Nikolaos Koutras	ADITESS Contribution
0.5	11/12/2014	Daniel Keenaghan	KS Contribution
1	07/01/2015	Georgios Stavropoulos	First Consolidated Version

1.1	19/01/2015	Anargyros Sideris	Updated FINT's contribution
1.2	20/01/2015	Prokopios Drogaris	Provided Section on Privacy, Stigmatization etc.
1.3	23/01/2015	Mauro Borioni	Internal Review
1.5	02/02/2015	Marcos Nieto	VICOM response to Review comments
1.5	02/02/2015	Lisa Charalambous	ADITESS response to Review comments
1.5	04/02/2015	Anargyros Sideris	FINT response to Review comments
1.5	04/02/2015	Daniel Keenaghan	KS response to Review comments
2.0	05/02/2015	Georgios Stavropoulos	Final Version
2.1	18/06/2015	Liza Charalambous	Revised Audio analytics sections
2.5	29/06/2015	Anargyros Sideris	Revised Figures
2.6	30/06/2015	Georgios Stavropoulos	Pre-Final Revised Version
2.7	07/07/2015	Sarah Doyle	Internal Review
2.9	14/07/2015	Georgios Stavropoulos	Addressed comments of Internal Review
3.0	20/07/2015	Georgios Stavropoulos	Final Version. Added Section 1.1 with surveillance SoA, moved code to appendix, updated Video and Depth analytics, included list of developed modules in sections 3.1.3 and 3.3.3 for embedded and cloud side respectively.

List of abbreviations

Abbreviation	**Definition**
URI	Uniform Resource Identifier
JSON	JavaScript Object Notation
GUI	Graphical User Interface
XML	Extensible Mark-up Language
SOAP	Simple Object Access Protocol
UDP	User Datagram Protocol
UI	User Interface
DB	Data Base
GUID	Globally Unique IDentifier
RGB-D	RGB (Red-Green-Blue) images along with Depth information
IP	Internet Protocol

Executive Summary

This deliverable presents the results of Work Package 2 –System Architecture of the P-REACT project, carried out under the European Union (EU) 7th Framework Program (FP7). More specifically, this report outlines the conceptual architecture of the system, in order to provide a holistic view on the architecture, its building blocks, components, interdependencies among components and related constraints such as development methodology. It also describes in detail the functional and technical specifications of the components that comprise the P-React system architecture, defining the structural, development, deployment and dynamic view of the system. The methodology followed in this deliverable towards this

direction is presented in Section 1.

Initially, the conceptual architecture of the P-React system is presented (SectionSection 2). This is a high-level view on the overall architecture, describing the two major components of the P-React system, the embedded system and the cloud, along with their fundamental building blocks (modules).

Then, the structural view of the system is defined (Section 3), presenting the different architectural elements that deliver the system's functionality. This view provides the system's decomposition into different components, demonstrating the responsibilities and functionalities of each of them.

In Section 4, the development view is presented, describing how the architecture supports the development process. Here, aspects like components technical requirements and dependencies, programming technologies and use of existing software are covered.

The deployment view, described in Section 5, defines the physical environment in which the system is intended to run, including required hardware environment (e.g. processing nodes, network interconnections, etc.), technical environment requirements for each node, mapping of software elements to the runtime environment, third-party software requirements and network requirements.

Focus is given to the analysis of the dynamic behaviour of the system, in Section 6, where the actual use cases and scenarios are correlated with each component and the way that each component acts within them. The functionality of the system is described in detail though UML use case and sequence diagrams.

A detailed specification for all the core architectural elements is given in Section 7, in order to provide a deep insight on the P-REACT system and to guarantee seamless interoperability among the components. Focus is given on the internals of the components, the interfaces, inputs/outputs and data types.

Finally, it should be noticed that although this report is delivered according to the P-React DoW in Month 10, the architectural elements and their detailed specifications will remain an open issue till all components and subsystems are built and all modules have been integrated to the P-REACT system. Thus, this document can be considered as a living document.

Table of contents

Annexes

Tables

Figures

1. Introduction

The DoW describes this deliverable as:

D2.3 - P-REACT Conceptual Architecture: The P-REACT Conceptual Architecture will include functional, technical, interoperability and large scale deployment specifications. Detailed view of the architectural elements comprising P-REACT end-to-end system. Provisional UML diagrams outlining the dynamic and static behaviour of the system. Moreover, the risks of stigmatization, discrimination and the adequacy of measures will be addressed.

The aim of this document is to present the P-REACT conceptual system architecture. The architecture of the various components is presented, along with the functional, development and deployment views. Also the static and dynamic model analysis of the system is presented.

The document is organized as follows:

1. Section 2 presents the conceptual architecture of the system.
2. Section 3 describes the functional view of the various components
3. In Section 4 the development view is presented.
4. Section 5 describes the requirements for the deployment of the system
5. Section 6 presents the static and dynamic model analysis
6. Section 7 describes the specifications of the various architectural elements
7. And Section 8 concludes the present document

In the next sub-section, a short state of the art of the current video surveillance systems is presented, along with the P-REACT's proposed system advantages.

1.1. Video Surveillance State of the Art

Traditionally video surveillance systems use a network of IP cameras connected to a monitoring centre, where the various video feeds are watched by security or law enforcement personnel. While such systems are widely used, to date limited usage has been made of analytics to automate the monitoring process. A typical example is the Operation Virtual Shield (OVS) [1], [2], [3] program which deploys a network of thousands of cameras in a metropolitan area. The OVS system processes camera feeds in real time and is able to detect dangerous activities and perform facial recognition per operator request. A related project is Combat Zones That See (CTS) funded by United States Defence Advanced Research Projects Agency (DARPA) [4]. At its core CTS incorporates technology that can identify and track objects of interest (pedestrians, vehicles etc.). Similar initiatives are currently developed and deployed in China [5] and UK [6].

In a smaller scale, video surveillance systems for domestic or business usage are also commercially available. Such installations are usually smaller and simpler compared to the systems of the previous paragraphs. There are several companies that are active in this sector including among others ADT[7], Tyco Integrated Security[8], CPI Security[9], Lorex [10], Q-See [11]and Guardian Protection Services[12]. The solutions offered by these companies include IP cameras (wired or wireless) with video feeds stored in a centralized or cloud-based server. Most of them allow remote access to the cameras using a mobile device or remote PC. The cameras used may include night vision (low-light or thermal technology) and incorporate some basic algorithm for motion detection. If an abnormal event is detected, a notification is sent to the operator of the system either as an email notification or as a video clip.

One can easily identify a series of shortcomings in the available systems. First of all, typical surveillance systems rely on basic video analytics such as motion detection. Recent advances in computer vision and signal processing have created algorithms for background subtraction [13], [14], human detection and tracking [15], [16], [17], activity recognition [18], [19] and human re-identification[20], [21] with improved precision and recall in realistic conditions (e.g. dynamic scene, illumination variations) that are mature enough for commercial deployment. P-REACT's approach is to insert an intermediate low-cost embedded device that will be responsible for locally analysing camera feeds in real time using advanced audio and video analytics algorithms. Indeed the use of audio analytics is uncommon and considered pioneering especially when combined with video analytics. Regarding the archiving of video data, P-REACT will offer a cloud storage solution, enabling the user to perform content-based queries (e.g. re-identification queries, text-based queries using video tagging etc.). The overall architecture is highly modular as additional embedded systems, cameras and cloud resources will be seamlessly added on demand. Currently, several established commercial solutions cannot scale having an upper limit on the number of cameras they can manage and on the volume of video content manage whereas P-REACT can handle new video feeds by dynamically allocating additional cloud resources Furthermore, as the entire system uses open source software and low-cost devices, one can expect that its overall cost will be lower compared to existing solutions. Indicatively, P-REACT advocates the use of Linux over a proprietary operating system and uses depth cameras instead of thermal cameras that are significantly more expensive. Moreover, since video content will be sent to the cloud only if a certain activity is detected network resources utilization will be more efficient.

2. Conceptual Architecture

This Section describes the high-level view on the P-REACT system architecture, the distinction between the Embedded System and the Cloud, while describing the major building blocks. Figure 1depicts the P-REACT's system conceptual architecture. The individual elements of the architecture will be described over the next sections. Essentially, the P-REACT solution is divided into two core building blocks; the

Embedded Sensor and the Cloud. The Embedded sensors are positioned in close proximity to the sensors (camera and microphone) located in retail premises and urban transport locations.

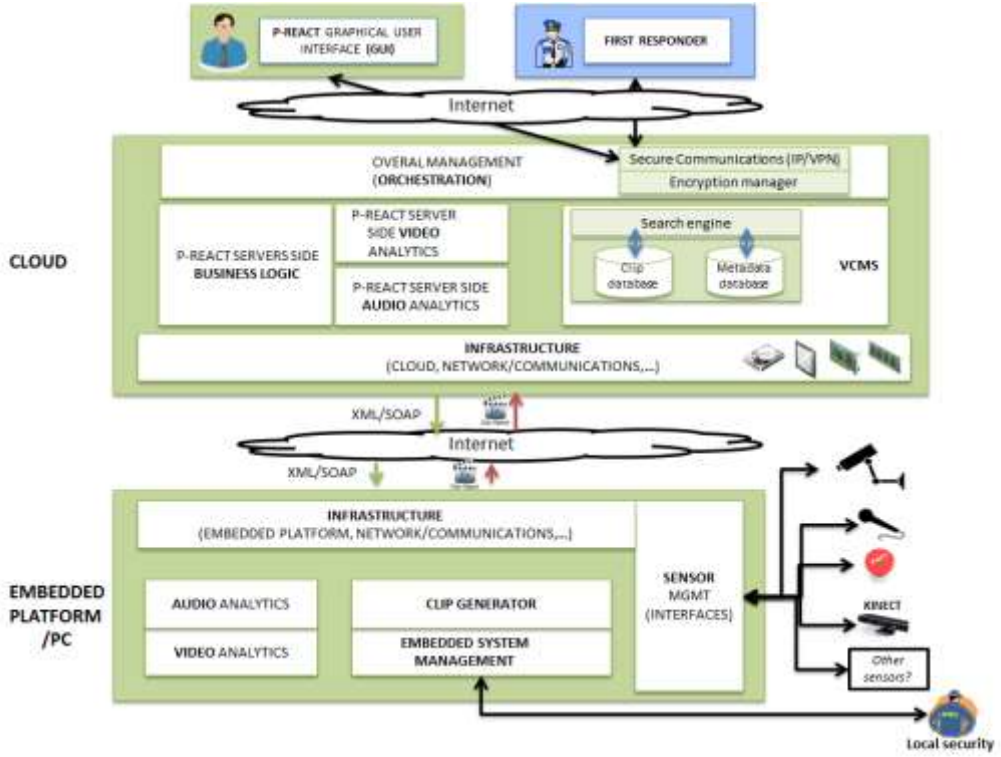

P-REACT GRAPHICAL USER INTERFACE (GUI)
FIRST RESPONDER
Internet
CLOUD
OVERAL MANAGEMENT (ORCHESTRATION)
Secure Communications (IP/VPN)
Encryption manager
P-REACT SERVERS SIDE BUSINESS LOGIC
P-REACT SERVER SIDE VIDEO ANALYTICS
P-REACT SERVER SIDE AUDIO ANALYTICS
Search engine
Clip database
Metadata database
VCMS
INFRASTRUCTURE (CLOUD, NETWORK/COMMUNICATIONS,...)
XML/SOAP
Internet
XML/SOAP
EMBEDDED PLATFORM /PC
INFRASTRUCTURE (EMBEDDED PLATFORM, NETWORK/COMMUNICATIONS,...)
AUDIO ANALYTICS
VIDEO ANALYTICS
CLIP GENERATOR
EMBEDDED SYSTEM MANAGEMENT
SENSOR MGMT (INTERFACES)
KINECT
Other sensors?
Local security

Figure 1 Conceptual System Architecture

2.1. Embedded System

The Embedded System is the part of the P-REACT system that is installed locally in the premises of a small shop or a bus station. It is responsible for managing and capturing data from the audio and video sensors, as well as to perform the necessary analysis on the captured data in order to detect volume crime events. This section will describe its main building blocks which are the (1) Embedded System Manager, (2) the Sensors Manager, (3) the Analytics Module and (4) the Clip Generator and can be seen in the lower part ofFigure 1.

2.1.1.Embedded System Management-ESM

Embedded System Manager (ESM) is the "brain" of the Embedded System (ES). Its main goal is to effectively coordinate all the individual components of ES towards detecting "petty crime" incidents, recording them to Clips using open formats, and uploading them to the Cloud for further analysis. In this context, ESM utilises a Controller and a number of Databases, with the former used for facilitating the message exchange among the various ES components (Message Dispatcher) based on some predefined configurable "logic" and the latter used for storing/retrieving information relative to the operation of each ES module (DB Operator).

In more detail, Controller's primary building units are the Message Dispatcher (MD), Logic and DB Operator (DBO). MD unit can be perceived as the central communication hub of the Embedded System:

messages (e.g. requests, informational) from any component arrive first here and messages to any component are generated or forwarded from here. On the other hand, the Logic unit holds the "intelligence" of the Embedded System; it provides the "rules of engagement", the thresholds of characterising an event as a petty crime and the consequent actions (Clip generation, send alarm etc.). Using the same analogy, if the Logic unit is Embedded System's "intelligence", DBO is the "memory"; it stores and retrieves information related to ESM components operation.

Regarding the databases used in the ESM, Sensors DB holds information related to the Embedded System's sensors (i.e. cameras, microphones etc.), including, but not limited to, the Sensor's id, type and access Uri. In the same context, Analytics DB stores specifics about the available analytics algorithms. Credentials DB keeps usernames and passwords needed to connect to the P-REACT cloud while Sensors Data DB maintains information about the data captured from the sensors and which of it have been sent to the analytics modules or streamed to the cloud. Finally, Analytics Results DB piles up the sensor's data analysis results (e.g. anomaly scores) whilst the Clip DB accommodates metadata describing the generated Clips (e.g. Clip's id, storage Uri, size, etc.).

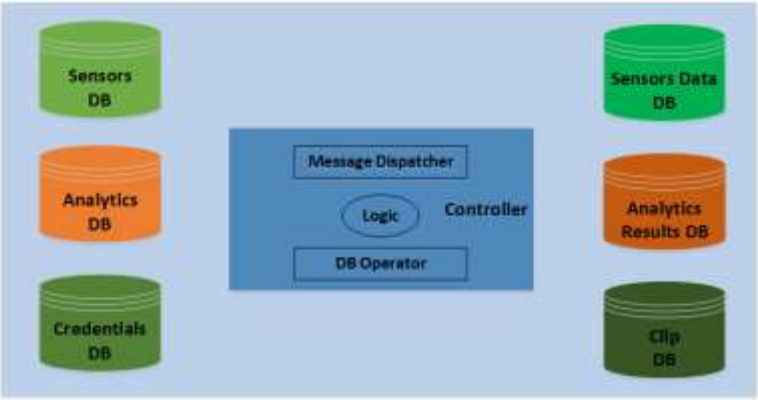


Figure 2 Embedded System Manager's conceptual overview

2.1.2.Sensors Manager

Sensors manager accounts for the operations needed to acquire sensors' data or get/update sensors' configuration. A software module, namely the Sensors Manager (SM) module, provides these operations acting as the intermediate between the actual sensors (e.g. cameras, microphones etc.) and the rest of the Embedded System (ES) modules. In more detail, SM comprises of a Sensors Data Streamer (SDS), a Sensors Configurator (SC) and a Clip Data Storage unit. SDS handles all the operations related with the sensors data such as getting sensors' data and forwarding them to the ES analytics, storing Sensors' data as Clips (Clip Data), generating metadata that describe the sensors' data (e.g. A/V codec, bitrate, access URI, etc.). On the other hand, SC manages all the sensors' configuration operations like updating or getting a sensor's configuration. Finally, CDS is a specific folder in the Embedded System's file system where the Clip Data (i.e. sensors' data) are saved.

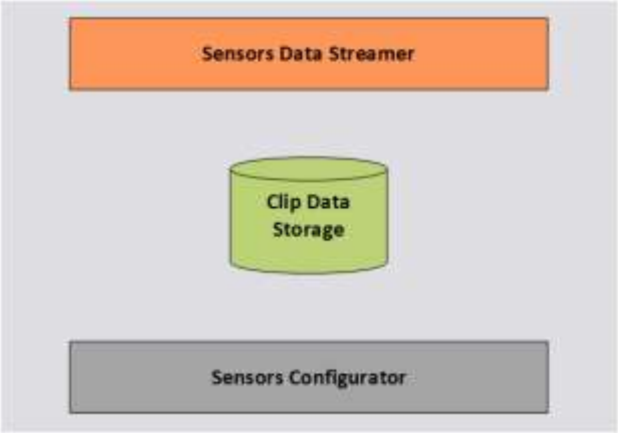


Figure 3 Sensor Manager's conceptual overview

2.1.3. Analytics Module

The Analytics Module consists of a number of sub-modules that are responsible for performing the necessary actions in order to detect abnormal behaviours in the area monitored by the embedded system. In the current design, it consists of two sub-modules, Video and Audio analytics, but it can be easily expandable to include other type of analytics if needed. The Video Analytics sub-module is divided into two parts, one responsible for dealing with standard RGB video, and one to deal with more advanced depth videos.

Video Analytics

In this section the architecture of the video analytics methods that will be deployed on the embedded system is described. Due to the limited computational resources of the Embedded System, certain design and implementation choices have to be made in order to ensure the robust and real time performance of the video analytics component. One the most important design principles governing the architecture of the video analytics is to comply with privacy and ethical regulations that are explained in Deliverables D1.5 and D1.6. In order to comply with these regulations, the video analytics algorithms will utilize a local video buffer spanning a few minutes, and only in the case of an abnormal event will send video data to the cloud along with any metadata (video, audio, depth etc.). This ensures that no data will be stored on the embedded system or sent to the cloud unless there is an event, thus preserving privacy of the general population. In the following paragraphs the details of the individual subsystems of the video analytics module are described.

The purpose of video analysis is to provide the P-REACT system the ability to automatically extract useful, rich or even hidden information from raw video footage. The extracted information contains data that can be consumed by other processes or machines to trigger subsequent actions (e.g. storage, forwarding transform, etc.).

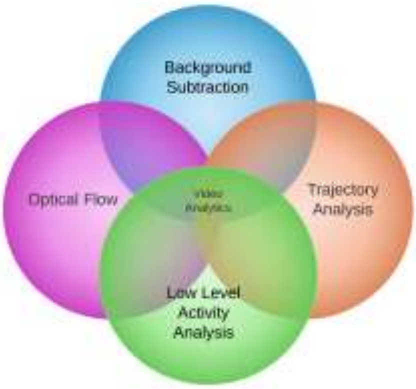


Figure 4 Video analytics components on the embedded system

In the context of surveillance applications, and in particular, in the context of the P-REACT project, the specific goal of video analytics is to automatically detect petty crimes occurring in the scenes viewed by the cameras. Given the distributed nature of the P-REACT architecture, this goal can be achieved dividing the actions into two parts: (i) detect events or symptoms that might be related to petty crimes, typically in the form of abnormal or distinctive visual patterns (e.g. rapid motion, sterile zone trespassing, fighting detection etc.) in real-time; (ii) process, at the cloud-level, the information generated by the embedded-level and the video clips in an offline fashion, to apply more powerful algorithms that eventually recognize the person(s) involved in the event in clips stemming from other embedded systems or in clips stored in the VCMS Although the system will be able to recognize people across different clips and views, the actual identity of each individual thus providing forensic evidence while at the same time preserving privacy. This separation of the processing is mandated by the limited computational power of the embedded-platform, which is aimed to be a low cost device. The conceptual view of the video analytics is depicted in Figure 5

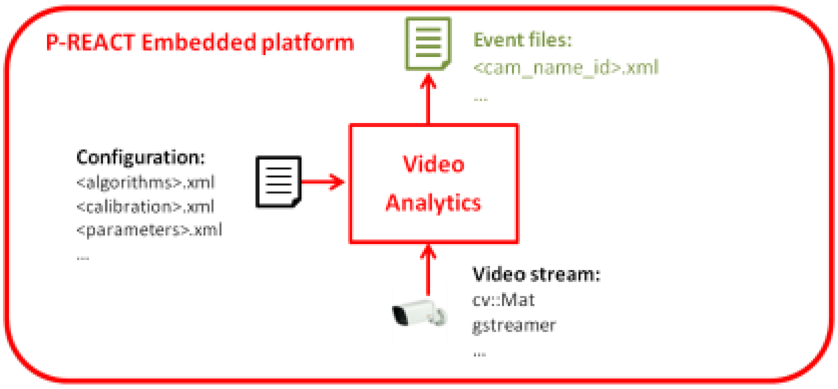


Figure 5 Video analytics at the embedded level.

The limited computational power available at the embedded-level limits the types of algorithms that can be deployed. Iterative approaches, non-linear regression or brute-force scanning techniques have to be avoided.

Some examples of the algorithms that fit both the requirements of online processing and light-weight complexity are:

- **Background subtraction algorithms**: pixel-wise analysis of the intensity or colour values to determine whether a pixel represents the background or the foreground. The output can be rendered as a binary mask that can be further used to determine sterile zone trespassing, size of objects, etc.

- **Optical flow**: keypoint-level analysis of the apparent motion of the scene. These are typically techniques that use only the current and previous frames to compute the differences and determine the apparent instantaneous motion. This output flow information can be used later to determine whether the motion is in the scene is normal or has abnormal patterns (e.g., people running, aggressive person movements, etc.)

- **Blob analysis and linear tracking**: object-level analysis that can use simple connected-component analysis to move from pixel-level or keypoint-level information (such as the provided by the abovementioned methods) to object-level. Linear tracking methods, such as the Kalman filter does follow the online processing philosophy and can be used to efficiently generate trajectories of simple motions. Such analysis can provide as output sizes, speed and trajectory information of (unknown) objects in the scene.

- **Context-based scanning**: brute force analysis (or sliding window scanning) is a well-known technique that exhaustively explores the image in search of specific patterns that match a given target object (e.g. faces, cars, etc.). The computational cost of these techniques is excessive for embedded-platforms. However, contextual information (e.g. perspective information of the scene, regions of interest, domain-specific rules) can be used to reduce the number of hypotheses to explore and thus effectively exploit this type of detectors.

The Embedded System Manager (ESM) will inform to the Video Analysis module which algorithms must be executed based on the configuration parameters. This dynamic configuration process permits the P-REACT system to adapt the behaviour of the video analytics to the system context.

Besides the commonly used colour cameras, the P-REACT solution will also utilize depth sensors. This type of sensors provide richer information for event detection while at the same time are privacy preserving. On the other hand, depth processing is more computationally demanding, so it will be included as an “advanced” Embedded System in the final solution. Furthermore, depth sensors are not affected by issues such as such as shadows, luminance variations and partial occlusions, which affect colour images processing disadvantages,

An approach similar to the colour images processing will be followed for depth images, towards detecting abnormal events by processing the 3D information acquired from the depth images over time (4D processing), aiming at human driven event detection and recognition. Furthermore, at the same time the individual’s privacy will be respected and preserved according to EU’s ethical laws, since no colour information will be utilized in the depth video analytics module.

Audio Analytics

Humans classify audio signals with no conscious effort. Recognizing when someone is calling for help or focusing on a conversation even under noisy environments are relatively easy tasks for a human but a real challenge to machines. The cognitive decomposition and understanding of an auditory scene may become troublesome to a computer system; especially when faced with processing power constraints. Processing of both video and audio on a limited processing power embedded platform does not leave space for the employment of complex methods. In this section, the concepts of audio analytics on the embedded device are briefly described.

The system captures, through a microphone, a continuous audio stream which then analyses with the aim to detect and identify audible events of interest. Analysis of the audio stream is carried out in a block based method where for each block, a number of spectral, cepstral and unvoiced coefficients are extracted and used for subsequent classification. For the needs of the project, the audio analytics module on the ES is designed to focus on the detection of certain events (screaming, glass breaking and gun-shooting/loud explosions) as well as the detection of keywords belonging to a small dictionary (around 10 words). Despite the fact that gun-shooting may not be considered as petty crime, it was included in the analysis with the aim to demonstrate that the detection of versatile events is possible. These audio triggers have been selected as they are indicative of many petty crime incidents like assault, break-ins etc. Furthermore, the chosen audio triggers take into account the limitations of the ES, as well as the levels of reliability, robustness of the analytic approaches. The integration of audio analytics to the P-REACT system aims to fill the gap when video/depth analytics are not in the position to provide results (dark environments, incidents outside the field of view, inability to detect events due to overcrowded areas) without impacting significantly the rate of false positives. Furthermore, by combining video and audio analytic, it is expected that false alarms will be reduced and detection rates increases.

The implementation of event detection on the ES will be implemented with the use of a method based on decision trees. The selection of the specific classification method depends on both the classification accuracy as well as the required processing time. Both parameters are critical as they impact greatly the overall performance of the system. Inadequate classification performance would result in an increase of false positives while slow classification speed would result in overburdening the ES resources resulting in delays and a non-real time solution.

On the other hand, keyword detection will be implemented with the deployment of pattern recognition methods. This task is quite challenging, however, its successful completion is expected to add great value as it will allow the triggering of the system based on predefined words that may signify important events. In the scenario of a robbery in a shop, the system might be trained to listen to a specific keyword (irrelevant of the event) which could allow the cashier to surreptitiously trigger the system. For the needs of the project, a small dictionary with keywords has been generated to serve this purpose. The dictionary contains the keywords: help, police, thief, ambulance, accident, robbery, murderer, call

911, call the cops/police, fire. The prototype of this module, within this project, is designed to support English. However its successful operation should demonstrate the potential in its use and motivate future projects in its development in a number of different languages.

In the event where an alarm is raised, the transfer of the extracted audio characteristics/coefficients through the network to the cloud storage is initiated; this allows the coefficients to be available for further analysis.

The development of a classifier that detects specific sounds is comprised of two stages, the training of the model using prior knowledge (pre-recorded sounds) and the prediction stage at which the model is deployed on site and is called to categorise new material (see Figure 6). During the training phase we will consider features of sounds that we are interested in detecting, as well as background noise and other general sounds; overall a set of collected sounds of different types and from different environments. This process is necessary to enhance system robustness and remove any bias. The second phase involves the deployment of the solution on the Embedded System. The necessity to operate and take decisions in real time requires splitting the received data stream into frames of predefined size; each frame is sequentially analysed and a set of extracted features is obtained. The prediction module is then called to make the binary decision (i.e. the sound is indicative of event detection). A similar approach is expected to be followed for keyword detection for the provision of a timely method capable of producing results under a resource constrained system. The parameterisation of the described process will be determined during the implementation of the algorithms where a number of experiments testing the statistical validity of the results will be carried out in an effort to ensure effective operation.

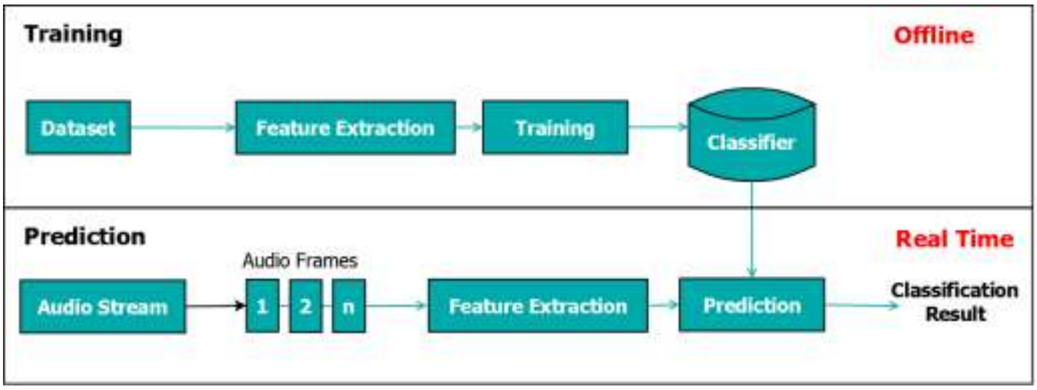


Figure 6 the diagrammatic representation of processes and tasks that take place during audio analysis.

2.1.4. Clip Object / Clip Object Generator

Clip generation refers to the production of data units containing sensors' data (named Clip Data) and metadata (named Clip Objects). Clip Generator (CG) module is responsible for carrying out this task and to do this, it exploits a Clip Object Generator (COG), an Integrator and a Clip Storage (CS) unit. COG produces the Clip Objects, JSON files that not only contain information regarding the Clip(s) Data (i.e. sensors' data) and the hosting ES but may also contain analysis data derived from the Embedded

System's analytics modules. In turn the Integrator "incubates" the Clips by merging the previously produced Clip Object and the available Clip Data. Finally, CS is a specific folder in the Embedded System's file system where the Clips are stored.

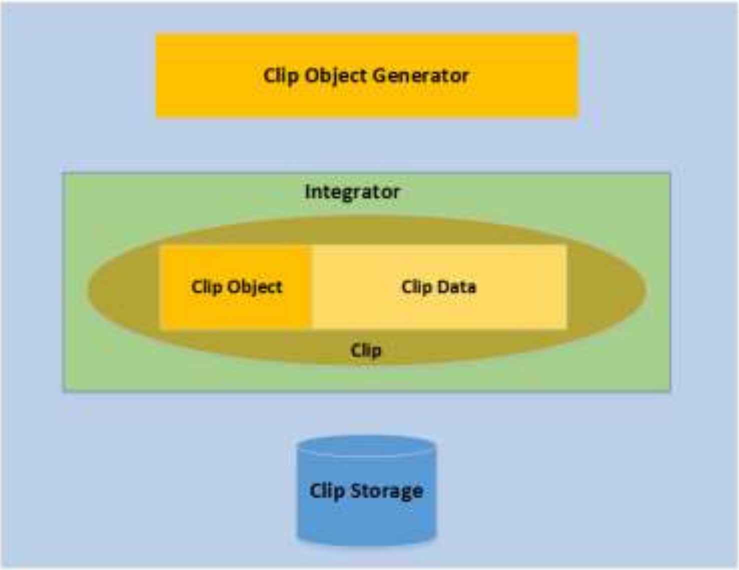


Figure 7. Clip Generator's conceptual overview.

2.2. Cloud

The cloud system of the P-REACT's solution serves as a centralized node where all the embedded systems are connected. It hosts a number of databases and a storage unit where all the uploaded clip data is stored, as well as advanced analytics modules (video and audio). The operations on the cloud are coordinated by an orchestration module, while the decision making is undertaken by the Business Logic module. The architectural elements of the cloud system can be seen in the top part of Figure 1. Over the next paragraphs, the individual architecture of the aforementioned modules will be presented.

2.2.1.Orchestration

The Orchestration component is a central point via which all communication takes place (see Figure 8). This allows all logging operations to take place at a central location (all events pass through the Orchestration); also serves as a single point in the system for setting, controlling and enforcing policy.

The Orchestration operates with a priority queue responsible for storing all service requests. Each request is assigned to a priority according to how quickly needs to be serviced. This is done to allow the separation of real-time operations (high priority) and batch operations (low priority). The main disadvantage of such centralized systems is the heavy load of requests which a single point (the Orchestration) is called to handle.

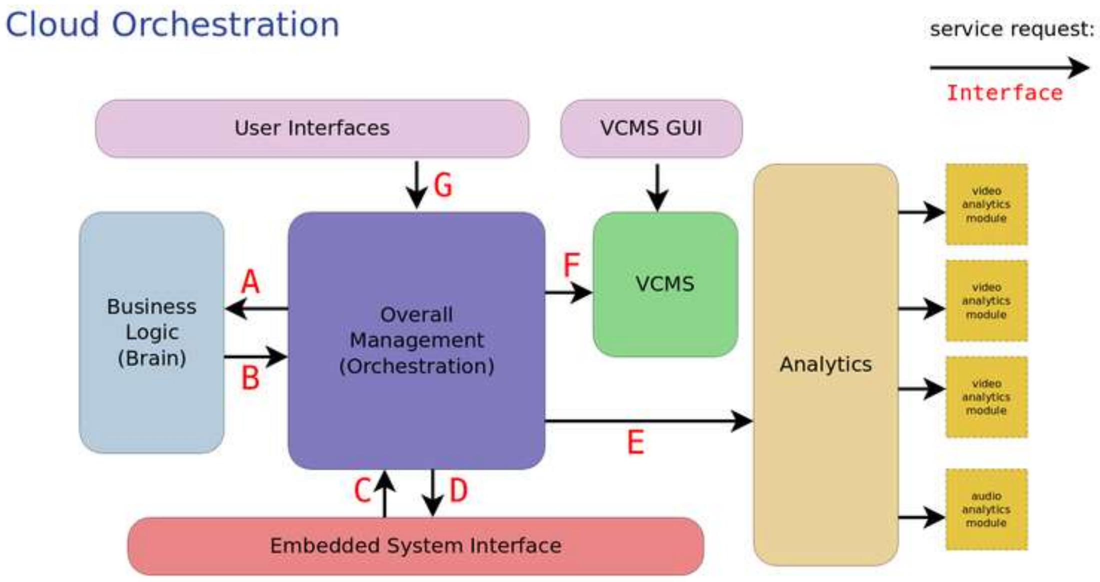


Figure 8 Cloud Orchestration block diagram interfacing the interconnections between modules

2.2.2.Business Logic

The Business Logic Component (aka 'Brain') is the component which manages the cameras, VCMS and interfaces with the human operators via the GUI. It is composed by three parts are:

- Alert Raising, which decides what type of alert, should be raised. The decision is taken with input from cloud-side video analytics and statistical analysis.
- Interaction & Control that provides feedback to the Embedded Systems. It is transferred via the "Encryption Manager/Secure Communication" component in the form of an xml/soap object.
- Real Time monitoring, which is the point where Clips are accessed, either live Clips or stored Clips, and displayed to the GUI. GUI/User can validate Clips.

2.2.3.Analytics Module

The analytics module on the cloud is responsible for two tasks:

1. Verify the event sent by an embedded system
2. Identify the involved persons

As in the case of the embedded system, a video and an audio sub-module will be used. The video sub-module will utilize both colour and depth information as it is available. Also, as mentioned in Section 2.1.3 0, the raw audio signal is not transferred to the cloud due to ethical considerations, but only a set of features is transferred.

Video Analytics

In this section we describe the architecture of the video analytics component that will reside on the cloud subsystem of the P-REACT platform. In contrast to the Embedded System, computational resources are not an issue at this level and therefore more intensive computer vision algorithms can be applied.

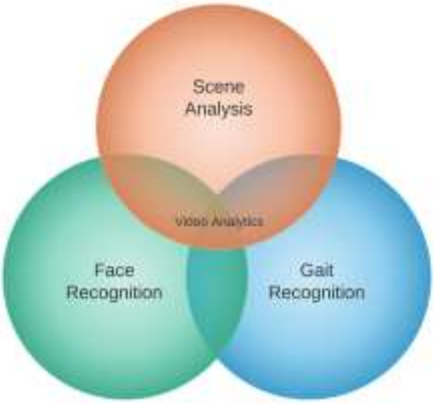


Figure 9 Video analytics components on the cloud

If an incident occurs, the video analytics component of the cloud will use the data sent by the respective Embedded System and at first will verify the existence of the event in order to minimize the false alerts stemming from the lack of resources at the embedded system. This will be achieved by executing algorithms similar to the ones residing on the embedded system, but with more powerful hardware, thus with more accuracy. By having the ES only sending clips, the privacy and fundamental rights of the general public are more shielded. The second task of the video analytics on the cloud is to try to match the individuals involved with its database of past incidents and/or with clips coming from Embedded Systems neighbouring with the one that raised the alert. To achieve this, the system will employ privacy preserving algorithms that perform face, gait and appearance recognition. If an offender is recognized, the operator of the system will be notified to take further action. Furthermore, authorized operators of the system will have the ability to record on demand videos from areas where an abnormal event has occurred. All data will be stored in an encrypted database following the procedures described in Deliverable D2.2.

Audio Analytics

The role of cloud audio analytics is to perform further analysis on the coefficients with the aim to validate and subsequently determine the severity of the alert. Cloud analytics may be seen as a tool for assisting Business Logic by signifying the activation of nearby ESs and also as a second level validator to reduce the number of false positives.

Audio features are transferred to the cloud once an alarm is generated at the Embedded System. Processing on the cloud imposes two significant advantages related to time and processing resources.

Cloud analytics still processes the available data in blocks however a larger selection of analytics algorithms can be applied. For the needs of the project, three algorithms will be implemented on the cloud, one method will be based on decision trees and random forests, the second one will be based on artificial intelligence methods (neural networks, fuzzy logic), and finally a classical statistical method based on decision boundaries. The decision of the specific algorithms to be used will be determined during WP3 and based on the classification performance of the tested methods. However, the selection of algorithms with different fundamentals is intended (methods deriving from decision trees, statistical methods and computational methods) and expected to challenge the robustness of the triggering while also providing more information on the composition of the sound.

2.2.4. Video Content Management System (VCMS)

The Video Content Management System contains structures for managing (and updating) metadata for the fast indexing and retrieval of content while also providing functionality allowing transcoding and playback of clips. The VCMS is responsible for handling three fundamental structures, the Clip objects, the user objects and the analytics objects.

A Clip object is the main way of exchanging information between different components in the architecture and is a set of key-value pairs (metadata) which describe the video and audio. The Clip object does not contain any binary data. The binary information (aka clip data) is stored on the Open Stack on the cloud. The Clip object contains a field which contains the URI pointing to the binary data. Clip objects are stored in the VCMS for archiving and fast retrieval. User objects contain basic information about each authenticated user and the analytics objects contain information about the various analytics methods and a URI referring to the binary executables which run on the Embedded System.

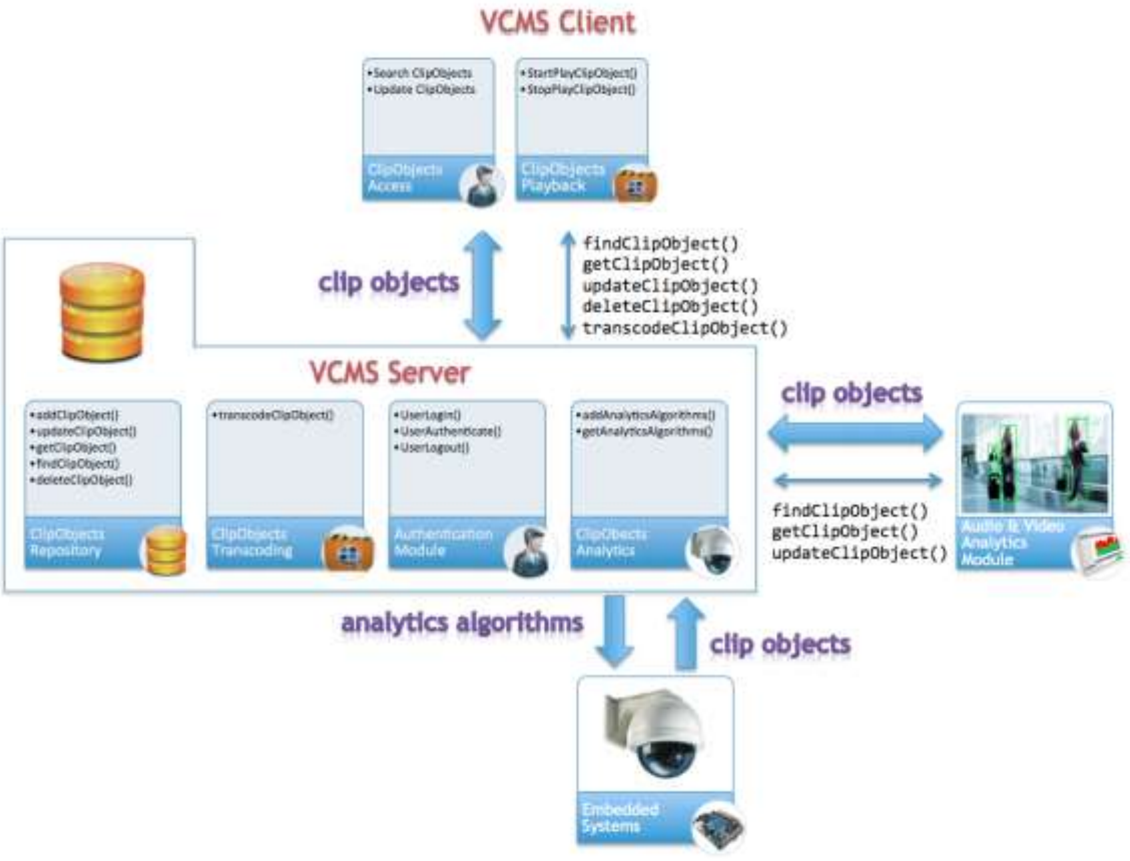


Figure 10 the VCMS system: communication between elements and possible operations

The VCMS system supports video in various formats and from multiple sources and has the ability to archive content as well as mechanisms for both the manual manipulation and automatic processing of content. The system may accept processing commands and provide access to data through its Application Programming Interface (API). The API is configured and is rich enough to allow the development of web-based graphical user interfaces (GUI) for searching, retrieving, and manipulating video clips, for managing metadata and visualizing stored content.

2.3. Communication

Secure communication between the Cloud and the Embedded System(s) is of vital importance towards achieving P-REACT's goals. A software module, namely the Communicator module (COM), residing both in the Cloud and the Embedded Systems, takes over this task. In this respect, COM utilises an Encryption Manager (EM) unit that can be used for adding extra layers of protection to the exchanged data, a Clip Handler (CH) unit that handles the transfer of Clips and a Secure Communication Channel Manager (SCCM) that setups an encrypted communication channel between the Embedded System and the Cloud. SCCM will exploit OpenVPN framework, utilising either Public Key Infrastructure (PKI) either pre-shared key approaches, towards establishing the encrypted channels whereas EM, for keeping low the processing overhead in the Embedded System, could exploit shared keys (symmetric encryption) created at the Cloud's EM—these keys can safely be exchanged via the encrypted communication channel.

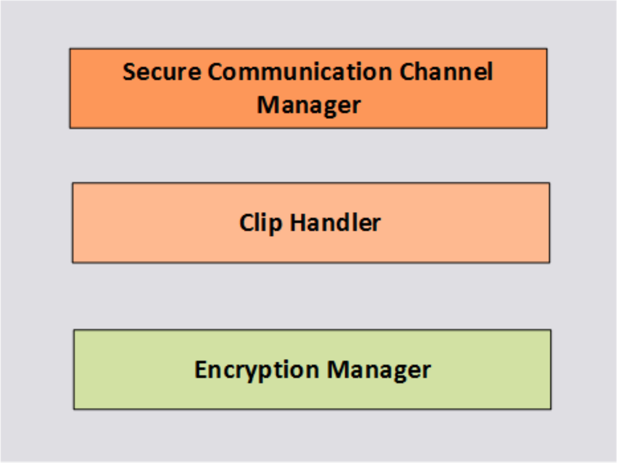


Figure 11 Communication Manager's conceptual overview

2.4. Privacy By Design Approach

P-REACT has undertaken a privacy-by-design approach from its early inception in order to ensure that individuals’ privacy and protection of their personal data are taken into consideration at all stages of systems lifecycle. The core principles for the protection of personal data, namely transparency, proportionality, data minimization and ethical values, as they have been described in:

a. the Charter of Fundamental Rights of the European Union,

b. the European Convention for the Protection of Human Rights and Fundamental Freedoms,

c. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data and

d. EU Data Protection Regulation Reform Proposal

All the aforementioned directives have been taken into consideration when putting together both functional and non-functional requirements, as they have been already described within D.2.2 - P-REACT Local Embedded framework and system on the cloud requirements. Additionally, prior to the finalization of the Conceptual Architecture, a Privacy Impact Assessment has been carried out and is presented in D 1.5 – Privacy Impact Assessment. Within the scope of this report, the confidentiality of data collected, processed and stored is addressed by the Encryption Manager which is presented in Section 2.3. However it should be noted that by assessing the proportionality between (private/public) interests to be protected, means to be used and invasion/harm for the monitored individuals we have to take into account that we are talking about misdemeanour infractions such as shoplifting and minor cases of public disturbance and not “serious crimes”.

Regarding the risks of stigmatization and discrimination, due to the fact that P-REACT is based on CCTV, it is a possibility that data revealing race or ethnic origin, political opinions, religion or beliefs, data concerning health or sex orientation/life. We take all the necessary measures in order to avoid invasion of privacy, revealing of sensitive (categories) of data and consequently risks of stigmatization and discrimination. Adopting the definition used within the Council of Europe - Application of Convention 108 to the profiling mechanism document[1], originally proposed by Bygrave, 2001 [1] “profiling is the process of inferring a set of characteristics (typically behavioural) about an individual person or collective entity and then treating that person/entity (or other persons/entities) in the light of these characteristics. As such, the profiling process has two main components: (i) profile generation – the process of inferring a profile; (ii) profile application – the process of treating persons/entities in light of this profile. Three stages are therefore necessary in order to perform what we call profiling:

- large quantities of digitised data from observation of the behaviour and characteristics of individuals,
- determination of the probability relations (correlations) between certain behaviours/characteristics and other behaviours or characteristics,
- Inference, based on certain behavioural variables or observable characteristics of an individual identified in general terms, of new characteristics or past, present or future behavioural variables.

As discussed on Section 2.1.3, the embedded platforms will employ algorithms based on unobtrusive

[1] http://www.coe.int/t/dghl/standardsetting/dataprotection/Reports/CRID_Profiling_2008_en.pdf

features such as motion and depth information, as expressed from point trajectories and depth maps. Only in the case of an abnormal event will the Embedded System send a clip object to the cloud along with all relevant information (audio signal, depth map etc.), in order to perform "subject" identification and only reveal data to competent and authorised individuals under the necessary procedural, legal and judicial authorisation. Also, as already mentioned, in order to avoid stigmatization, extra steps to ensure a minimal false alarm rate will be taken, by validating the existence of an event on the cloud, and if the event is not verified, all relevant data will be deleted.

3. Functional View

The functional view of software architecture defines the architectural elements that deliver the system's functionality. The functional view documents the system's functional structure that demonstrates how the system will perform the functions required. According to Rozanski & Woods [33], the functional structure model of the Functional View typically contains functional elements, interfaces and external entities/ connectors:

- Functional Elements constitute well-defined parts of the system that have particular responsibilities and expose interfaces that allow them to be connected to other elements. A functional element can be a software component, an application package, a data store, or even a complete system.
- Interfaces are specifications, defining how the functions of an element can be accessed by other elements. An interface is defined by the inputs, outputs, and semantics of each operation offered and the nature of the interaction needed to invoke the operation.
- External Entities are connectors which can represent other systems, software programs, hardware devices, or any other entity the system communicates with. These are described as dependencies to other systems or components.

3.1. Embedded System

In the following subsections the functional view of the Embedded System's modules will be presented.

3.1.1.Embedded System Manager

As illustrated in the Section 2.1.1 and depicted in the following figure, ESM utilises a number of Databases, a Controller and several communication interfaces, towards coordinating Embedded System's (ES) components to perform efficiently their tasks. As a result, ESM interacts with the Analytics modules towards enabling access to the Sensors data, receiving the Analytics results, and (re)configuring the Analytics algorithms and techniques. The outcome of the last two interactions is stored to the Analytics Results and Analytics DBs respectively. In the same fashion, ESM exchanges messages with the Sensors Manager towards requesting, on behalf of the local Analytics modules or even the Cloud's, data of a specific sensor. Furthermore, ESM can request the (re)configuration of the Sensors. If ESM's internal "logic" decides that a possible petty crime incident occurs, the Clip Generator is activated and the generated Clips are uploaded to the Cloud via the Communicator module.

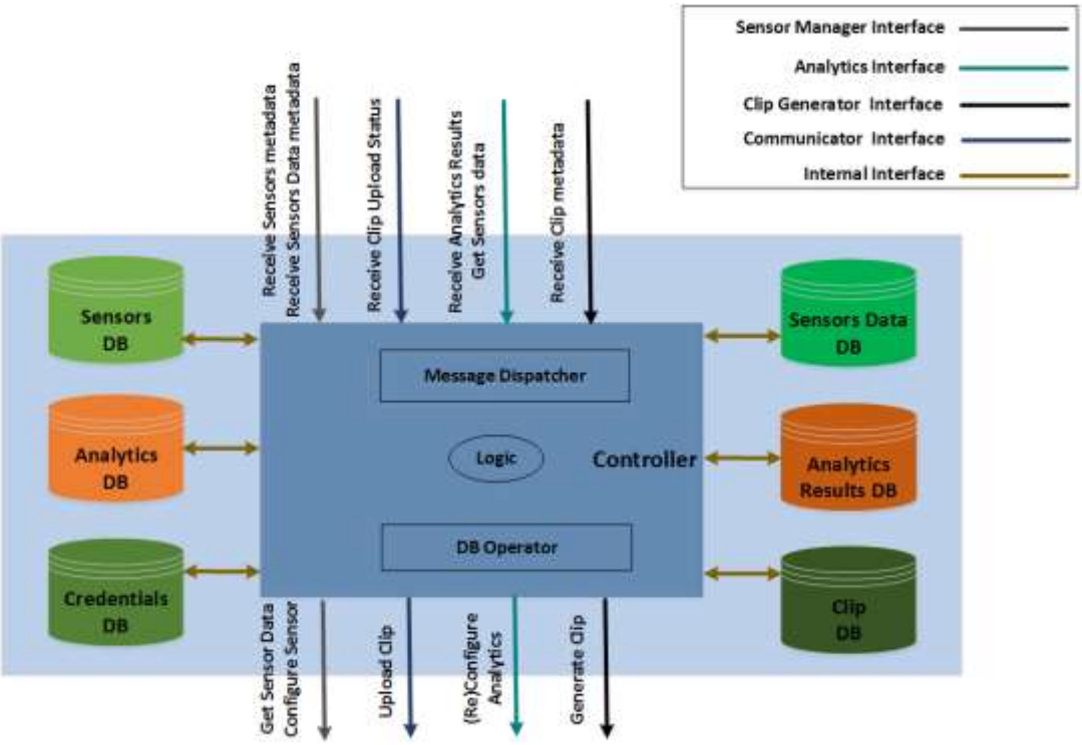


Figure 12 Embedded System Manager's functional overview

In summary the following interfaces are utilised for the communication with the rest of ES components:

1. Sensors Manager:
   a. send_sensor_data_request(): This interface allows ESM to send a message requesting, on behalf of Analytics modules or even the Cloud, data from a specific sensor. The request indicates if the data will be offered in the form of network streams, or files, or both.

   b. receive_sensor_data_metadata(): This interface enables ESM to receive from SM the results of the sensor data request. If the operation was successful, the response contains metadata describing the offered Sensor's data, otherwise the response indicates the failure.

   c. send_sensor_configuration_request(): This interface will allow ESM to request, itself or behalf of others, a change in the sensors configuration.

   d. receive_sensor_configuration(): This interface enables ESM to receive from SM the results of the configuration request.

2. Analytics:
   a. receive_sensor_data_request(): This interface allows ESM to receive the Analytics requests from a specific sensor's data.

   b. receive_analytics_results(): This interface allows ESM to receive the Analytics results.

c. send_configure_analytics(): This interface allows ESM to (re)configure the Analytics running on the ES.

3. Clip Generator:

    a. send_generate_clip(): This interface allows ESM to inform Clip Generator to initiate or halt the production of Clips.

    b. receive_clip_metadata(): This interface enables ESM to receive metadata for every generated Clip.

    c. receive_clip_data_uris_request(): This interface allows ESM to receive a request for the Uris of the Clip data that will be part of the next to be generated Clip.

    d. receive_analytics_metadata_request(): This interface allows ESM to receive a request for the Analytics metadata (if any) that will be part of the next to be generated Clip.

4. Communicator:

    a. send_upload_clip_request(): This interface enables ESM to request the upload of a Clip to the Cloud.

    b. receive_clip_upload_status(): This interface enables ESM to receive from the Communicator information regarding the uploading of a Clip to the Cloud.

    c. send_channel_setup_request(): This interface enables ESM to send to Communicator a request for setting up the secure communication channel.

    d. send_channel_status_request(): This interface enables ESM to inquire Communicator about the status of the secure communication channel.

5. Internal:

    a. store_clip_metadata(): It stores the Clip metadata to the Clip DB.

    b. store_sensors_metadata(): It stores the Sensors metadata to the Sensors DB.

    c. store_analytics_metadata(): It stores the Analytics metadata to the Analytics DB.

    d. store_credentials_data(): It stores the credentials data to the Credentials DB.

    e. store_sensors_data_metadata(): It stores the Sensors' data metadata to the Sensors Data DB.

    f. store_analytics_results(): It stores the Analytics results to the Analytics Results DB

3.1.2.Sensor Manager

Sensor Manager (SM) comprises of two main processing units, namely the Sensors Data Streamer (SDS) and the Sensors Configurator (SC) units (see2.1.2); several external and internal communication interfaces; and one Storage place for saving the actual Clip data (e.g. Video Clips).

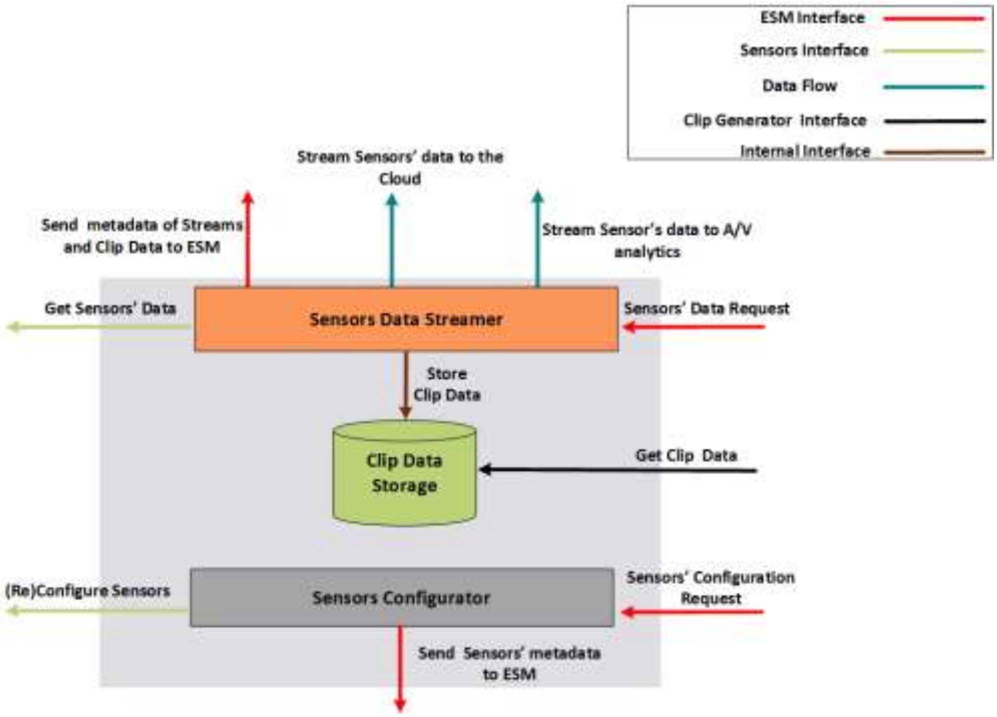


Figure 13 Sensor Manager's functional overview

In more detail, Sensors Data Streamer (SDS) is the component responsible for interacting with the Embedded System's (ES) sensors for getting their data. In this context SDS, after receiving a request from the Embedded System Manager (ESM) for the sensors' data (e.g., Video, Audio), captures this data by exploiting the information (e.g. sensor's access URI) contained in the request message and then streams them to the local Video Analytics modules (also to the Cloud's ones, if the request message indicates so) besides storing them as clips, for a finite amount of time, in the Clip Data Storage. Furthermore, SDS generates, for each stream and clip, metadata describing their attributes (e.g. Video/Audio codec, bitrate, access URI, etc.) and sends them to ESM via the appropriate interface. On the other hand, Sensors Configurator (SC), as the name implies, is used for (re)configuring ES sensors, a process that is related to the sensors capabilities and can be initiated upon a remote (Cloud) or local (ESM) request. Upon a successful (re)configuration SC creates a message containing the new Sensor configuration and sends it to ESM.

In summary the following interfaces are utilised for the communication with the rest of ES components:

1. ESM:

   - receive_sensor_data_request(): This interface allows SM to receive a message from ESM requesting, a specific sensor's data. The request indicates if the data will be offered in form of network streams, or files, or both.
   - send_sensor_data_metadata(): This interface enables SM to inform ESM about the results of the sensor data request. If the operation was successful, the response contains metadata describing the offered sensor's data otherwise the response indicates the failure.

- receive_sensor_configuration_request(): This interface allows SM to receive a message from ESM, requesting a change in the sensors configuration.
- send_sensor_configuration(): This interface enables SM to inform ESM the results of the configuration request.

2. Sensors:
   - get_sensor_data(): This interface allows SM to get and offer the Sensor's data conforming to the received message of sensor data request.
   - configure_sensor(): This interface allows SM to configure the sensors according to the received configuration request.
3. Clip Generator:
   - receive_clip_data_request(): This interface allows the Clip Generator to get in hand the available Clip Data.
4. Internal:
   - store_clip_data(): It stores the Clip Data to the storage folder.

3.1.3. Analytics Modules

In the following sections, the functional view of the analytics modules for the embedded system will be presented. The analytics on the embedded system that will be developed are divided in two main categories: (1) video and (2) audio analytics. For the video analytics, the methods that will be developed include:

1. Motion Detection, where motion will be detected in restricted areas
2. Fighting detection: two or more people fighting
3. Chasing detection: two or more people chasing
4. Bag Snatching: one or more people steal a bag from another

Audio analytics will include the following methods:

1. Keyword detection, where a set of predefined words will be detected as an alarm
2. Screaming
3. Glass breaking, such as windows
4. Gun shots/loud explosions

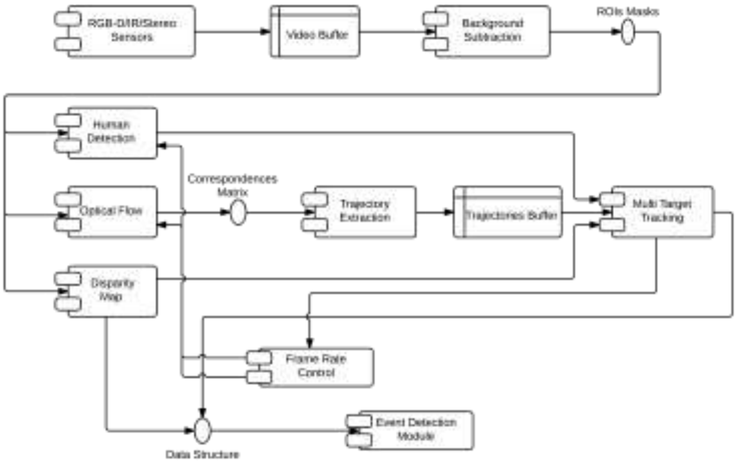


Figure 14 Video Analytics module functional overview

Video Analytics

In designing the architecture of the video analytics module, the limited computational resources of the Embedded System were taken into account and incorporated a throughput control mechanism that ensures the real time operation of the system while at the same time utilizes its maximum potential. Another underlying principle of the design that speeds up the processing pipeline is that the interfaces between the components should consist of data structures residing exclusively on the primary memory of the Embedded System. Moreover, the described architecture provides a local storage mechanism for redundancy in the unlikely event of a severed network connection, as is described in Deliverable D2.2. The functional view of the video analytics module is illustrated in Figure 14.

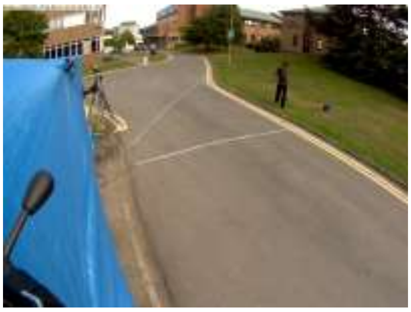


Figure 15 Initial RGB Frame

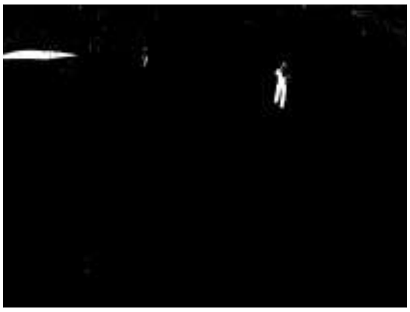


Figure 16 Background subtraction

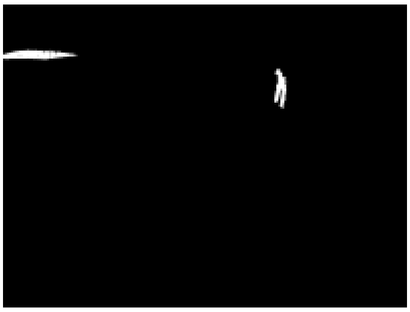


Figure 17 Foreground denoising

Figure 18 Optical flow field

The input to the video analytics module consists of raw sensory data coming from RGB/IR and is stored in the primary memory of the system on a video buffer. This buffer contains the information provided in the past seconds and the duration of the covered time period depends on technical specifications of the system, such as the amount of available memory and the camera's resolution. Initially the background will be subtracted from the data, in order to limit further processing only on the extracted foreground Regions of Interest (ROIs). Background subtraction can easily be performed robustly in real time without exhausting the resources of the system. The background model of this component will be dynamically updated in order to adjust to common variations of the background, such as illumination changes and repetitive motions from clutter [23], [24]. To remove any spurious artefacts on the detected foreground regions, simple morphological operations such as erosion and dilation can be applied [25] with negligible computational cost. Simple examples of these operations are depicted in Figure 16 and Figure 17.

As is depicted in Figure 17, the interface between the background subtraction and the optical flow component is a binary image containing the extracted ROIs. This image is stored in the primary memory of the system. The next functional element of the module computes the optical flow of the ROIs. It should be noted that this component connects with frame rate control, in order to ensure the real time operation of the system. If at any time instance, the frame throughput of the system decreases due to increased coverage of the ROIs, frame rate control will reduce the density of the optical flow. Optical flow builds on the assumption that the appearance of a point is constant between consecutive frames and by searching for similarly looking pixels across the frames one can infer the motion of the points [26] [27]. As a result, a point correspondence matrix is produced for the two frames. This matrix is also known as the optical flow field. The optical flow field of a single frame is shown in Figure 18.

The optical flow field is sent by the optical flow component to the trajectory extraction component as an array in primary memory. The task of the trajectory extraction component is to use the flow field in order to form long term point trajectories. The optical flow field can be used to compute the motion of any point on a region that is adequately covered through linear interpolation. Trajectories corresponding to new keypoints are added, while other trajectories are terminated if they exit the field of view (FOV) of the camera. Keypoints are initialised using a mesh grid where points with low texture are removed by using a simple keypoint detection algorithm such as the Harris corner detector [28]. The texture of a point is easily computed from the gradient of a frame after a Sobel filter is applied [25]. Furthermore, trajectories are terminated if the texture of the corresponding pixels becomes too low. The system stores the trajectories in an internal buffer on the primary memory which contains the more recent trajectories up to a certain frame in the past. The time period that the stored trajectories cover, depends on both the technical specifications of the system such as the amount of memory and the camera resolution, as well as on runtime parameters related to the texture of the objects that appear on the scene and the coverage of the detected ROIs.

Another component of the video analytics module is that of human detection. This subsystem will be applied on the ROIs of a few frames and detect regions where human are likely to exist. One of the most effective algorithms for this task is presented in [29] where Histograms of Oriented Gradient (HOG) descriptors are used. The operation of this component will be overseen by the frame rate control subsystem, which will allow the human detection algorithm to run only when the necessary computational resources are available. Optionally, a component producing the disparity map of the scene may be present in case a depth sensor is available. The information of the trajectories the human detection and the disparity will be fused in order to perform multi target tracking in a similar manner to [30], which can resolve occlusion cases (like people passing behind cars or each other, thus hiding from the cameras for a little time) that are of special interest for the P-REACT platform. Although human tracking is sparse along the temporal axis, P-REACT system using the computed trajectories will interpolate the position of the detected individuals in every frame. The output of these methods will be used in order to detect various abnormal events. In order to achieve this, a hierarchy of classes deriving from the concept of Event Detector will be created. Online and offline derivations will be implemented to support both embedded and cloud-level analytics, respectively. Although there might be several many ways to implement such event detectors (e.g. using stochastic or Markovian filters for the online case, or batch execution processes for the offline case), we want to focus on the **"Rule Manager"** concept, that represents the core functionality of the video analytics at this level. For each specific location, the Rule Manager can be fed with one or more Rules, each of them configured to detect a given event (or set of events), and optionally parameterized with contextual information from the scene.

The implementation of the Rule Manager will be done to be as flexible and scalable as possible, so that new Rules can be added dynamically, or existing Rules can be configured to be applied with different parameters or into different regions of the image, etc.

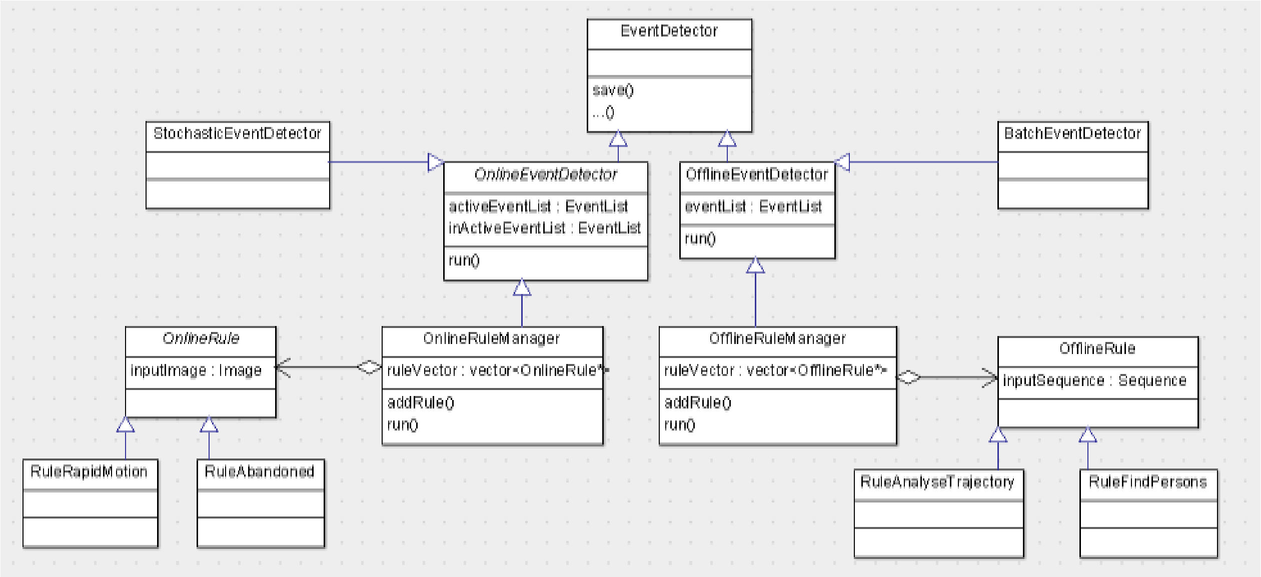


Figure 19 Basic UML diagram of the Rule Manager approach for video analytics. Online (for embedded) and offline (for cloud-level)

The Online Rule Manager is then a module that reads images in a sequential manner, and feeds these images into the Rules to update their status. Each Rule generates one or more events, which can be active or inactive.

Figure 19 shows the basic UML diagram of the proposed Rule Manager architecture. This picture illustrates the abstract base class EventDetector, and the derived classes OnlineEventDetector and OfflineEventDetector, which can be applied at the embedded level and the cloud level, respectively. These classes just define the type of information they can compute, i.e. the list of events, and in the case of the OnlineEventDetector, a list with active events and the list of inactive or completed events.

The implementation of such detectors can be done with Rule Managers: OnlineRuleManager and OfflineRuleManager, although other approaches could be used instead as illustrated by the StochasticEventDetector or the BatchEventDetector classes in the figure. These managers are structures that are simply configured to launch a number of Rules (OnlineRule and OfflineRule, for each case), and handle the birth, death and update of events.

As an example, the Rule Manager can be launched and connected to the following Rules:

- RuleRapidMotion: This rule computes and analyses the optical flow of the scene to learn motion patterns and trigger alarms when the motion is faster than normal.
- RuleTrespassing: Provided a region of interest that determines the existence of a sterile zone, this rule can determine when a trespasser is present in the given ROI using background extraction.
- RulePerson: Using human detection algorithms, this rule can trigger an alarm or message when a person is detected in the images.
- RuleGroupPersons: Detecting individuals can lead to the detection of groups when they get close.
- RuleOpposingFlow: Individuals walking in the opposite direction of the main flow, or in a direction that is forbidden.

The type of event is eventually defined by the available information about the scene, which can be provided during the installation stage of the system (e.g. set up a camera and define that is looking closely to a door), or automatically with the processing algorithms (e.g. using person detectors). Normally, the main actors of these scenes are the persons, therefore, events will typically relate to activities of individuals and/or groups.

The ability of the P-REACT system to remotely configure the embedded-platform naturally fits with the Rule Manager concept because different Rules and parameters can be defined dynamically from the cloud via the ESM.

The functional elements of this module are therefore:

- OnlineRuleManager: Which aggregates Rules and launches them.
- OnlineRule: Abstract class for implementing new Rules
- Set of preloaded Rules

The interfaces are:

- Input interface: Video stream in the form of a sequence of images received sequentially at a given input frame rate, required analysis type.
- Output interface: Updated list of evidences detected by the Rule Manager that can be exported to

JSON files at each frame time. A running log file is created with the information of all the detected events.

External entities:

- Contextual information: Camera and/or scene calibration
- Parameter/Config files: Preloaded or dynamically set by the P-REACT system through the ESM, to start or reset Rules and their behaviour.

The aforementioned approach for colour images will be developed collaboratively by CERTH and VICOMTECH and will result in a common video analytics module while a similar approach will be followed for depth images processing.

Audio Analytics

The functional view of the audio analytics module is shown in Figure 20. The continuous stream enters the lightweight analytics module on the embedded system. The audio is split into discrete blocks whose length will be determined during the algorithms performance in the simulated experiments while overlapping may also be introduced if it is found that assist in the overall performance of the system. Feature extraction then takes place discriminating between voiced and unvoiced characteristics. This step is essential as event detection involves both voiced (screaming) and unvoiced (e.g. loud noises,gun shooting, glass breaking) events. The extracted features are then combined into a single vector which serves as the input for a number of binary classifiers. Each classifier is trained to distinguish between two events/situations. The results of this stage are then forwarded to another module which is responsible for producing the overall classification label. This module takes into consideration correlations between the different categories and adds another level of results validation. The output of this module passes through a conditional test and if an alert is detected with a reasonable confidence the generation of the JSON object with alarm specific information is initiated and forwarded to the ES manager for further action.

The parameterisation of attributes and processes described will be determined during the implementation of the algorithms where a number of experiments testing the statistical validity of the results will be carried out in an effort to ensure effective operation.

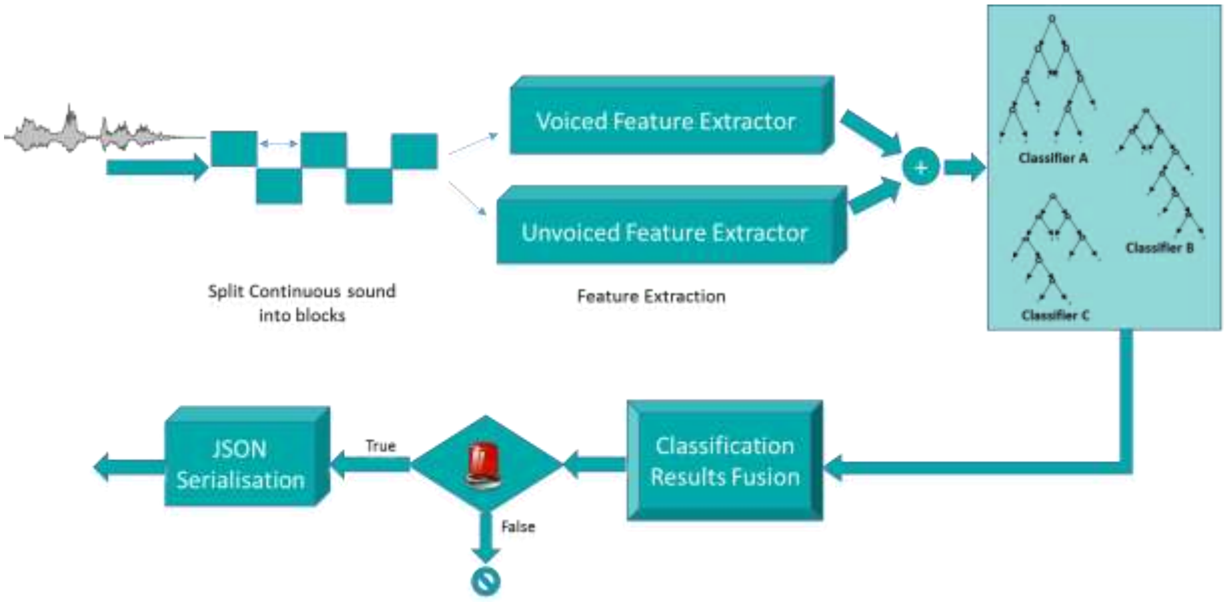


Figure 20 Embedded System Audio Analytics Functional View

3.1.4.Event Detection

The task of the event detection module will be to fuse the information from the various analytics modules of the Embedded System in order to make the final decision whether an abnormal event has occurred and notify the clip generator to create a clip and send it to the cloud. For the design of this module the limited computational resources of the Embedded System were taken into account, and avoided computationally intensive activity recognition algorithms that could hinder the real time operation of the system.

3.1.5.Clip Object / Clip Object Generator

In this section, Clip Generator's (CG) components and communication interfaces are presented. As the following figure depicts, CG comprises of two main processing units, namely the Clip Object Generator (COG) and the Integrator units; several external and internal communication interfaces; and one Database (Clip Storage) for storing the generated Clips.

More specifically, when a request for Clip Generation is received from the Embedded Systems Manager (ESM), COG gets any metadata produced from the local Analytics along with the Clip(s) Data Uris and produces the Clip Object (JSON metadata file). The Integrator unit merges the Clip Object and the Clip(s) Data to one Clip and then stores it to the Clip Storage, besides informing ESM for the newly created Clip. Until ESM emits a deactivation signal the process is repeated for the newly created Clip Data. It is noted that the Clips are stored for a finite amount of time (usually until they are uploaded to the Cloud).

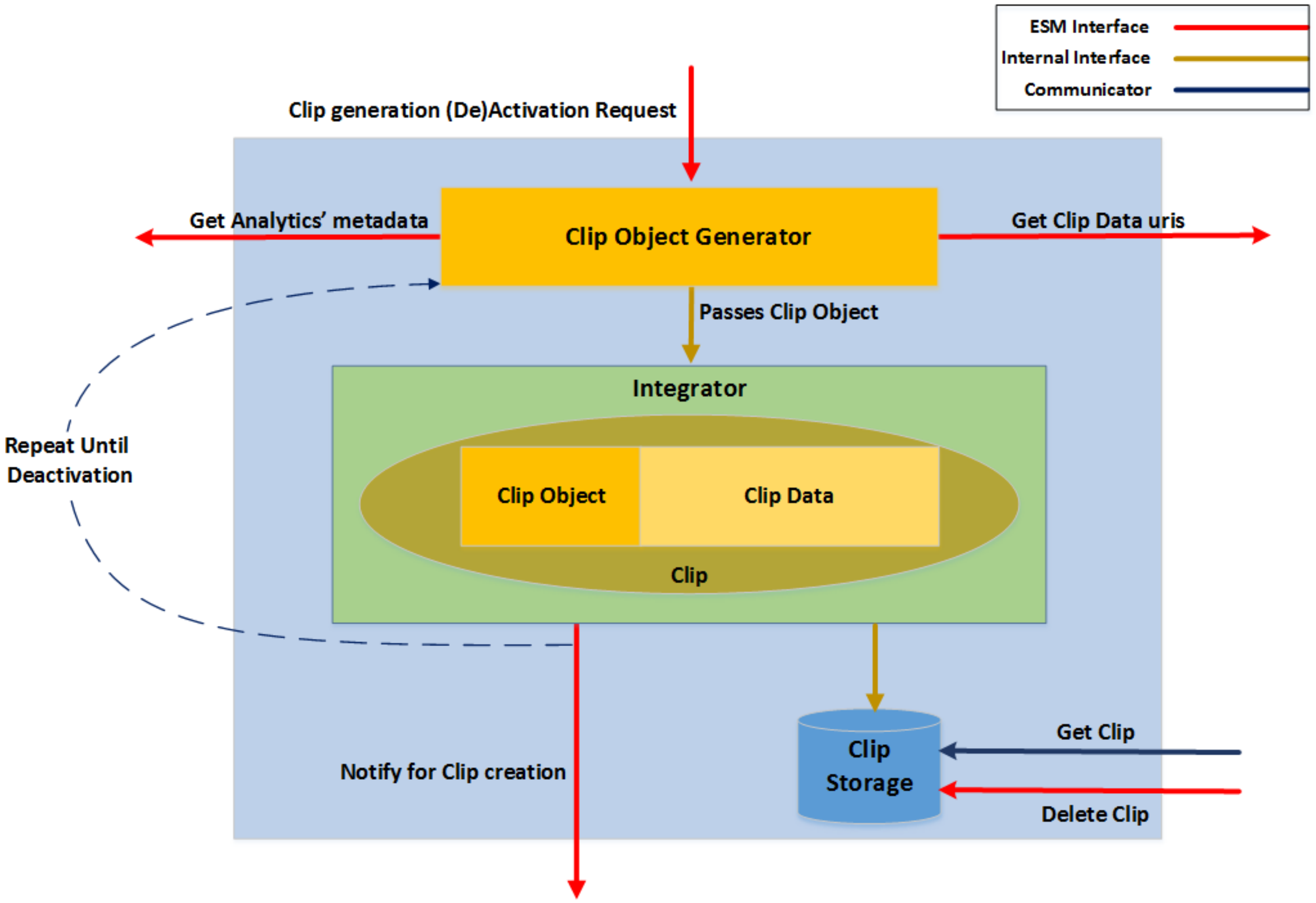


Figure 21 Clip Generator functional overview

In summary, the following interfaces are utilised for the communication with the rest of ES components:

1. Communicator:
   a. receive_clip_request(): This interface enables the Communicator to get the Clip from its storage and to upload it on the Cloud.
2. ESM:
   a. receive_generate_clip(): This interface allows CG to receive from ESM a request for initiating or halting Clip generation.
   b. send_clip_metadata(): This interface enables CG to send to ESM metadata for every generated Clip.
   c. send_clip_data_uris_request(): This interface allows CG to find out what Clip Data will be part of the next to be generated Clip.
   d. send_analytics_metadata_request(): This interface send to ESM a request for the Analytics metadata (if any) that will be part of the next to be generated Clip.
   e. receive_delete_clip(): This interface enables for the Clip deletion when it is not needed anymore.

3. Internal:

   a. store_clip(): It stores the Clip to the Clip Storage.

   b. pass_clip_object(): It passes the generated Clip object to the Integrator.

3.2. Communication Module

In this section, the functional view of Communicator module is presented. This module is responsible for the secure exchange of data between the Cloud and the Embedded Systems (ES). Communicator comprises of three main processing units, namely the Secure Communication Channel Manager (SCCM), the Clip Handler (CH) and the Encryption Manager (EM) units, but as the following two figures depict Communicator's functional view varies according to its operating place (Cloud or Embedded side). The following subsections elaborate on the functional view of each unit.

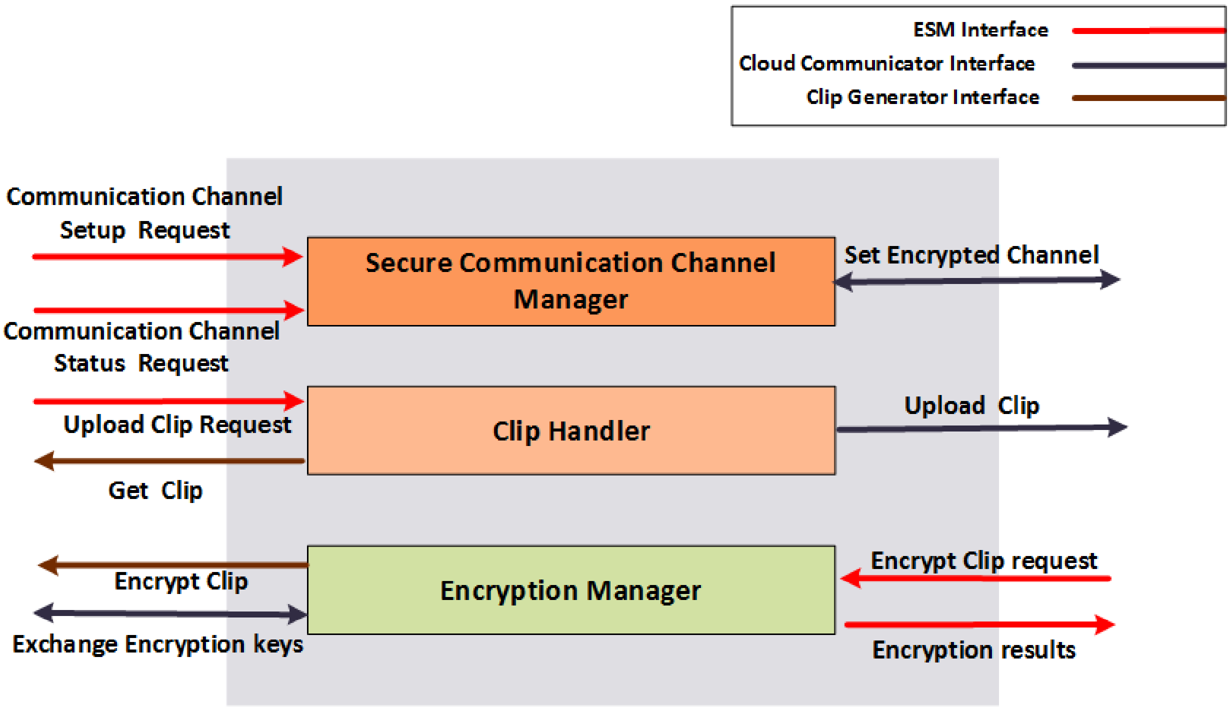


Figure 22 (Embedded System) Communicator - Functional overview

3.2.1. Encryption Manager

The Encryption Manager (EM) is activated when additional layers of protection are required for the data (e.g. Clips, messages) exchanged between the Cloud and the Embedded Systems. EM stores the relevant encryption keys (Private Keys, Public Keys), and handles key management operations (generation of new keys, revocation of old keys). When an Embedded System needs to send encrypted data, the Embedded System Manager requests from EM to encrypt the data prior forwarding them to the communication channel.

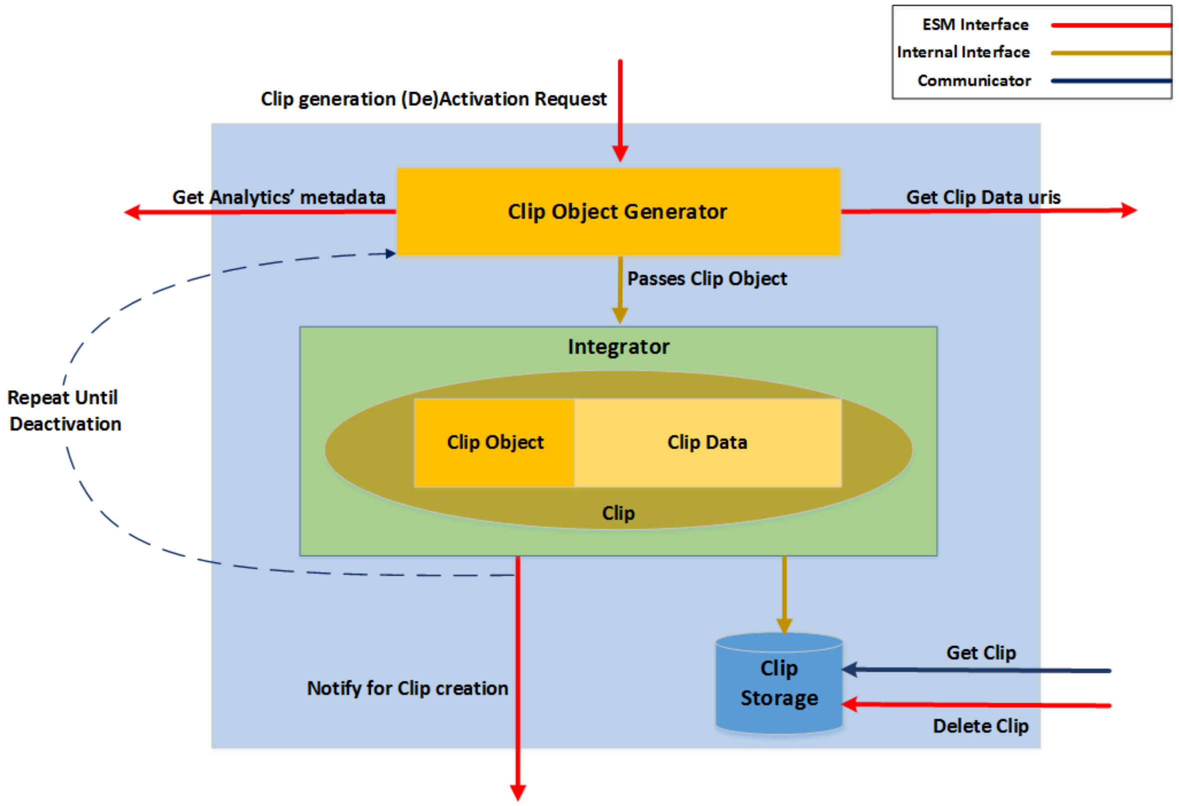


Figure 23 (Cloud) Communicator - Functional overview

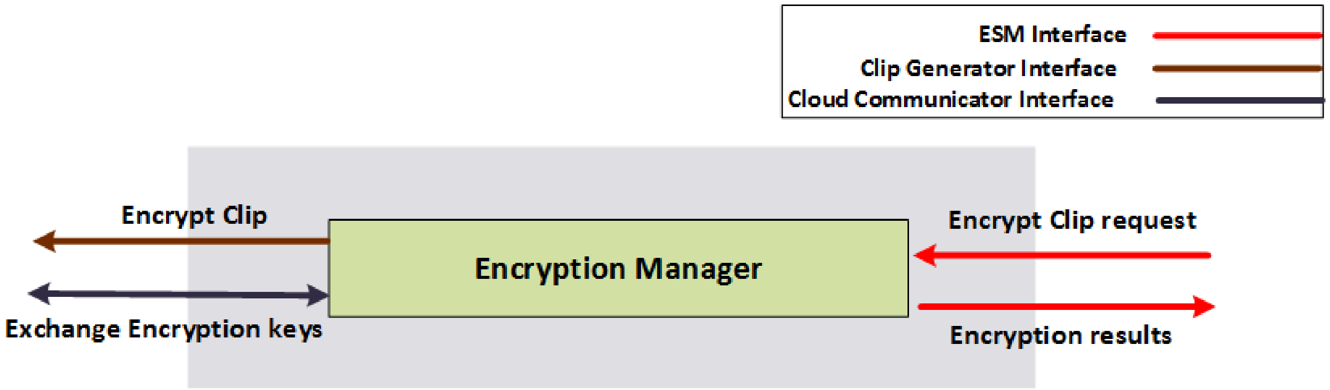


Figure 24 (Embedded System) Encryption Manager - Functional overview

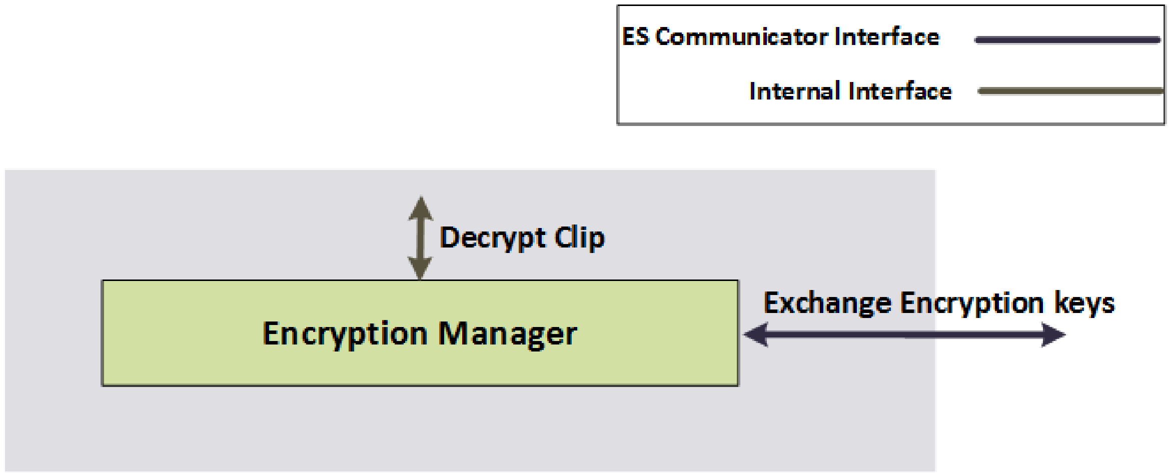


Figure 25 (Cloud) Encryption Manager - Functional overview

In summary the following interfaces are utilised for the Embedded System's Encryption manager:

1. ESM:
   a. receive_encrypt_clip_request(): This interface enables EM to receive from ESM a request for encrypting a specific Clip.
   b. send_encryption_result(): This interface enables EM to inform ESM for the encryption results.
2. Clip Generator:
   a. encrypt_clip(): This interface enables EM to perform the encryption operation.
3. Cloud Communicator:
   a. exchange_encryption_keys(): This interface enables the Cloud's and Embedded System's Communicators to get the appropriate keys for encrypting and decrypting the transferred Clips.

Whereas the following interfaces are utilised for the Cloud's Encryption manager:

1. Embedded System Communicator:
   a. exchange_encryption_keys(): This interface enables the Cloud's and Embedded System's Communicators to get the appropriate keys for encrypting and decrypting the transferred Clips.
2. Internal:
   a. decrypt_clip(): This interface enables for the decryption of the received encrypted Clips.

3.2.2.Secure Communications

In this section, Secure Communication Channel Manager (SCCM) unit is presented from functional point of view. SCCM is the unit that provides the secure communications between the Embedded System (ES) and the Cloud. To achieve this, SCCM builds on OpenVPN framework, supporting both Public Key Infrastructure (PKI) and pre-shared key approaches.

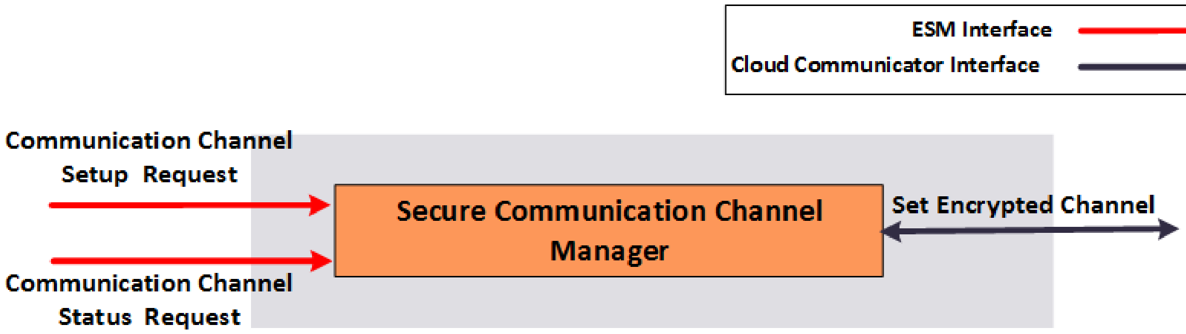

ESM Interface
Cloud Communicator Interface
Communication Channel Setup Request
Communication Channel Status Request
Secure Communication Channel Manager
Set Encrypted Channel

Figure 26 (Embedded System) SCCM - Functional overview

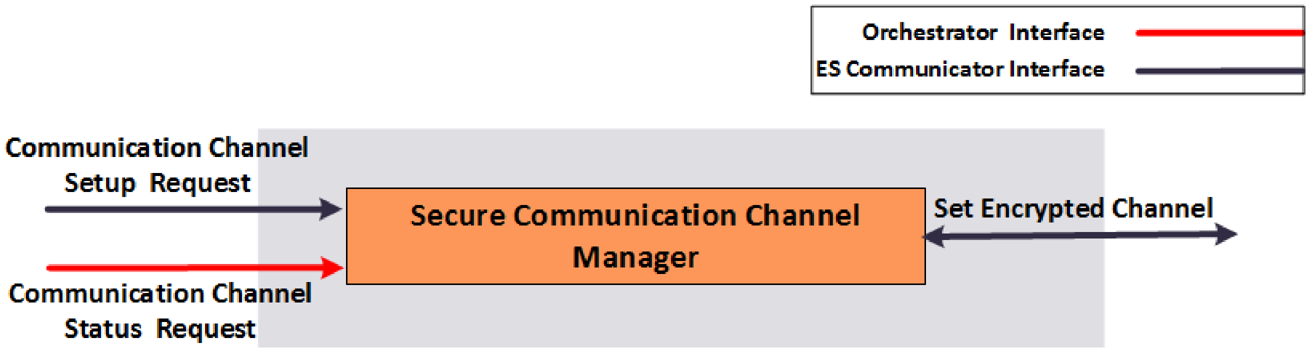

Orchestrator Interface
ES Communicator Interface
Communication Channel Setup Request
Communication Channel Status Request
Secure Communication Channel Manager
Set Encrypted Channel

Figure 27 (Cloud) SCCM - Functional overview

In summary the following interfaces are utilised for the Embedded System's SCCM:

1. ESM:
   a. receive_channel_setup_request(): This interface enables Communicator to receive from ESM a request for setting up the secure communication channel.
   b. receive_channel_status_request(): This interface enables Communicator to receive from ESM a request for the status of the secure communication channel.
2. Cloud:
   a. set_encrypted channel(): This interface initiates the negotiation and establishment of an encrypted channel with the Cloud based on OpenVPN framework.

Whereas the following interfaces are utilised for the Cloud's SCCM:

1. Embedded System Communicator:
   a. receive_channel_setup_request(): This interface enables Cloud's SCCM to receive from Embedded System's SCCM a request for setting up a secure communication channel.
   b. set_encrypted channel(): This interface initiates the negotiation and establishment of an encrypted channel with the Embedded System based on OpenVPN framework.
2. Orchestrator:
   a. receive_channel_status_request(): This interface enables SCCM to receive from Orchestrator a request for the status of the secure communication channel.

3.2.3. Clip Handler

This section presents the functional overview of Clip Handler, the module that manages the Clip transfer operation. Whenever a new Clip is generated at the Embedded System (ES) level, the Embedded System Manager (ESM) instructs Clip Handler to transfer the Clip to the Cloud. There the respective Clip Handler receives the Clip, if the Clip is encrypted requests a decryption from the Encryption Manager, and sends the contained Clip Data and Clip Object to the Cloud Storage and to the Orchestrator respectively.

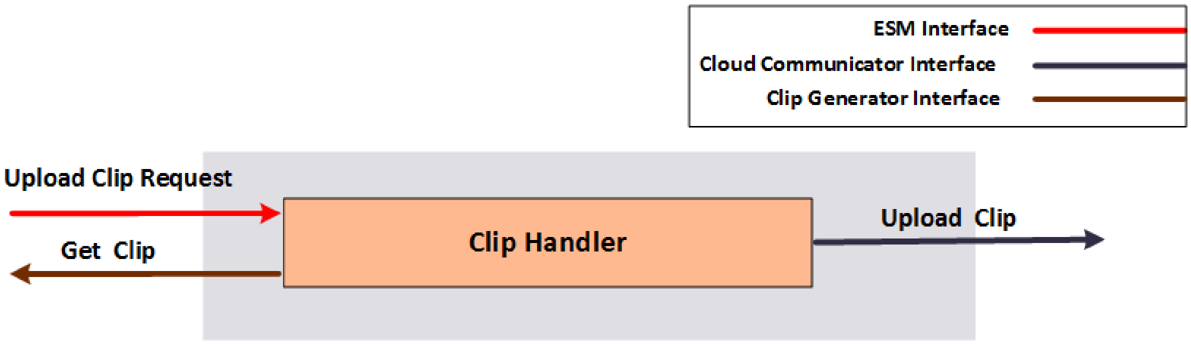


Figure 28 (Embedded System) Clip Handler - Functional overview

In summary the following interfaces are utilised for the Embedded System's Clip Handler:

1. ESM:
   a. receive_upload_clip_request(): This interface enables Clip Handler to receive from ESM a request for uploading a Clip to the Cloud.
2. Cloud Communicator:
   a. upload_clip(): This interface realises the Clip's transfer to the Cloud.
3. Clip Generator:

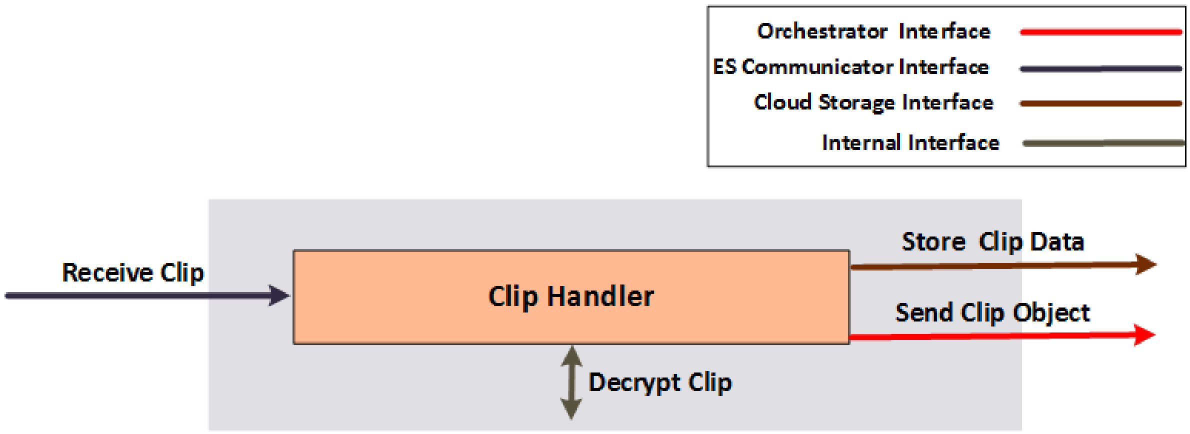


Figure 29 (Cloud) Clip Handler - Functional overview

   a. get_clip(): This interface retrieves the Clip form Clip Generator's Storage.

Whereas the following interfaces are utilised for the Cloud's Clip Handler:

1. Embedded System Communicator:
   a. receive_clip(): This interface enables Cloud's Clip Handler to receive a Clip.
   b. set_encrypted channel(): This interface initiates the negotiation and establishment of an encrypted channel with the Embedded System based on OpenVPN framework.
2. Orchestrator:
   a. send_clip_object(): This interface enables Clip Handler to send to Orchestrator a Clip Object.

3. Cloud Storage:
    a. store_clip_data(): This interface enables Clip Handler to store to Orchestrator the Clip Data.
4. Internal:
    a. decrypt_clip(): This interface enables for the decryption of an incoming encrypted Clip.

3.3. Cloud

3.3.1.Orchestration

The Orchestration module exposes an API to allow for communication between it and the other various modules that comprise the P-REACT system. It provides an extensive set of functions that allow management of clips, embedded systems and more. As the developments proceed, these functionalities will be extended in order to facilitate any new needs of the system. A detailed documentation of the currently supported functionalities of the API is presented in ANNEX I, while a graphical representation of the interfaces between the Orchestrator and the rest of the Cloud modules is presented in Figure 30.

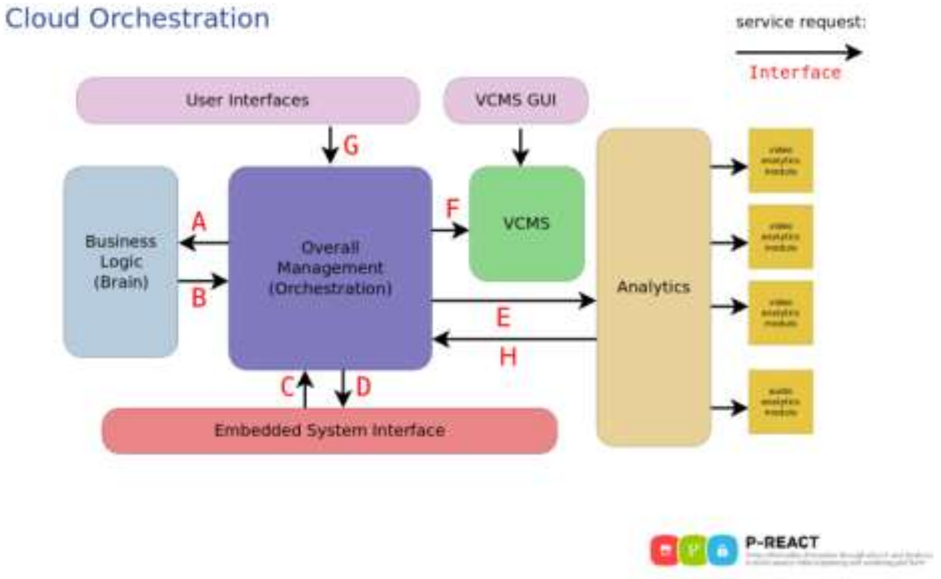


Figure 30 Cloud module organisation showing interface relationships

3.3.2.Situational Awareness

Decision and control is performed by the Business logic component shown in Figure 30. This component is responsible for receiving a new clip object, decide on what type of analytics should be used in each case, receive the result of the analytics modules and determine what further actions are required.

3.3.3. Analytics Modules

As was the case in the embedded system, on the cloud side the analytics module is divided into two main categories: (1) video and (2) audio.

For the video analytics, the methods that will be used in the cloud are the same ones as in the embedded system but with different configuration (since on the cloud there are more resources available) that will allow a more detailed analysis, in order to verify the existence of an event (thus eliminating false alarms), plus a set of more advanced methods that will allow the identification of the suspects in other clips (either stored in the VCMS or coming from other embedded systems). These methods include:

1. Gait analysis: Identification of a person from his/hers walking pattern
2. Face analysis: Identification of a person from his/hers facial characteristics
3. Geometry based identification: that will identify a person based on his/hers upper body geometry

Audio analytics on the cloud will be used in order to validate the existence of an event thus eliminating false alarms.

Video Analytics

The video analytics module residing on the cloud platform will receive its input from the various Embedded Systems hooked on the P-REACT platform through a secure channel. Its main aim will be to identify offenders without resorting to techniques that violate the privacy of the general public but also to validate the existence of the event that was reported. Its functional view is shown in Figure 31.

To perform its task the video analytics module will rely on RGB and/or Depth information to perform gait, face and appearance based recognition on the corresponding components. Then, using the produced information, it will search on its archive of past abnormal events as well as in new clips coming from neighbouring embedded systems in order to identify the people involved in the current event. The results of its operation will be sent to the decision and control module where an analyst will be notified as to the video content management system module.

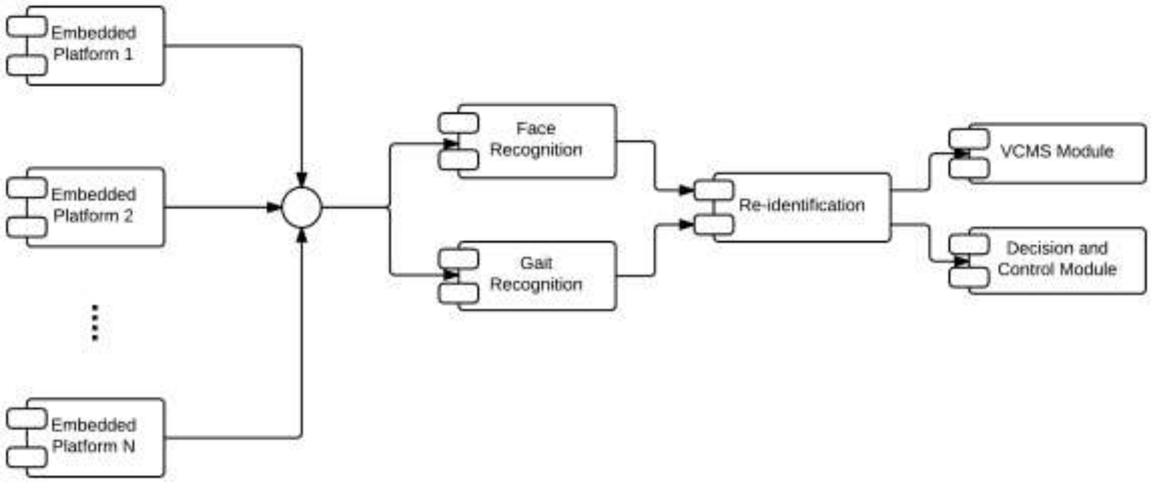


Figure 31 Functional view of the video analytics module on the cloud

The information arriving to the video analytics at the cloud-level are video clips which contain both visual information (the images) and additional metadata with the result of the process of the embedded-level plus other administrative metadata (location, time, etc.).

The analysis can be done according to at least two main functionalities: (i) search and (ii) re-analyse.

The search analysis can receive a single sample clip, and return a map to existing clips stored in the database or coming from other embedded systems with associated match indicators that determine the correlation or similarity with the sample.

Regarding the re-analyse capability, the same Rule Manager concept can be applied as in the embedded-level. The main difference is that the Rule Manager can contain Rule execution configurations not necessarily sequential in time, and that Rules can contain more complex analysis of the information.

The functional elements of this module are therefore:

- Search functionality:
    - Video descriptor: Converts the given clip into a descriptor using feature extractors.
    - Classifier: Compares two feature vectors using a classifier to determine the match level.
    - Classifier trainer: This is the module that learns models from the stored data and is updated when new data is stored.
- Re-analysis or enhanced analysis
    - OfflineRuleManager: Equivalent version of the Rule Manager at the embedded-level, but with an offline nature.
    - OfflineRule: Base class from which specific Rules can derive.
    - Set of preloaded Rules: For instance with specific algorithms to verify the presence of objects according to their appearance (e.g. persons, faces),

The interfaces are:

- Input interface: Input video clip with metadata, analysis type to be processed (defined by the Business Logic)
- Output interface: Updated metadata for clips; list of matching levels and links to existing videos in dataset; trained models.

External entities:

- Video clip database: With the video and metadata (cloud storage and VCMS databases)
- Trained classifiers/models: The classifiers train models that are stored in the cloud (Cloud Storage), and updated when new clips are available.
- Parameter/Config files: Preloaded or dynamically set by the P-REACT system, to start or reset Rules and their behaviour.

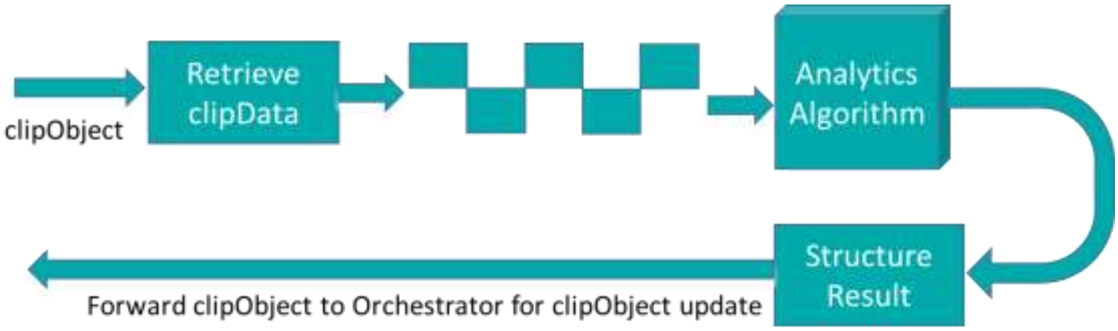


Figure 32 Cloud Side Audio Analytics Functional View

Audio Analytics

The audio analytics on the cloud accepts an analysis request with the clipObject of interest. The analytics module will proceed with the retrieval of the data by reading from the URI location specified in the clipData field of the clipObject. Depending on the specified analytics algorithm ID, the analytics module proceeds with the analysis of the data with that specific algorithm. Upon the completion of analysis, the output is structured according to the specifications and the analytics module proceeds communication with the orchestrator by providing the input clipObject as well as the analysis results. This process is depicted in Figure 32

3.3.4.VCMS

The VCMS is commonly called by the Orchestration module through its API and available methods. The implemented API supports a number of different functions for managing users and clip objects. An extensive list of the supported functions and their description is provided in ANNEX II

4. Development View

For the development viewpoint we describe how we handle revision control of source code files in a centralized repository and aspects of the development process to enhance the quality of the software. Further we provide an overview of the modules technical requirements and dependencies.

4.1. Key Components Development

In this section we provide an overview of the responsible partners and the technical requirements and dependencies for each key component. In Figure 33, the responsible partner of each of the components of the previously presented system architecture is shown.

Although the system’s architecture is modular, and each module seems to be independent, there is a very strong interaction during the development, and this is due to the fact that each module determines and depends on various parameters and requirements for one or more other modules.

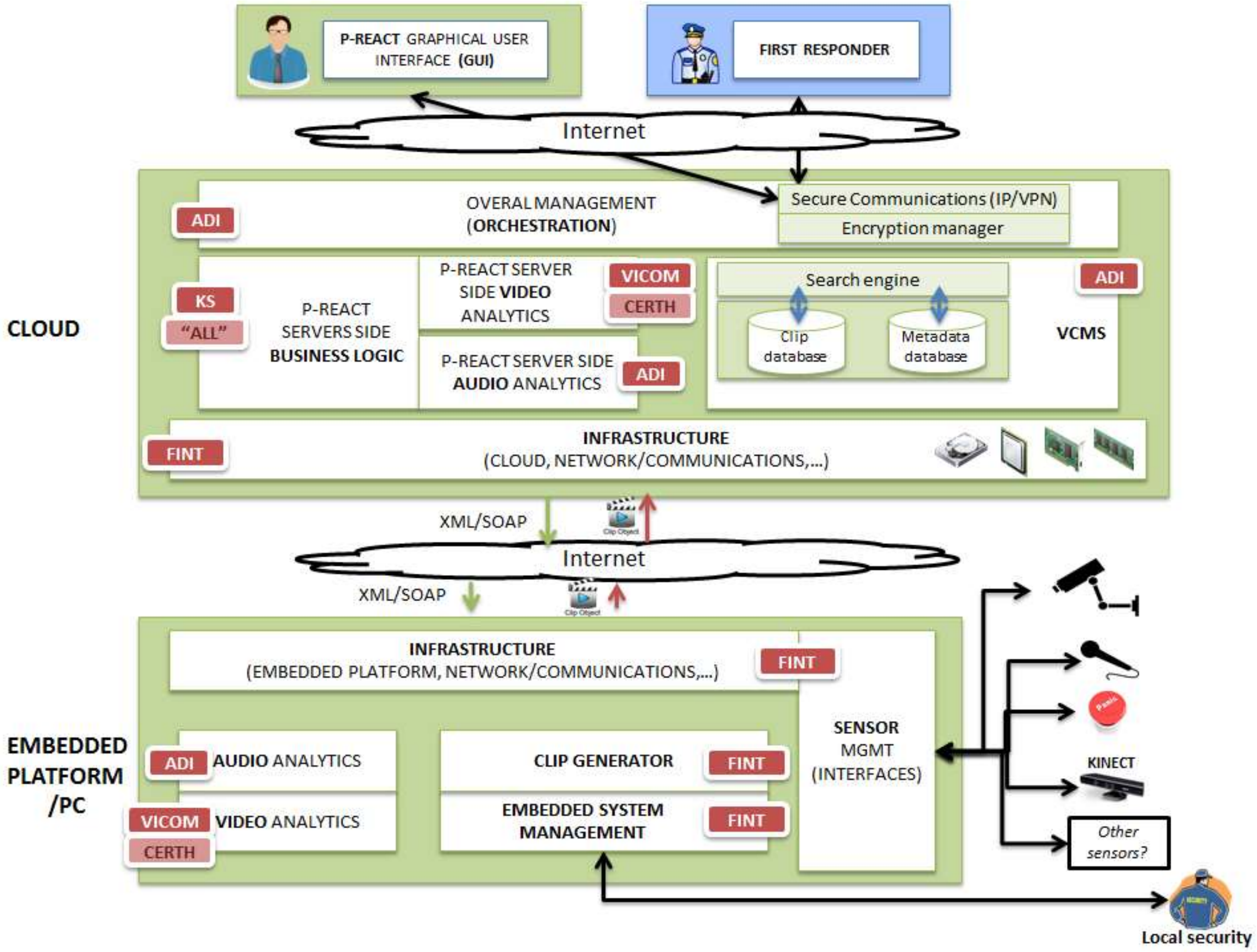

P-REACT GRAPHICAL USER INTERFACE (GUI)
FIRST RESPONDER
Internet
ADI
OVERAL MANAGEMENT (ORCHESTRATION)
Secure Communications (IP/VPN)
Encryption manager
CLOUD
KS
"ALL"
P-REACT SERVERS SIDE BUSINESS LOGIC
P-REACT SERVER SIDE VIDEO ANALYTICS
VICOM
CERTH
P-REACT SERVER SIDE AUDIO ANALYTICS
ADI
Search engine
Clip database
Metadata database
ADI
VCMS
FINT
INFRASTRUCTURE (CLOUD, NETWORK/COMMUNICATIONS,...)
XML/SOAP
Clip Object
Internet
XML/SOAP
Clip Object
INFRASTRUCTURE (EMBEDDED PLATFORM, NETWORK/COMMUNICATIONS,...)
FINT
SENSOR MGMT (INTERFACES)
EMBEDDED PLATFORM /PC
ADI
AUDIO ANALYTICS
CLIP GENERATOR
FINT
VICOM
VIDEO ANALYTICS
CERTH
EMBEDDED SYSTEM MANAGEMENT
FINT
KINECT
Other sensors?
Local security

Figure 33 System Architecture with components responsible partners

Starting from the embedded system, Future Intelligence (FINT) is responsible for developing the main components, which include the system manager, the sensors manager as well as the clip generator and the communications module. In order to achieve this goal, the hardware requirements for the sensors will first be defined, as well as the software requirements for the analytics modules (both video and audio). The latter also determined the hardware specifications of the embedded system. Since the requirements for the audio analysis are much less than for video, the video analytics module plays a more important role in determining the hardware requirements. In the P-REACT system, both colour and depth sensors are going to be used. The colour sensors in general require less computing power for analysing their provided data (colour images) than the depth sensors. For this, two variations of the embedded system are going to be build: a basic one that will be utilized with colour cameras and audio, and a more advanced one, that will be used for depth analysis and audio.

The audio analytics module will be developed by ADITESS while the video analytics by Vicomtech and Center for Research and Technology Hellas (CERTH). More specifically, Vicomtech will focus on a video analytics module for colour cameras and CERTH will focus on analysing depth information, while keeping a high level of collaboration.

On the cloud side, FINT is responsible for developing the main infrastructure, which includes the actual cloud system, the network and the communications modules. This requires input from the partners that are responsible for the rest of the modules. More specifically, the Orchestration module, (see section 2.2.1 for description) is built by ADITESS, the Business Logic component (section 2.2.2) by Kinesense (KS), the video analytics by Vicomtech and CERTH and the audio analytics (section2.2.3) by ADITESS. ADITESS is also responsible for developing the Video Content Management System as well as the cloud storage where the actual clip data will be stored. As was the case in the embedded system, the modules of the cloud system are also not independent from a development point of view.

The development of the various components of the system started in project month 5, so by the time the present deliverable was prepared (Project Month 10) the modules are still in beta versions. Despite being still in beta version, a complete integrated test including most of the modules has been performed, identifying the weak spots and ensuring that collaboration between the responsible partners is at a very high level and the requirements set by each module are met.

4.2. Source Code and Configuration Management

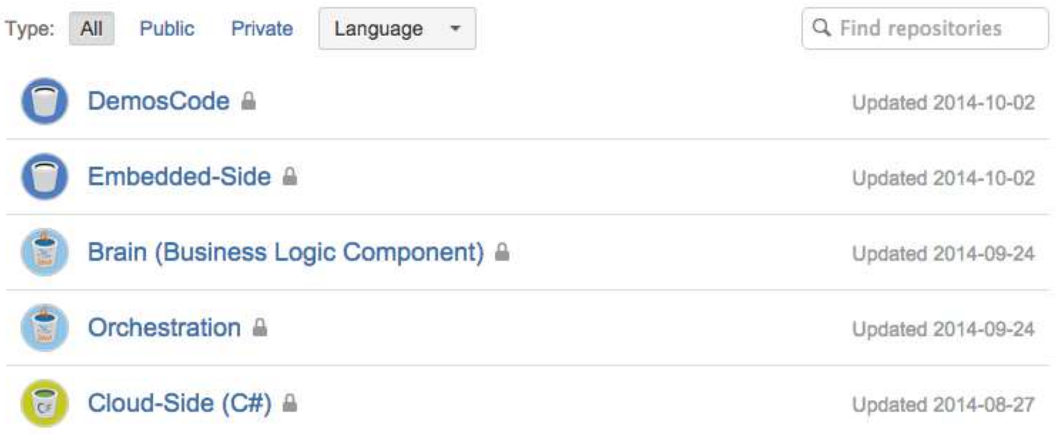
4.2.1.Source Code Management with Git

During the development phase of the project, the revision control system 'Git' will be used to manage all changes to code. Git is a distributed revision control system with an emphasis on speed, data integrity, and support for distributed, non-linear workflows. A centralized workflow will be adopted while developing for the project using a central repository to serve as the single point-of-entry for all changes to the project. Developers have to create a local copy of the central repository. That is accomplished via the git clone command.

While working on a local copy of the project, it is possible to view the current state of the repository via the git status command, add some files to your staging area via the git add command and commit changes via the git commit command.

In their own local copies of the project, developers can edit files and commit changes. Once they finish committing changes, which are stored locally, they can publish changes to the official project by pushing their changes to the central repository. Pushing changes to the central repository is accomplished via the git push command.

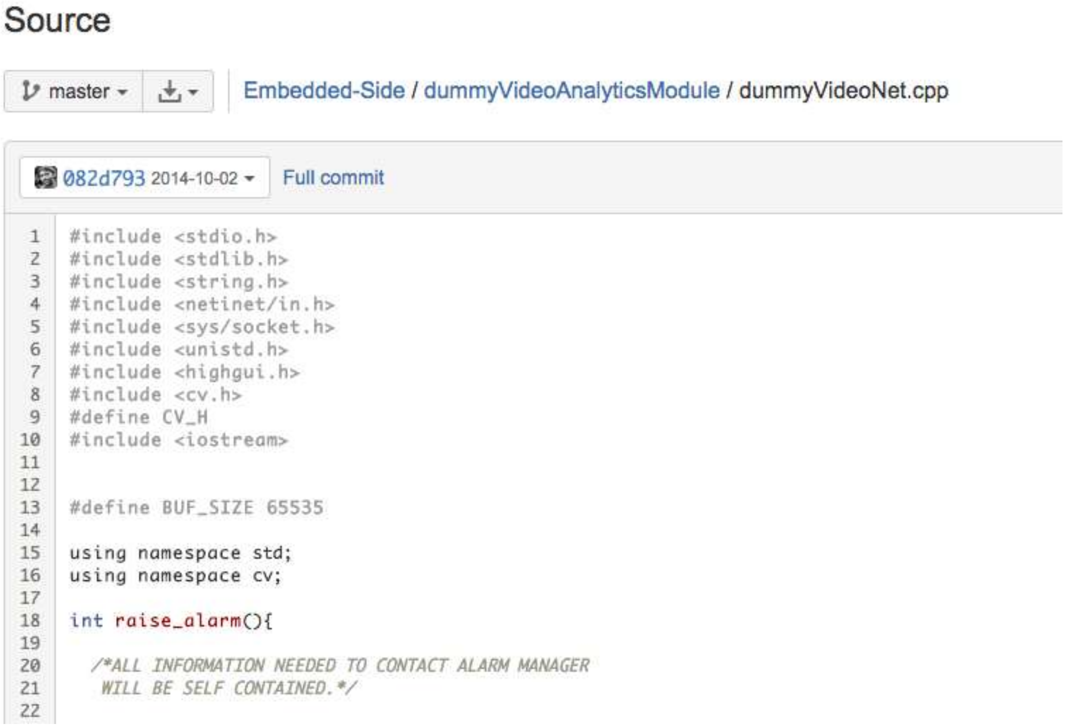
As illustrated in the next figure, all current code repositories (DemosCode, Embedded-Side, Brain (Business Logic Component), Orchestration and Cloud-Side (C#)) is hosted online on Bitbucket using a private centralized team account which makes managing repositories easier for multiple developers.



DemosCode	Updated 2014-10-02
Embedded-Side	Updated 2014-10-02
Brain (Business Logic Component)	Updated 2014-09-24
Orchestration	Updated 2014-09-24
Cloud-Side (C#)	Updated 2014-08-27

Figure 34 Bitbucket P-REACT team overview showing all available repositories

Bitbucket provides, among others, unlimited private repositories, online browsing and search functionalities for all the repositories as well as simplified user management.



Source
master
Embedded-Side / dummyVideoAnalyticsModule / dummyVideoNet.cpp
082d793 2014-10-02
Full commit

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <unistd.h>
#include <highgui.h>
#include <cv.h>
#define CV_H
#include <iostream>


#define BUF_SIZE 65535

using namespace std;
using namespace cv;

int raise_alarm(){

  /*ALL INFORMATION NEEDED TO CONTACT ALARM MANAGER
   WILL BE SELF CONTAINED.*/

```

Figure 35 Bitbucket online code snippet browsing

4.2.2.Coding Conventions

The coding conventions that have been decided so far are the following:

- Information between different modules will be exchanged using JSON objects
- GUID’s will be used as object identifiers
- Date/Time fields will be using ISO 8601 format (YYYY-MM-DDYHH-MM-SSTZ) or UNIX timestamps.

More coding conventions will be added as the development continues, and the complete list will be included in System Integration report Deliverable (D4.1)

4.2.3. Use of Existing Software

In this paragraph, the existing software that will be used in the P-REACT's solution is presented. This includes the libraries, Application Programming Interfaces (API's) and Software Development Kits (SDK's) that are required to develop the project's solution.

- **OpenCV,** open source computer vision library, (used for basic image processing and computer vision algorithms) http://opencv.org/downloads.html
- **Viulib**, Vicomtech's Vision and Image Understanding Library, (used for advanced computer vision algorithms) http://www.vicomtech.org/viulib
- **Gstreamer**, open source video and audio capturing, (used for capturing video and camera streams) http://gstreamer.freedesktop.org/
- **Videoman**, open source video capture and rendering library, (used for capturing and rendering video streams) http://videomanlib.sourceforge.net/
- **tinyxml2**, open source lightweight parsing library for XML files, (used to easy read/write parameter files) http://www.grinninglizard.com/tinyxml2/
- **Qt,** cross-platform framework, (used in GUI applications for calibration tools) http://qt-project.org/
- **Gstreamer,** open source multimedia framework, (used for getting, streaming and saving the sensors' A/V data) http://gstreamer.freedesktop.org/
- **Glib**, general-purpose utility library (used for the advanced data types and string/file utilities it provides) https://developer.gnome.org/glib/2.42/
- **GIO**, Virtual File system and Networking API, (used for taking into account the exchanging info between the SW modules) https://developer.gnome.org/gio/stable/
- **JSON-GLib**, JSON library (used for reading/writing from/to the JSON files) https://developer.gnome.org/json-glib/stable/
- **libtar**, library for handling tar files, (used for the creation of the Clip object (metadata+sensors' data)) http://www.feep.net/libtar/
- **Microsoft Kinect SDK,** used to enable data capture and processing from the Kinect v2 sensor
- **Spring**, deployed for the implementation of web services (see http://spring.io/guides/gs/rest-service/)
- **Weka**, for machine learning (see http://www.cs.waikato.ac.nz/ml/weka/)

Regarding the operating system, the Embedded System will be based on Linux (Debian, www.debian.org), while an advanced one will be based on Windows 8 (required for the use of the Kinect v2 sensor).

On the cloud side, a number of virtual operating systems will be deployed, in order to accommodate the various needs of the architecture. The virtual OS's will be Linux based (Debian), as well as Microsoft Windows if required (in order to use the Kinect SDK)

5. Deployment View

The deployment view focuses on aspects of the system that are important after the system has been tested and is ready to go into live operation. This view defines the physical environment in which the system is intended to run, including:

- Required hardware environment (e.g., processing nodes, network interconnections, etc.)
- Technical environment requirements for each node
- Mapping of software elements to the runtime environment
- Third-party software requirements
- Network requirements

5.1. Hardware Requirements

In order for the P-REACT system to be installed locally (in the premises of gas stations, small shops, public transportation infrastructure etc.) an Embedded System along with a camera and/or a microphone is required. The Embedded System that is been used currently (in the development phase) is a low cost system that consists of an A10 Cortex-A8 CPU running at 1GHz and a Mali-400 GPU. It is equipped with 512MB of RAM and 4GB of NAND Flash storage. It hosts two usb2.0 ports for camera and microphone connectivity and a 100Mbit Ethernet port for communication purposes. This unit provides a balance between cost and performance and is compliant with the P-REACT concept of using a low cost Embedded System for on-site analysis.

Besides the aforementioned basic version of the Embedded System, another version will be used, in order to facilitate a depth sensor, providing more advanced event detection and analysis. This system has to be much more powerful than the basic version, since processing depth information requires more processing power. Despite this, the usage of the new Microsoft Kinect sensor (version 2) imposes even more restrictions; the minimum hardware requirements for the Kinect v2 sensor are 64bit dual core CPU running at 3.1GHz, 4GB of RAM, a USB3.0 port and a DirectX 11 capable GPU. A system with these specifications cannot be considered as a “low-cost” system at the present time, but in the future it will be much more affordable, and the P-REACT system can be much more effective with the utilization of depth sensors. Finally, while the developments proceed, other depth sensors will be examined, in order to provide more flexible solutions.

On the sensors’ side of hardware requirements, the cameras that are going to be used need to provide at least VGA resolution (640x480 pixels) at 30fps and is desired (but not required) to have a night mode (infra-red) in order to be able to function properly during night. For the microphone 16-bit audio at a sampling rate of 16 kHz is adequate.

On the cloud side, an OpenStack computing node will be used that will host a number of virtual hosts for the various cloud components (Figure 36). The following table displays the current list of virtual hosts that are being used with respect to the system architecture that was presented in Section 2.

Table 1 Hardware requirements for the Virtual OS's

Virtual Host	OS	Architecture	VCPUs	RAM (GB)	Storage Capacity
Orchestrator	Ubuntu	x86_64	1	2	20GB
Business Logic	Ubuntu	x86_64	2	4	20GB
Video Analytics	Ubuntu	x86_64	4	8	20GB
Audio Analytics	Ubuntu	x86_64	1	2	20GB
VCMS	Ubuntu	x86_64	2	4	20GB
Cloud-Embedded interfaces	Ubuntu	x86_64	2	2	20GB

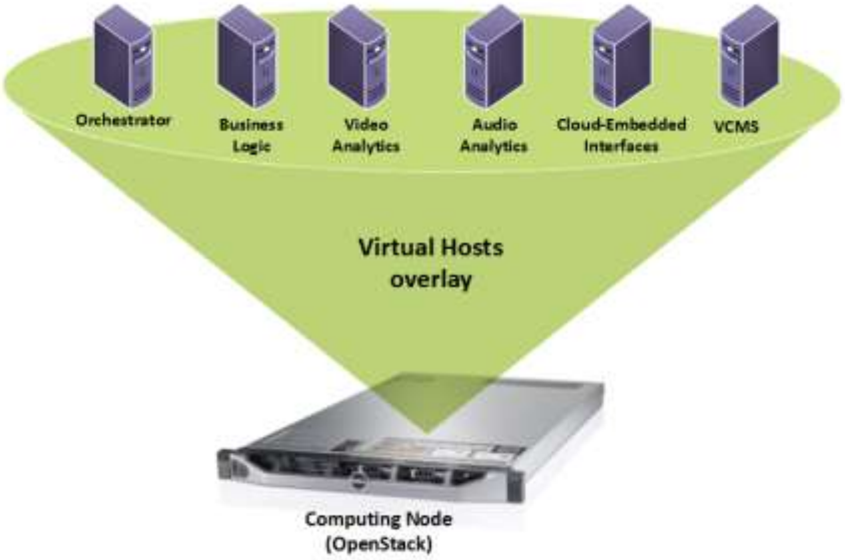


Figure 36 Virtual Hosts on P-REACT's cloud

5.2. Existing Software and Hardware Requirements

The P-REACT system is designed as a stand-alone system, so no major existing hardware requirements exist. However, it can be used with existing sensors. This means that if a small shop or a gas station already has a surveillance system with one or more cameras, the P-REACT system can use these cameras, and not require the purchase of a new one, thus reducing the cost for the shop owner. The same applies for the microphones. Also, the Embedded System could be replaced by an existing system on site, but this can be non-trivial due to operating system requirements, the Embedded System uses Debian Linux, or the advanced system with the Kinect requires Microsoft Windows 8.1, an existing computer might not have these.

5.3. Network Requirements

A network connection is required between the Embedded System and the cloud for both sending sensor data (clips) from the Embedded System to the cloud, but also for the cloud to be able to monitor and control the Embedded System. The network requirements for the cloud side (for monitoring and controlling purposes) are minimal, since mostly control signals need to pass through. On the other hand, the Embedded System will need to send data from the sensors that include images, videos, depth information and audio clips along with metadata. A typical clip object can be about 10MB or more in size, and needs to be transmitted relatively fast in order for the processing on the cloud side to complete as soon as possible leading to the prevention or investigation of the detected incident. A typical ADSL line with 1Mbit upload will require 8 seconds to upload 1Mbyte of data, which results in about 80seconds for a typical clip. This should be the minimum requirement for the system to perform adequately.

6. System Concept and Structures

6.1. Introduction

This Section presents the broad functional areas of both the embedded and cloud system of the P-REACT platform through context diagrams. Moreover, the high-level picture of each system's boundaries and its adjacent external entities is described. The context diagram is necessary to introduce the readers to the context in which the architectural elements that compose the P-REACT platform will take place. It should be noticed that the context diagrams can be considered as a "living" section of the architecture in order to keep it up-to-date with high-level changes.

In the previous Sections the conceptual architecture of the embedded and cloud system was analysed from different viewpoints (functional, development, deployment). In the following sections, both the static and the dynamic behaviour of the systems are presented. The static behaviour is depicted through the use of context diagrams, whereas the dynamic behaviour is illustrated through use case diagrams that are identified and analysed.

6.1.1.System Context Diagram Definition and Scope

Context diagrams outline and illustrate how the system operates at high level and give an overall picture of the system boundaries and its adjacent external entities. Thus, by providing them, the key stakeholders may obtain a clear picture of the context in which the system analysis will take place.

Summarizing, the context diagram purpose and aim will be to introduce and provide the necessary information for the logical decomposition (static analysis) of the architectural elements that comprise the P-REACT system.

6.1.2. Architectural Elements Perspectives

By looking at the system architecture from various viewpoints, provides meaningful information to the architecture derivation process and helps define the various architectural structures. However, to broaden the modularity, reliability and credibility of the system, it is useful to outline and consider specific quality properties, outlined in table 2 below, during the final stages of the architecture definition process. Towards defining the static and dynamic structures of the P-REACT framework and its main architectural elements, the architectural perspective is also considered. In this section, several quality properties are addressed for all architectural elements of the system, as these are outlined in the following table:

Perspective	***Desired Quality***
General Purpose	
Performance and Scalability	The ability of the system as a whole including its architectural elements to predictably execute within its mandated performance that cope with system requirements and is able to handle increased processing volumes of information.
Availability and Resilience	The ability of the system as a whole to be fully or partly operational as and when required and to effectively handle failures on all levels (hardware, software) that could potential affect system availability and credibility.
Security	The capacity of the system to reliably and effectively control, monitor and additional audit if the policies defined are met (e.g. what actions on what assets/resources) and to be able to recover from failures in security-related attacks.
Evolution	The capability of the system and its architectural elements to be flexible enough in the case of non-foreseen changes during deployment or installation process.
Internationalization	The capacity of the system to be independent from any particular language, country or cultural group.
Additional Perspectives to cope with P-REACT non-functional requirements	
Maintenance	The ability of the system to comply with coding guidelines and standards. Includes also the functionality that needs to be provided to support maintenance and administration of the system during its operational phase.
Privacy & Regulation	The ability of the system and its architectural elements to

	conform to national and international laws, policies and other rules and standards.
Usability	The ease with which key stakeholders of a system are capable to work effectively and to interact with it in a user-friendly way.

Table 2: Quality properties and perspectives that designers shall consider during the Architecture Definition Process.

For each of the aforementioned perspectives, the importance on the embedded and cloud system of the P-REACT framework may vary and the benefits of addressing them in both are essential towards providing a common sense of concerns that shall guide the architectural elements definition process and their later implementation and deployment to the validation and integration phase. In this respect, it is anticipated that by addressing in the architecture definition process the importance of the aforementioned perspectives will further help the later decision making (implementation, deployment and operational phases). For both the embedded and cloud system, a table will be provided, in order to ensure that all concerns and non-functional requirements are addressed and to exhibit what quality properties are considered within the system.

6.1.3.Use Cases Definitions and main sequence diagrams

Within P-REACT, towards fulfilling the objective of presenting the dynamic behaviour of the system, a general-purpose standard modelling language has been used, namely the Unified Modelling Language (UML). As defined in the standard, the UML is a “graphical language for visualizing, specifying, documenting the artefacts and architectural elements of a software-intensive system”. The UML provides a number of standard schematic, sketching and diagramming notations and encapsulates best practices in software design process into a standard and easily extensible notation. In this respect, architect designers can use notations such as use cases, class diagrams, sequence diagrams and activity diagrams. Moreover, it has several mechanisms, such as stereotyping, thus allowing architecture designers and modellers to tailor or elaborate the language to suit their needs and circumstances.

Towards analysing the dynamic behaviour of the P-REACT framework and drafting the detailed specifications of each architectural element, the basic use cases and application scenarios are elaborated in the following paragraphs with the corresponding diagrams (e.g. UML sequence diagrams) taking into account the overall user and system requirements.

The purpose of this analysis will be to illustrate the main functionalities that will be provided by the P-REACT system components and sub-systems. The exact and detailed module specifications for each architectural element were presented in Section 3.

General characterization and definition of use cases

A use case (UC) typically illustrates a discrete unit of interaction between a user (human or machine) and the system. In the context of system design and engineering, UC is a description of a system's behaviour as it responds to a requirement or request originating from outside of the system, i.e. a use case capitalizes on "who" can do "what" with the system in question.

It describes the functional requirements of the system, the external objects at the system boundary and the response of the system. For instance, a use case for the Embedded System could be to detect an abnormal event and send the respective clip object to the cloud system. It should be noticed that each system's functionality, must always be completed for the use case to terminate or finalize. Moreover, a use case may 'include' another use case's functionality or 'extend' another use case with its own behaviour (multi-level approach).

Summarizing, a UC (or a set of connected use cases) has the following peculiarities:

- Illustrates and organizes in a high-level the functional requirements of a system.
- Specifies the ultimate goals of the system/actor (user or machine) interactions.
- Presents unique scenarios from triggered events to tangible objectives and goals.
- Illustrates the main flow of events.

Application/Business Scenarios Scope and relation to Use Cases and Actors

In most cases, the use case process is used to identify and capture system's dynamic behavioural requirements by providing in details scenario-driven narratives. The business scenarios, which are related to the actual pilot sites foreseen in the project, are compatible to the use cases presented in the following sections. Moreover, the business cases can be easily abstracted to accommodate different domains (e.g. public places, restricted areas etc.) using the Use Cases identified.

In this report, focus is given on the key stakeholders (actors) that play significant role to the application scenarios identified for the P-REACT system. Use Cases are typically related to 'actors'. An actor is a human or machine entity that interacts with the system to perform meaningful work.

UML Use Case, Sequence and Activity diagrams Use Case Diagram

Use Case Diagram

A use case diagram in its simplified view is a graphical representation of a use case and can graphically visualize the different types of users of a system and the various ways that they interact with the system.

Sequence Diagram

A Sequence Diagram is typically used to illustrate the following:

- Usage scenario, which is a description of a potential way a subsystem, is utilized in the actual system in question. The logic of a usage scenario may be part of a use case (or it can extend another) and can be illustrated as a system-level sequence diagram. The logic of a usage scenario may also be a pass through the logic contained in several use cases.

- The logic of methods, in which sequence diagrams, are utilized to explore the logic of a complex operation, function, or procedure.
- The logic of services, in which a provided service is effectively a high-level method, often one that can be invoked by a wide variety of actors. This includes web-services as well as business transactions and interchanged messages implemented by a variety of technologies such as SOAP, TCP/IP XML messages, etc.

Within this document focus is given to the functional description and to providing UML sequence diagrams that visualize the dynamic architectural elements' interaction and behaviour.

Activity Diagram

An Activity Diagram is used to describe the business and operational step-by-step workflows of the involved architectural elements in a system. The diagrams illustrate the overall flow of control and can be considered as a special form of the state-chart diagram that is suitable for expressing the activity execution flow. It is commonly used for objects like classes, packages and operations that a system may encapsulate. Activity diagrams can be used in conjunction with the UC diagram techniques, towards illustrating the model behind the system being designed.

6.2. Embedded System

6.2.1.Static Model Analysis

In this section we will provide a static analysis of the Embedded System, by minutely describing its logical decomposition in its architectural elements along with their high-level interfaces and interactions. To facilitate understanding we provide a basic system context diagram which depicts the most important architectural components of the system in Figure 29.

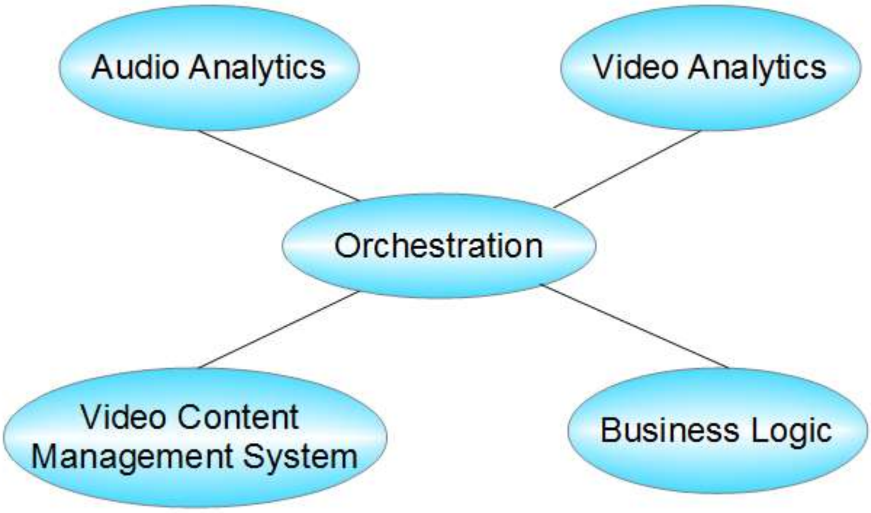


Figure 37 Basic context diagram of the Embedded System

The list of the architectural elements that compose the high-level logical structure of the Embedded System is presented below, while their analytic description can be found in Sections 2 and 3:

- The ***Sensor Manager (SM)*** component, which is the entry point of the raw sensor data (video/audio) to the Embedded System.
- The ***Video Analytics*** component, which receives video data from the SM and applies various computer vision algorithms in order to detect any anomalous events.
- The ***Audio Analytics*** component, which processes the audio data sent from the SM and decides whether the captured stream corresponds to an anomalous event.
- The ***Embedded System Manager (ESM)*** component, which combines the results of the analytics modules and decides whether an activation signal should be triggered on the Clip Generation component.
- The ***Clip Generator*** component, which generates a Clip Object when an activation signal is sent from the ESM.

Besides the aforementioned structural elements of the Embedded System there is also the ***Communication*** component that operates as the interface between the embedded and the cloud system. The Communication component consists of several sub-systems that ensure smooth data exchange between the embedded and the cloud system. These sub-systems are:

- The ***Encryption Manager***, which is responsible for adding extra layers of protection by encrypting the Clips before they are uploaded to the Cloud.
- The **Secure Communication Channel Manager,** which, as the name implies, secures the communication between the Embedded Systems and the Cloud. To achieve this it exploits OpenVPN framework.
- The **Clip Handler**, subsystem with the task of transferring the Clips, generated from the Embedded Systems, to the Cloud.

6.2.2.Dynamic Model Analysis

In this section, the main use cases of the Embedded System are described focusing on the dynamic behaviour of the system.

The basic use case of the Embedded System is illustrated in Figure 38, where the interaction between the components is shown. Multimedia streams are initially aggregated on the *Sensor Manager* which stores them on an internal buffer. Audio and video streams are propagated to the *Audio and Video Analytics* components respectively through a UDP socket. The analytics components send a JSON object to the *Embedded System Manager* through a TCP socket. The object contains the identity of the sensor, the anomaly score and a timestamp. Based on the scores received the *Embedded System Manager* decides whether a Clip Generation Event should be triggered. If the trigger is activated the *Clip Generator* component requests the available multimedia data from the *Sensor Manager*. Once the multimedia data are received, they are packaged with the metadata received from the analytics modules in a JSON object and then forwarded to the *Communication Manager*. The *Communication Manager* sends the Clip Object to the cloud system.

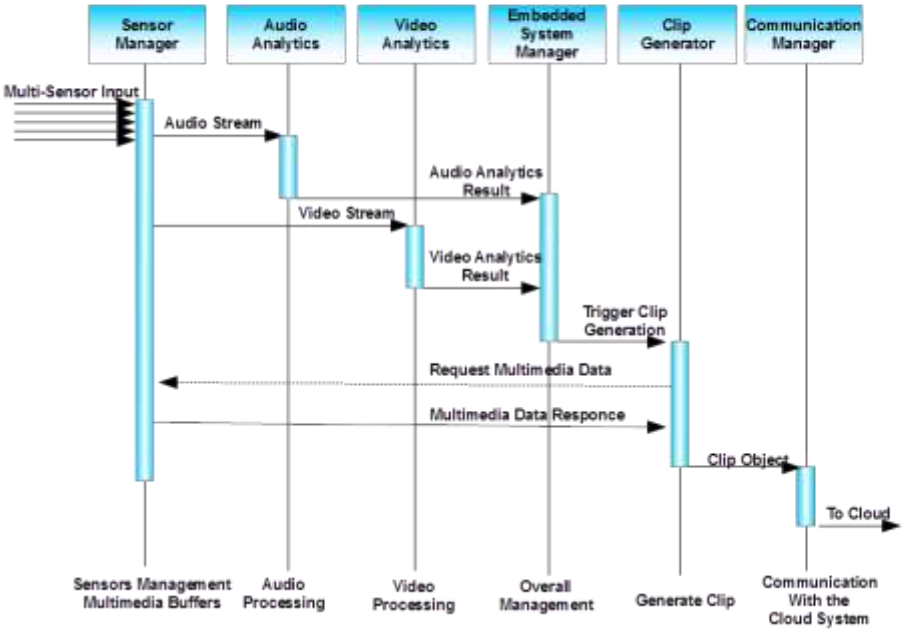


Figure 38 Basic use case of the Embedded System

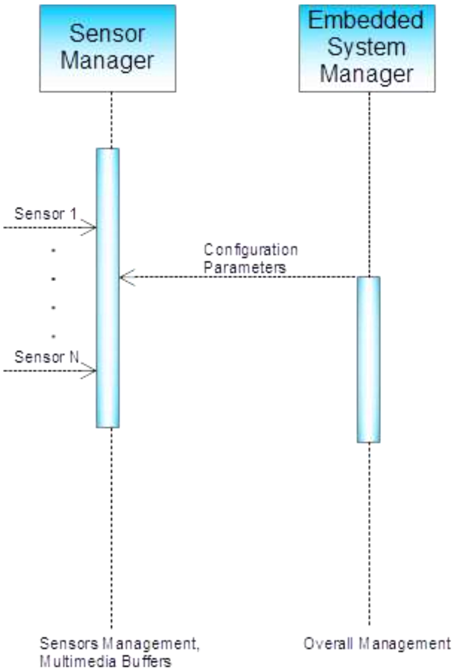


Figure 39 Use case of the embedded system where the sensors' parameters are configured

Another use case of the Embedded System is depicted in Figure 39 where the parameters of the sensors are configured (e.g. video resolution, zoom, audio volume etc.). Here the Embedded System Manager communicates with the Sensor Manager sending the necessary configuration parameters. This configuration occurs, when the system is initialised or when a request is received from the cloud system

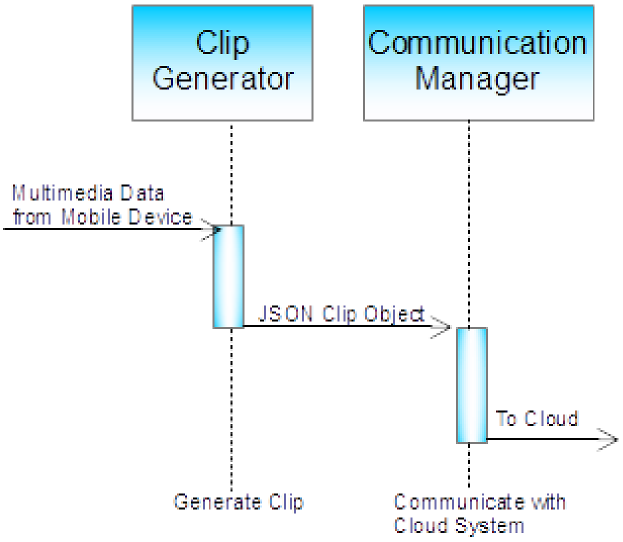


Figure 40 Use case where multimedia files are received from a mobile device and a clip is generated

The use case of Figure 40depicts the interaction of the involved components when multimedia data are sent from a mobile device to the *Clip Generator*. A JSON clip object is generated and is sent to the *Communication Manager*.

6.3. Cloud System

6.3.1.Static Model Analysis

The aim of this section is to provide a logical decomposition of the structural elements that comprise the Cloud System through a static model analysis. In the following paragraphs we will present in detail each individual component along with its interfaces and interactions. A system context diagram is shown in Figure 41.

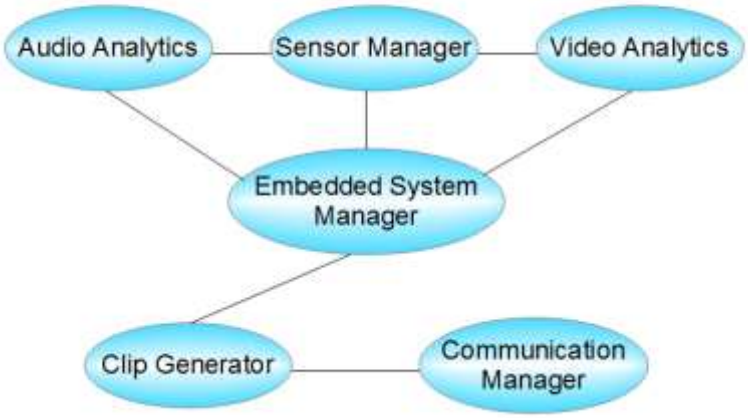


Figure 41 Basic context diagram of the cloud system

As illustrated in the figure above, the architectural components that compose the high-level logical structure of the cloud system are the following:

- The ***Orchestration*** component, which is the central component of the Cloud System and interacts with most of the other components.
- The ***Business Logic*** component, which will determine the analysis type for a new Clip Object.
- The ***Video Analytics*** component, which applies several computer vision algorithms on a specific Clip Object.
- The ***Audio Analytics*** component, which analyses the audio data from a clip object. The ***Video Content Management System (VCMS)*** component, which is responsible for storing audio and video metadata on the cloud as well as maintaining a URI linking to the actual video content of a Clip Object.

6.3.2.Dynamic Model Analysis

In this section we present the dynamic behaviour of the cloud system through a series of use cases. It should be noted that there is the possibility that the described use cases will be modified as the project evolves. Nonetheless, the basic use case of the cloud system is depicted in the diagram of Figure 42.

Here, a JSON Clip Object is received by the *Orchestration* component, sent by the *Communication Manager* of an Embedded System. Simultaneously, the multimedia data are stored on the cloud and a unique URI is generated. Moreover, a request to add the clip is sent to the *Video Content Management System*. The *Business Logic* is also informed in order to decide the type of analysis that will be applied on the clip. The analysis type is sent back to the *Orchestration* which communicates with the analytics components that are responsible for the further processing of the clip.

The analytics components retrieve from the *Cloud Storage* the respective multimedia data using the clip URI. After audio/video processing concludes, the results are forwarded to *Orchestration* which updates the clip object on the *Video Content Management System* and informs the *Business Logic* that the processing of the clip has concluded. *Business Logic* will then decide whether an authorized user of the system should be notified.

Another use case of the cloud system is illustrated in Figure 43. The *Business Logic* component is able to send to the *Orchestration* component a request to change the parameters of the sensors on an Embedded System. This request is propagated to the *Communication Manager* of the Embedded System which informs the *Embedded System Manager* following the use case of Figure 39. It should be added that instead of the *Business Logic* component, the user of the cloud system will also be able to request the reconfiguration of the sensors in a similar use case.

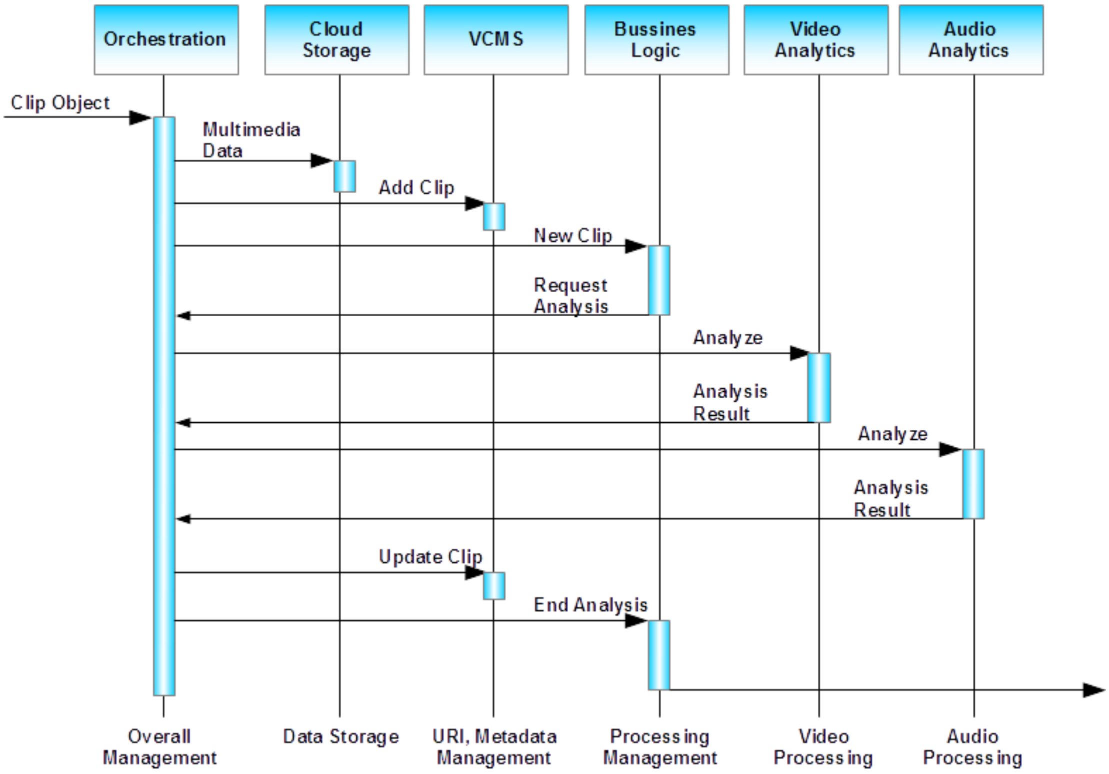


Figure 42 Basic use case of the cloud system

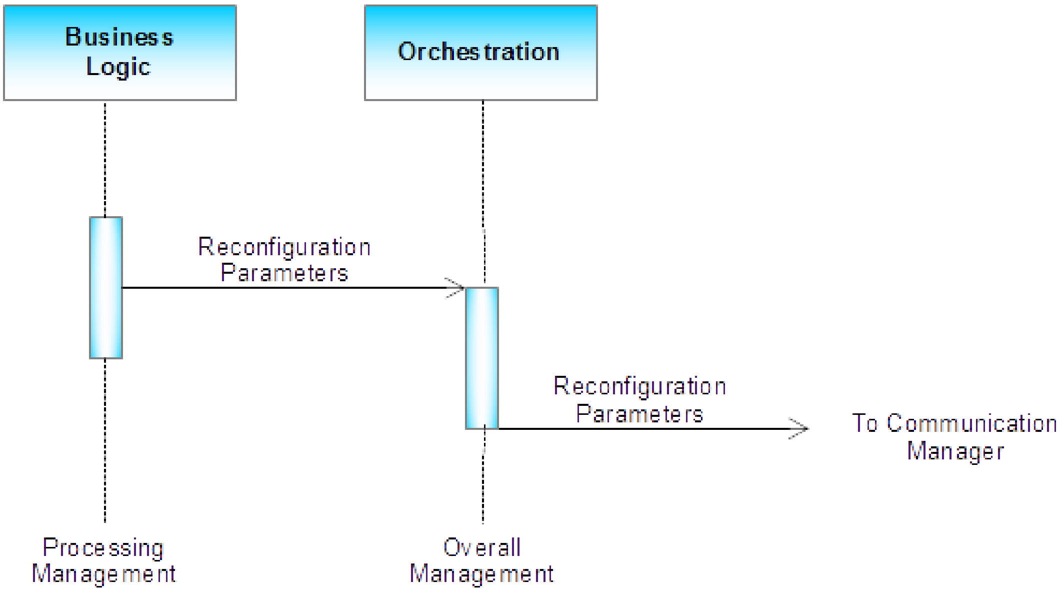


Figure 43 Use case where the sensors of an Embedded System are reconfigured

7. System Architectural Elements Specifications

7.1. Introduction

This Section highlights the scope of the P-REACT system and presents the broad functional areas to be provided by the P-REACT's Cloud and Embedded platforms, the high-level picture of each system boundaries and its adjacent external entities (in terms of the use of context diagrams).

The previous Section illustrated the static and dynamic behaviour of the P-REACT system along with the high-level context diagrams of the P-REACT architecture. The static analysis illustrated the main architectural elements, whereas the dynamic analysis has been performed through the UML Sequence Diagrams, taking into account the latest revisions of the P-REACT Use Cases for both the Embedded and the Cloud system.

In this Section, the core architectural elements that have been illustrated with system context diagrams in Section 3 are further analysed and a more detailed specification of them is provided. The specifications are given for each framework separately, whereas the common functional elements are described in separate section in this Section.

7.2. Embedded System

7.2.1.Sensors

The P-REACT system utilizes standard of-the-shelf cameras and microphones, so no specialized protocols and/or ports are required. In most cases the sensors will be directly connected with the Embedded System through a USB port, whereas IP cameras will also be supported, with the latter being connected through Ethernet or Wi-Fi with the Embedded System. The following tables present the detailed characteristics of the sensors that will be used. At this point it has to be noted that since the P-REACT system can support a number of different cameras and microphones, the minimum requirements are presented and not a specific model.

Sensor/Device: Video Sensing	
Sensor Description and Functionality	
1. Name	*The P-REACT system supports most of-the-shelf cameras*
2. Short Description	*The sensor will be utilized to detect abnormal events in both indoor and outdoor environments of small shops, gas station or public transportation infrastructure*
3. Measurement	*The sensor must be capable to acquire colour images and should have a night mode. The Sensor will be able to acquire data that will be fed to the* P-REACT*'s Video analytics algorithms for further analysis.* *The sensor is attached to a PC using the USB-interface. IP cameras are also supported.*
4. Functionality	*The sensor will be part of the* P-REACT *system in order to detect abnormal events (e.g. theft, fighting, anti-social behaviour) on both outdoor and indoor environments of small shops, gas stations and public transportation infrastructure. Moreover, in case an abnormal event is detected, data will be sent to the cloud system in order to identify the suspect(s).*
Sensor Characteristics (Physical)	
5. Dimensions	*Varies*
6. Weight	*varies*
7. Material	*N/A*
8. Mounting	*It should be made manually*
Sensor Characteristics (Operational)	

Sensor/Device: Video Sensing	
Sensor Description and Functionality	
9. Measurement Range	-
10. Measurement Resolution	*The sensor should provide minimum VGA resolution images (640x480 pixels) at 30 frames per second*
11. Accuracy	*N/A*
12. Zero Error	-
13. Humidity	*Operating temperature range applies (e.g. 5 to 35 degrees Celsius).*
14. Pressure	-
15. Lifetime	- *(N/A)*
Sensor Requirements (Hardware)	
16. Power Requirements	*Varies. Usually powered over the USB connection*
17. Data Connections	*USB 2.0*
18. Data Format	*The RGB video stream should be 24-bit VGA resolution (640 × 480 pixels)*
19. Data Epoch Rate	*The data rate is at minimum 30 Hz but it highly depends on the actual application use for real-time performance.*
20. Data Availability	*The data stream is continuous and can be acquired also as required (or on demand). The API and the SDK of the camera is flexible enough to acquire the images as soon as they are required on the aforementioned maximum data epoch rate.*
21. Transmission Frequency	*30 Hz*
22. Transmission Power	*N/A*
Sensor Requirements (Software)	
23. Specific Software Required?	*No*
24. Software Details	*N/A*
Sensor Requirements (Ethical / Privacy Issues)	
25. Specific Software Required?	*The sensor acquires data that could be used to identify a person. However, within* P-REACT *special countermeasures will be taken into account in order to address all ethical, legal and privacy issues.*
26. Software Details	N/A
Miscellaneous	
27. References	-
28. Comments	*This will be the main proposed system for the* P-REACT *solution. It offers good performance at a low cost, and in many cases will be able to utilize already existing equipment.*

Sensor/Device: Depth Imaging

Sensor/Device: Depth Imaging	
Sensor Description and Functionality	
1. Name	*Kinect Sensor v2 (Microsoft)*
2. Short Description	*The sensor will be utilized to detect abnormal events in indoor environments of small shops or gas stations. It can also be utilized on the interior of public transportation (buses, trains etc.)*
3. Measurement	*The sensor is capable to acquire colour, Infrared and depth images. Within* P-REACT, *only the depth images will be used in each case. The depth sensor will be able to acquire depth data that will be fed to the* P-REACT*'s Video analytics algorithms for further analysis (Event detection, person identification).* *The sensor is attached to a PC using the USB-interface (USB 3.0)*
4. Functionality	*The sensor will be part of the* P-REACT *system in order to detect abnormal events (e.g. theft, fighting, shop lifting) on indoor environments of small shops, gas stations and public transportation. Moreover, in case an abnormal event is detected, data will be sent to the cloud system in order to identify the suspect(s).*
Sensor Characteristics (Physical)	
5. Dimensions	*The Kinect Sensor head size is 292.1 x 304.8 x 184.15 (in mm)* *<L x W x H in mm>*
6. Weight	*~1. kg*
7. Material	*N/A*
8. Mounting	*It should be made manually*
Sensor Characteristics (Operational)	
9. Measurement Range	*The operating sensor range for depth images is from 0.4m to 8m (extended range with lower accuracy).*
10. Measurement Resolution	*The measurement resolution is given for both colour and depth sensing lenses:* *Colour VGA motion camera 1920 x 1080 pixel resolution @30 FPS* *Depth Camera 512 x 424 pixel resolution @30 FPS* *Infrared Camera 512 x 424 pixel resolution @30 FPS* *The corresponding field of view is the following:* *Horizontal field of view- 70 degrees* *Vertical field of view- 60 degrees*
11. Accuracy	*At the maximum range the random error of depth measurements increases quadratically. It can reach even 4cm at the maximum range (8 meters [34])*
12. Zero Error	*The Kinect Sensor is pre-calibrated with its lens.*
13. Humidity	*Operating temperature range applies (e.g. 5 to 35 degrees Celsius).*
14. Pressure	-

Sensor/Device: Depth Imaging	
15. Lifetime	*- (N/A)*
Sensor Requirements (Hardware)	
16. Power Requirements	*The power is supplied from the mains by way of an AC adapter.12 Watts power demand through a standard USB port interface.*
17. Data Connections	*USB 3.0*
18. Data Format	*The output of the sensor is raw data for colour and depth images (RGB and depth images with pixel values representing depth in mms).* *The RGB video stream uses 32-bit FullHD resolution (1920 × 1080 pixels) with a Bayer colour filter, while the monochrome depth sensing video stream is in resolution of 512 × 242 pixels) with 16-bit depth.*
19. Data Epoch Rate	*The data rate is about 30 Hz but it highly depends on the actual application use for real-time performance.*
20. Data Availability	*The data stream is continuous and can be acquired also as required (or on demand). The API and the SDK of the camera is flexible enough to acquire the images as soon as they are required on the aforementioned maximum data epoch rate.*
21. Transmission Frequency	*30 Hz*
22. Transmission Power	*N/A*
Sensor Requirements (Software)	
23. Specific Software Required?	*The sensor is accompanied with a corresponding SDK. The latest version is Windows Kinect SDK 2.*
24. Software Details	*Kinect SDK is available for free and use for research purposes.* • *Kinect SDK is available on [34]*
Sensor Requirements (Ethical / Privacy Issues)	
25. Specific Software Required?	*The sensor acquires data that could be used to identify a person. However, within* P-REACT *only the depth images will be used for analysis and special countermeasures will be taken into account in order to address all ethical, legal and privacy issues.*
26. Software Details	The same API described in 24.
Miscellaneous	
27. References	*More information for the Kinect Sensor can be found in the official site from Microsoft:* *http://www.microsoft.com/en-us/kinectforwindows/*
28. Comments	*The Kinect Sensor will be part of the* P-REACT *system, as a more advanced proposal for volume crime detection in indoor environments.*

Sensor/Device: Audio Sensing	
Sensor Description and Functionality	
1. Name	*The P-REACT system supports most of-the-shelf microphones*
2. Short Description	*The sensor will be utilized to detect abnormal events in both indoor and outdoor environments of small shops, gas station or public transportation*

Sensor/Device: Audio Sensing	
Sensor Description and Functionality	
	infrastructure
3. Measurement	*The sensor must be capable to acquire audio at 16-bit audio at a sampling rate of 16 kHz* *The sensor is attached to a PC using the USB-interface.*
4. Functionality	*The sensor will be part of the P-REACT system in order to detect abnormal events (e.g. screaming, breaking glass) on both outdoor and indoor environments of small shops, gas stations and public transportation infrastructure. Moreover, in case an abnormal event is detected, data will be sent to the cloud system in order to identify the suspect(s).*
Sensor Characteristics (Physical)	
5. Dimensions	*Varies*
6. Weight	*varies*
7. Material	*N/A*
8. Mounting	*It should be made manually*
Sensor Characteristics (Operational)	
9. Measurement Range	*-*
10. Measurement Resolution	-
11. Accuracy	*N/A*
12. Zero Error	-
13. Humidity	*Operating temperature range applies (e.g. 5 to 35 degrees Celsius).*
14. Pressure	-
15. Lifetime	*- (N/A)*
Sensor Requirements (Hardware)	
16. Power Requirements	*Varies. Usually powered over the USB connection*
17. Data Connections	*USB 2.0*
18. Data Format	*WAVE format*
19. Data Epoch Rate	
20. Data Availability	-
21. Transmission Frequency	-
22. Transmission Power	*N/A*
Sensor Requirements (Software)	
23. Specific Software Required?	*No*
24. Software Details	*N/A*

Sensor/Device: Audio Sensing	
Sensor Description and Functionality	
Sensor Requirements (Ethical / Privacy Issues)	
25. Specific Software Required?	*The sensor acquires data that could reveal a person's identity. However, within P-REACT special countermeasures will be taken into account in order to address all ethical, legal and privacy issues.*
26. Software Details	N/A
Miscellaneous	
27. References	-
28. Comments	*This will be part of the main proposed system for the P-REACT solution. It offers good performance at a low cost, and in many cases will be able to utilize already existing equipment.*
29. Transmission Power	*N/A*

7.2.2. Architectural Elements

7.2.2.1. Sensor Management Interfaces

Name:	*Sensor Management*				
Type:	*Interface*				
Container	*The sensor manager is the connectivity layer between the sensors and the* Embedded System				
Functionality:	*Sensor Management is responsible for capturing data from the sensors and feed it to the rest of the Embedded System's modules.*				
Connections:	*Sensor Devices* *Video- Audio analytics* *Clip Generator*				
INPUT PARAMETERS					
Attribute / Parameter	***Short Description***	***Data Type***	***Cardinality***	***Value Range***	***Source/ Involved Entities***
Video Raw data	*Streaming Data from the Camera*	*Streaming Images (byte arrays)*	*1..**	*Frames Per Second* *FPS shall be at least 15*	*Camera*

Depth Raw Sensor Data	*Streaming Data from the Depth Sensor*	*Streaming Images (byte arrays)*	*1..**	*Frames Per Second* *FPS shall be at least15*	*Depth Sensor (Kinect)*
Audio Raw Data	*Streaming Data from the microphone*	*Streaming Audio signal (byte arrays)*	*1…**	*Sequential Audio Frames of 100ms length*	*Microphone*
OUTPUT PARAMETERS					
Attribute / Parameter	***Short Description***	***Data Type***	***Cardinality***	***Value Range***	***Destination***
All of the aforementioned inputs	*See above for each input row*	*See above for each input row*	*See above for each input row*	*See above for each input row*	*Video/Audio analytics* *Clip generator*
Hardware requirements	*Hardware requirements vary depending on the sensors used. Detailed hardware requirements have been presented in Section 5.1*				
Software requirements and specification / development language	*The Sensors Management supports the integration of sensors using all the operating systems and hardware combinations where it runs. It has been developed in C/C++ programming language*				
Power requirements	*Typical power for each system.*				
Other resources required / environment	*The restrictions are highly correlated to the sensors utilized into each instantiation.*				
Communications	*Using JSON objects*				
Issues	*The Sensor Management shall be able to cope with all sensors described in Section 5.1.*				
Pseudo code	*Sensor Management* *Identify all sensors connected to the* Embedded System				

```
        Do Until interrupted
                For Each Sensor
                        Update data buffer
                        If requested pass sensor buffer to Video/Audio analytics
                End
                If requested pass all sensor buffers to clip generator
        Loop
End
```

7.2.2.2. Clip Generator

Name:	*Clip Generator*				
Type:	*Module*				
Container	*The clip generator receives data from the sensors managers, packs it to a clip object along with the necessary metadata (stemming from the* Embedded System *and/or the video/audio analytics modules) and sends it to the cloud.*				
Functionality:	*Clip Generator is responsible for creating a clip object, send it to the cloud and ensure that it has been delivered.*				
Connections:	*Sensor Devices* *Communication Module*				
INPUT PARAMETERS					
Attribute / Parameter	***Short Description***	***Data Type***	***Cardinality***	***Value Range***	***Source/ Involved Entities***
Sensor buffer data	*Buffered Data from the sensors*	-	*1..**	-	*Sensor Manager*
OUTPUT PARAMETERS					
Attribute / Parameter	***Short Description***	***Data Type***	***Cardinality***	***Value Range***	***Destination***
All of the aforementioned inputs	*See above for each input row*	*See above for each input row*	*See above for each input*	*See above for each input row*	*Video/Audio analytics* *Clip generator*

<table>
<tr><td></td><td></td><td></td><td><i>row</i></td><td></td><td></td></tr>
<tr><td><b>Hardware requirements</b></td><td colspan="5"><i>Hardware requirements vary depending on the sensors used. Detailed hardware requirements have been presented in Section 5.1</i></td></tr>
<tr><td><b>Software requirements and specification / development language</b></td><td colspan="5"><i>It has been developed in C/C++ programming language</i></td></tr>
<tr><td><b>Power requirements</b></td><td colspan="5"><i>Typical power for each system.</i></td></tr>
<tr><td><b>Other resources required / environment</b></td><td colspan="5"><i>N/A</i></td></tr>
<tr><td><b>Communications</b></td><td colspan="5"><i>Using JSON objects</i></td></tr>
<tr><td><b>Issues</b></td><td colspan="5">-</td></tr>
<tr><td><b>Pseudo code</b></td><td colspan="5"><i>Clip Generator
    If an event has been detected
    Request sensor data from Sensor Manager
    Create Clip Object
    Send Clip to Cloud
    Wait for receive confirmation
End</i></td></tr>
</table>

7.2.2.3. Embedded system Manager

Name:	*Embedded System Manager*
Type:	*Module*
Container	*The Embedded System manager module is responsible for monitoring the Embedded System and also to receive the output of the video and audio analytics and notify the clip generator if an event has been detected.*
Functionality	*The Embedded System manager monitors the Embedded System and performs*

:	*control, maintenance and/or update actions that are received from the cloud.* *It is also responsible for receiving input from the video and audio analytics modules and decides whether an event has occurred. In case an event has occurred, it is responsible for instructing the clip generator to create a clip.*				
Connections:	*Video/Audio analytics* *Clip generator* *Communication Module*				
INPUT PARAMETERS					
Attribute / Parameter	***Short Description***	***Data Type***	***Cardin ality***	***Value Range***	***Source/ Involved Entities***
Video/Audio analytics output	*The output of the video/audio analytics. This can be a binary or a decimal value indicating whether an event has occurred.*	*Binary/Decimal*	*1..**	*0-1*	*Video/Audio analytics*
Maintenance/Control/Update	*Signals coming from the cloud (through the communications module) that*	*JSON objects*	*1…**	*N/A*	*Cloud (through Communications module)*
OUTPUT PARAMETERS					
Attribute / Parameter	***Short Description***	***Data Type***	***Cardin ality***	***Value Range***	***Destination***
All of the aforementioned inputs	*See above for each input row*	*See above for each input row*	*See above for each input row*	*See above for each input row*	*Video/Audio analytics* *Clip generator* *Embedded System's OS*
Hardware requirements	*Hardware requirements vary depending on the sensors used. Detailed hardware requirements have been presented in Section 5.1*				

Software requirements and specification / development language	*It has been developed in C/C++ programming language*
Power requirements	*Typical power for each system.*
Other resources required / environment	*N/A*
Communications	*Using JSON objects*
Issues	*-*

7.2.2.4. Audio Analytics

Name:	*Audio Analytics Module*				
Type:	*Module*				
Container	*The audio analytics module on the Embedded System performs lightweight audio analysis in order to detect abnormal events*				
Functionality:	*The audio analytics module of the Embedded System receives data from the Sensor Manager and performs lightweight analysis on them. Its purpose is to determine whether an abnormal event has occurred, and provide its output to the Embedded System manager. The output of the module is either a binary value (with 0 meaning no event and 1 indicating an event), or a decimal value, indicating a confidence out the existence of an event*				
Connections:	*Embedded System Manager* *Sensor Manager*				
INPUT PARAMETERS					
Attribute / Parameter	***Short Description***	***Data Type***	***Cardinality***	***Value Range***	***Source/ Involved Entities***
Audio Raw Data	*Streaming Data from the*	*Streaming Audio signal (byte arrays)*	*1…**	*Sequential Audio Frames of*	*Microphone*

	microphone			*100ms length*	
OUTPUT PARAMETERS					
Attribute / Parameter	***Short Description***	***Data Type***	***Cardinality***	***Value Range***	***Destination***
All of the aforementioned inputs	*See above for each input row*	*See above for each input row*	*See above for each input row*	*See above for each input row*	*Embedded System Manager*
Hardware requirements	*Hardware requirements vary depending on the sensors used. Detailed hardware requirements have been presented in Section 5.1*				
Software requirements and specification / development language	*It has been developed in C/C++ programming language*				
Power requirements	*Typical power for each system.*				
Other resources required / environment	*N/A*				
Communications	*Using JSON objects*				
Issues	-				

7.2.2.5. Video Analytics

Name:	*Video Analytics Module*
Type:	*Module*
Container	*The video analytics module on the Embedded System performs lightweight video analysis in order to detect abnormal events*

Functionality:	*The video analytics module of the Embedded System receives data from the Sensor Manager and performs lightweight analysis on them. Its purpose is to determine whether an abnormal event has occurred, and provide its output to the Embedded System manager. The output of the module is either a binary value (with 0 meaning no event and 1 indicating an event), or a decimal value, indicating a confidence out the existence of an event.* *Depending on the available sensors, the Video analytics module will perform analysis on either colour data or depth data.*				
Connections:	*Embedded System Manager* *Sensor Manager*				
INPUT PARAMETERS					
Attribute / Parameter	***Short Description***	***Data Type***	***Cardinality***	***Value Range***	***Source/ Involved Entities***
Video Raw data	*Streaming Data from the Camera*	*Streaming Images (byte arrays)*	*1..**	*Frames Per Second* *FPS shall be at least 15*	*Camera*
Depth Raw Sensor Data	*Streaming Data from the Depth Sensor*	*Streaming Images (byte arrays)*	*1..**	*Frames Per Second* *FPS shall be at least 15*	*Depth Sensor (Kinect)*
OUTPUT PARAMETERS					
Attribute / Parameter	***Short Description***	***Data Type***	***Cardinality***	***Value Range***	***Destination***
All of the aforementioned inputs	*See above for each input row*	*See above for each input row*	*See above for each input row*	*See above for each input row*	*Embedded System Manager*
Hardware requirements	*Hardware requirements vary depending on the sensors used. Detailed hardware requirements have been presented in Section 5.1*				
Software requirement	*It has been developed in C/C++ programming language*				

s and specification / development language	
Power requirements	*Typical power for each system.*
Other resources required / environment	*N/A*
Communications	*Using JSON objects*
Issues	-

7.2.2.6. Communications Module

<table>
<tr><td><b>Name:</b></td><td colspan="5"><i>Communications Module</i></td></tr>
<tr><td><b>Type:</b></td><td colspan="5"><i>Module</i></td></tr>
<tr><td><b>Container</b></td><td colspan="5"><i>The communications module relays messages from the cloud to the Embedded System and sends clip objects to the cloud.</i></td></tr>
<tr><td><b>Functionality:</b></td><td colspan="5"><i>The communications modules is responsible for receiving messages from the cloud system and relay them to the Embedded System Manager module as well as forwarding Clip objects, that the Clip generator has created, to the cloud. For the latter, it also receives confirmation of clip received from the cloud and passes it to the clip generator.</i></td></tr>
<tr><td><b>Connections:</b></td><td colspan="5"><i>Embedded System Manager</i>
<i>Clip Generator</i>
<i>Cloud</i></td></tr>
<tr><td colspan="6"><b><i>INPUT PARAMETERS</i></b></td></tr>
<tr><td><b><i>Attribute / Parameter</i></b></td><td><b><i>Short Description</i></b></td><td><b><i>Data Type</i></b></td><td><b><i>Cardinality</i></b></td><td><b><i>Value Range</i></b></td><td><b><i>Source/ Involved Entities</i></b></td></tr>
<tr><td><i>Clip Object</i></td><td><i>Clip Object from the clip generator</i></td><td><i>JSON</i></td><td><i>1…*</i></td><td><i>N/A</i></td><td><i>Clip Generator</i></td></tr>
</table>

Signals from the cloud	*Embedded System Control/Maintenance/Update signals coming from the cloud.*	*JSON*	1…*	*N/A*	*Cloud*
OUTPUT PARAMETERS					
Attribute / Parameter	***Short Description***	***Data Type***	***Cardinality***	***Value Range***	***Destination***
All of the aforementioned inputs	*See above for each input row*	*See above for each input row*	*See above for each input row*	*See above for each input row*	*Cloud* *Clip Generator*
Hardware requirements	*Hardware requirements vary depending on the sensors used. Detailed hardware requirements have been presented in Section 5.1*				
Software requirements and specification / development language	*It has been developed in C/C++ programming language*				
Power requirements	*Typical power for each system.*				
Other resources required / environment	*N/A*				
Communications	*Using JSON objects*				
Issues	-				

7.3. Cloud Architectural Elements

7.3.1.1. Communications Module

7.3.1.2. Business Logic

Name:	*Business Logic*
Type:	*Component*
Container	The Business Logic Component (aka 'Brain') is the component which manages the cameras, VCMS and interfaces with the human operators via the GUI. Three parts are: • Alert Raising • Interaction & Control • Real Time Monitoring
Functionality:	**Alert Raising** This will decide which clips are important enough to raise an alert. This decision is taken with input from cloud-side video analytics and statistical analysis. **Interaction and Control** This is feedback to the cameras. Feedback is via the "Encryption Manager/secure communication" component. Feedback is in xml/soap. Feedback can: • switch on/off cameras • change sensitivity of particular cameras • Tell camera to switch algorithms • Update camera software • Upload new algorithm binaries to camera **Real Time Monitoring** This is the point where clips are accessed, either live clips or stored clips, and displayed to the GUI. GUI/User can validate clips.
Connections:	*Orchestration*

INPUT PARAMETERS					
Attribute / Parameter	***Short Description***	***Data Type***	***Cardinality***	***Value Range***	***Source/ Involved Entities***
New Event	*A notification that an event has been*	*JSON*	*1…**	*N/A*	*Orchestration*

	detected				
Control - Monitor	*Signals that are sent to the* Embedded System *s in order to control them (configure, change sensor settings etc.) or enable real time monitoring*	*JSON*	*1…**	*N/A*	*Orchestration*
OUTPUT PARAMETERS					
Attribute / Parameter	***Short Description***	***Data Type***	***Cardinality***	***Value Range***	***Destination***
All of the aforementioned inputs	*See above for each input row*	*See above for each input row*	*See above for each input row*	*See above for each input row*	*Cloud* *Clip Generator*
Hardware requirements	*Hardware requirements vary depending on the sensors used. Detailed hardware requirements have been presented in Section 5.1*				
Software requirements and specification / development language	*It has been developed in C/C++ programming language*				
Power requirements	*Typical power for each system.*				
Other resources required / environment	*N/A*				
Communica	*Using JSON objects*				

tions	
Issues	-

7.3.1.3. Video Analytics

Name:	*Video Analytics Module*				
Type:	*Module*				
Container	*The video analytics module on the cloud performs advanced video analysis in order to verify the event detected on the Embedded System and identify suspect(s)*				
Functionality:	*The video analytics module of the cloud receives a clip object and performs advanced video analysis on them. Its purpose is to verify the presence of an abnormal event, and detect/identify the suspects. The latter will be achieved using biometric analysis (gait, face and activity identification) while complying with ethical and privacy guidelines* *Depending on the available sensors, the Video analytics module will perform analysis on colour data and/or depth data.*				
Connections:	*Orchestration*				
INPUT PARAMETERS					
Attribute / Parameter	***Short Description***	***Data Type***	***Cardinality***	***Value Range***	***Source/ Involved Entities***
Video Raw data	*Streaming Data from the Camera*	*Streaming Images (byte arrays)*	*1..**	*Frames Per Second* *FPS shall be at least 15*	*Camera*
Depth Raw Sensor Data	*Streaming Data from the Depth Sensor*	*Streaming Images (byte arrays)*	*1..**	*Frames Per Second* *FPS shall be at least15*	*Depth Sensor (Kinect)*
OUTPUT PARAMETERS					
Attribute /	***Short***	***Data Type***	***Cardin***	***Value***	***Destination***

Parameter	**Description**		**ality**	**Range**	
All of the aforementioned inputs	*See above for each input row*	*See above for each input row*	*See above for each input row*	*See above for each input row*	*Embedded System Manager*
Hardware requirements	*Hardware requirements vary depending on the sensors used. Detailed hardware requirements have been presented in Section 5.1*				
Software requirements and specification / development language	*It has been developed in C/C++ programming language*				
Power requirements	*Typical power for each system.*				
Other resources required / environment	*N/A*				
Communications	*Using JSON objects*				
Issues	-				

7.3.1.4. Audio Analytics

Name:	*Audio Analytics Module*
Type:	*Module*
Container	*The audio analytics module on the cloud performs audio analysis in order to detect abnormal events*
Functionality:	*The audio analytics module of the cloud receives a clip object and performs analysis on them. Its purpose is to determine whether an abnormal event has occurred, and provide its output to the Brain. The output of the module is either a binary value (with 0 meaning no event and 1 indicating an event), or a decimal value, indicating a confidence out the existence of an*

	event				
Connections:	*Orchestration*				
INPUT PARAMETERS					
Attribute / Parameter	***Short Description***	***Data Type***	***Cardinality***	***Value Range***	***Source/ Involved Entities***
Audio Raw Data	*Streaming Data from the microphone*	*Streaming Audio signal (byte arrays)*	*1…**	*Sequential Audio Frames of 100ms length*	*Microphone*
OUTPUT PARAMETERS					
Attribute / Parameter	***Short Description***	***Data Type***	***Cardinality***	***Value Range***	***Destination***
All of the aforementioned inputs	*See above for each input row*	*See above for each input row*	*See above for each input row*	*See above for each input row*	*Embedded System Manager*
Hardware requirements	*Hardware requirements vary depending on the sensors used. Detailed hardware requirements have been presented in Section 5.1*				
Software requirements and specification / development language	*It has been developed in C/C++ programming language*				
Power requirements	*Typical power for each system.*				
Other resources required / environment	*N/A*				

Communications	*Using JSON objects*
Issues	-

7.3.1.5. VCMS

Name:	*Video Content Manager System*				
Type:	*Module*				
Container	*The VCMS is responsible for managing the clip objects that are stored in the* P-REACT *system.*				
Functionality:	*The VCMS receives from the orchestration clip objects and stores them accordingly. It also feeds clip objects to the orchestration when requested in order to be further analysed. Besides this, it provides a GUI in order for the operator to view or analyse clips on demand.*				
Connections:	*Orchestration*				
INPUT PARAMETERS					
Attribute / Parameter	***Short Description***	***Data Type***	***Cardinality***	***Value Range***	***Source/ Involved Entities***
Clip Objects	*Clip objects that are stored in the database*	*JSON*	*1…**	*…*	*Orchestration*
OUTPUT PARAMETERS					
Attribute / Parameter	***Short Description***	***Data Type***	***Cardinality***	***Value Range***	***Destination***
All of the aforementioned inputs	*See above for each input row*	*See above for each input row*	*See above for each input row*	*See above for each input row*	*Embedded System Manager*
Hardware requirements	*Hardware requirements vary depending on the sensors used. Detailed hardware requirements have been presented in Section 5.1*				

Software requirements and specification / development language	*It has been developed in C/C++ programming language*
Power requirements	*Typical power for each system.*
Other resources required / environment	*N/A*
Communications	*Using JSON objects*
Issues	-

7.3.1.6. Orchestration

Name:	*Orchestration*
Type:	*Module*
Container	*The Orchestration Module is the main module of the system.*
Functionality:	*The orchestration receives clip objects from the communication manager and feeds them to the Brain module. It forwards the Brain's instructions on the analytics algorithms that need to be used on a specific clip to the analytics module, and instructs the VCMS to store the incoming clip object. It also relays control and maintenance messages for the* Embedded System *s to the communications module.*
Connections:	*Brain* *VCMS* *Video/Audio Analytics* *GUI* *Communications Module*
INPUT PARAMETERS	

Attribute / Parameter	***Short Description***	***Data Type***	***Cardinality***	***Value Range***	***Source/ Involved Entities***
New Clip Object	*Incoming clip object*	*JSON*	*1…**	*…*	*Communication Module*
Analyse Clip object	*Instructions to analyse a clip object*	*…*	*1…**	*…*	*Brain*
Video/Audio Analytics response	*The result of the analysis on a clip object*	*JSON*	*1…**	*…*	*Analytics*
OUTPUT PARAMETERS					
Attribute / Parameter	***Short Description***	***Data Type***	***Cardinality***	***Value Range***	***Destination***
All of the aforementioned inputs	*See above for each input row*	*See above for each input row*	*See above for each input row*	*See above for each input row*	*Embedded System Manager*
Hardware requirements	*Hardware requirements vary depending on the sensors used. Detailed hardware requirements have been presented in Section 5.1*				
Software requirements and specification / development language	*It has been developed in C/C++ programming language*				
Power requirements	*Typical power for each system.*				
Other resources required / environment	*N/A*				
Communications	*Using JSON objects*				

Issues	-

8. Conclusions

In the context of WP2 and collaboration with other P-REACT WPs (WP1, WP3-WP4), the most important components of the P-REACT system have been presented and analysed, whereas the interaction among key components of each framework has been analysed through the use of UML sequence diagrams. Moreover, the static behaviour among key architectural elements has been presented using system context diagrams and in addition the dynamic behaviour has been analysed by elaborating the use cases defined in WP1 with additional information targeting mainly to outline the core functionalities of the P-REACT system and critical requirements that shall be taken into account during the implementation and integration phases of the project (WP3-WP4).

The architectural views and perspectives presented in this deliverable will further drive the design and implementations during the project lifetime as:

- The system context diagrams have illustrated the core components of the P-REACT system, the key stakeholders as well as the way they interact with the system. This part of the report outlined the static analysis for both the Embedded System and the Cloud.
- The detailed analysis of the use cases through UML diagrams have driven the refinement of the functional requirements of each architectural element and simultaneously have provided an overall design perimeter and the principle interactions among key components of P-REACT system.
- The detailed description of the architectural elements have provided a comprehensive view of the P-REACT components focusing mainly on its major architectural elements and providing high-level critical class diagrams, which allow to reason about and describe the dynamic behaviour of the system. The analysis presented for each architectural element, which encompasses data inputs and outputs, interrelation among system entities and correlation with respective use cases and key system requirements will enable during the implementation phase module developers and integrators to communicate about architectural issues in a more efficient and effective way.
- Moreover, interoperability issues have been addressed in this deliverable, as both Embedded System and the Cloud shall cope with several interoperability requirements stemming from the different needs of its stakeholders.

Although no major modifications are expected to the overall P-REACT system architecture and its major components, this deliverable can be considered as a living document that will address minor refinements that might be necessary in case of new unforeseen limitations that will come up during the implementation phase. It is worth mentioning that in the definition of the architectural elements process, all major module developer responsible partners were involved. The involvement of the developers to the architecture refinement process was significant because it resulted in a more coherent architecture definition (and its architectural elements), which also encapsulated the view of developers.

Summarizing, this deliverable has provided a sound groundwork for the technical developments of the P-REACT system that will take place in WPs 3 and 4, whereas the actual architectural elements of each framework will be implemented and validated.

ANNEX I. Orchestration API Description

Explanation of summary table columns

Method	The HTTP method to be used when requesting the resource
Resource URI	The address of the resource, any portions surrounded by {} are parameters.
Request URI	Parameters to be sent as part of the request, these will be passed to the service in the URI of the request.
Request Body	Parameters to be sent as part of the request, these will be included in the body of the request. The layouts of the listed data types are found here API Types
Response HTTP Status Codes	A summary listing of the possible HTTP response codes specific to that resource, a further explanation of the circumstances that would produce these codes is given in the individual resource documentation.
Response Body	The data that will be returned in the body of the response, the layouts of the listed data types are found here API Types
Interface	A reference to the interface that will use this particular resource, an explanation of the Interfaces can be seen in Figure 30

Summary Table

Method	Resource URI	Response HTTP Status Codes	Interface	Description
POST	/clips	201 (Created) 406 (Not Acceptable)	C	Add new clip
GET	/clips/{clip_id}	200 (Ok) 404 (Not Found)	G	Get clip by id
POST	/clips/{clip_id}/analyse	202 (Accepted) 404 (Not Found)	B	Instruct orchestration to analyse clip
POST	/clips/{clip_id}/analysisFinished	200 (Ok) 404 (Not Found)	H	Inform orchestration analysis has completed
POST	/clips/{clip_id}/analysisFailed	204 (No Content) 404 (Not Found)	H	Inform orchestration analysis failed
POST	/embeddedsystem/{system_id}/ report/failure	200 (Ok) 404 (Not Found)	C	Report Embedded System failure

POST	/embeddedsystem/{system_id} /report/tampering	200 (Ok) 404 (Not Found)	C	Report suspected Embedded System tampering
POST	/embeddedsystem/{system_id}/ register	200 (Ok) 404 (Not Found)	C	Register Embedded System
POST	/embeddedsystem/{system_id}/ unregister	200 (Ok) 404 (Not Found)	C	Unregister Embedded System

- **POST /clips**

Description: Used to add a new clip to the system, an Id will be assigned to the clip by the orchestration module (by requesting one from the VCMS module, Interface F).

Supported HTTP Methods: POST

Parameters

Name	Data Type	Format	Required	Example Value	Description
clip	clip	JSON	Yes	API Types	The clip object to be created in the system

HTTP Responses

Response	Description
201 (Created)	The clip was successfully added to the system.
406 (Not Acceptable)	The format of the supplied clip object was not valid.

Example Request

```
POST /clips HTTP/1.1
Host: api.p-react.eu
Content-Type: application/json

{
    "name": "clip_34.mjpeg",
    "algorithmName": "Fast",
    "dataType": "Video",
    "uri": "https://vcms.p-react.eu/videos/",
    "signature": "d41d8cd98f00b204e9800998ecf8427e",
    "validated": true,
    "startTime": "1410966187",
    "endTime": "1410966310"
}
```

- **GET /clips/{clip_id}**

Description: Gets a clip from the system

Supported HTTP Methods: GET

Parameters

Name	Data Type	Format	Required	Example Value	Description
clip_id	UUID /	URI	Yes	8c35d93e-d4d3-468e-b049-	An Id that uniquely

	GUID	Parameter		19b56b63ad1e	identifies a clip

HTTP Responses

Response	Description
200 (Ok)	The clip was retrieved successfully and is contained in the body of the response.
404 (Not Found)	The requested clip was not found on the system.

Example Request

```
GET /clips/8c35d93e-d4d3-468e-b049-19b56b63ad1e HTTP/1.1
Host: api.p-react.eu
```

Example Response

```
{
    "name": "clip_34.mjpeg",
    "id": "8c35d93e-d4d3-468e-b049-19b56b63ad1e",
    "algorithmName": "Fast",
    "dataType": "Video",
    "uri": "https://vcms.p-react.eu/videos/",
    "signature": "d",
    "validated": true,
    "startTime": "1410966187",
    "endTime": "1410966310"
}
```

- **POST /clips/{clip_id}/analyse**

Description: Requests the system to analyse a given clip.

Supported HTTP Methods: POST

Parameters

Name	Data Type	Format	Required	Example Value	Description
clip_id	UUID / GUID	UriURI Parameter	Yes	8c35d93e-d4d3-468e-b049-19b56b63ad1e	An Id that uniquely identifies a clip

HTTP Responses

Response	Description
202 (Accepted)	The request to analyse the clip was accept and has been queued in the system.
404 (Not Found)	The supplied clip id was not found on the system.

Example Request

```
GET /clips/8c35d93e-d4d3-468e-b049-19b56b63ad1e/analyze HTTP/1.1
Host: api.p-react.eu
```

- **POST /embeddedsystem/{system_id}/report/failure**

Description: Report a failure of an Embedded System.

Supported HTTP Methods: POST

Parameters

Name	Data Type	Format	Required	Example Value	Description
system_id	UUID / GUID	URI Parameter	Yes	e6bb3bf0-3f3e-11e4-916c-0800200c9a66	An Id that uniquely identifies an Embedded System

HTTP Responses

Response	Description
200 (Ok)	The failure report has been accepted and will be recorded.
404 (Not Found)	The specified Embedded System_id is unknown.

Example Request

```
GET /clips/e6bb3bf0-3f3e-11e4-916c-0800200c9a66/report/failure HTTP/1.1

Host: api.p-react.eu
```

- **POST /embeddedsystem/{system_id}/report/tampering**

Description: Report tampering of an Embedded System.
Supported HTTP Methods: POST
Parameters

Name	Data Type	Format	Required	Example Value	Description
system_id	UUID / GUID	URI Parameter	Yes	e6bb3bf0-3f3e-11e4-916c-0800200c9a66	An Id that uniquely identifies an Embedded System

HTTP Responses

Response	Description
200 (Ok)	The tamper report has been accepted and will be recorded
404 (Not Found)	The specified Embedded System_id is unknown.

Example Request

```
GET /clips/e6bb3bf0-3f3e-11e4-916c-0800200c9a66/report/tampering HTTP/1.1
Host: api.p-react.eu
```

- **POST /embeddedsystem/{system_id}/register**

Description: Register the Embedded System with the orchestration module.
Supported HTTP Methods: POST
Parameters

Name	Data Type	Format	Required	Example Value	Description
system_id	UUID / GUID	URI Parameter	Yes	e6bb3bf0-3f3e-11e4-916c-0800200c9a66	An Id that uniquely identifies an Embedded System

HTTP Responses

Response	Description
200 (Ok)	The Embedded System was successfully registered on the system.
404 (Not Found)	The specified Embedded System_id is unknown.

Example Request

```
GET /clips/e6bb3bf0-3f3e-11e4-916c-0800200c9a66/register HTTP/1.1
Host: api.p-react.eu
```

Example Response

```
{
    "status": "registered"
}
```

- **POST /embeddedsystem/{system_id}/unregister**

Description: Unregister the Embedded System from the orchestration module.
Supported HTTP Methods: POST
Parameters

Name	Data Type	Format	Required	Example Value	Description
system_id	UUID / GUID	URI Parameter	Yes	e6bb3bf0-3f3e-11e4-916c-0800200c9a66	An Id that uniquely identifies an Embedded System

HTTP Responses

Response	Description
200 (Ok)	The Embedded System was successfully unregistered on the system.
404 (Not Found)	The specified Embedded System_id is unknown.

Example Request

```
GET /clips/e6bb3bf0-3f3e-11e4-916c-0800200c9a66/register HTTP/1.1
Host: api.p-react.eu
```

Explanation of summary table columns

Method	The HTTP method to be used when requesting the resource
Resource URI	The address of the resource, any portions surrounded by {} are parameters.
Request URIURI	Parameters to be sent as part of the request, these will be passed to the service in the URI of the request.
Request Body	Parameters to be sent as part of the request, these will be included in the body of the request. The layouts of the listed data types are found here API Types
Response HTTP Status Codes	A summary listing of the possible HTTP response codes specific to that resource, a further explanation of the circumstances that would produce these codes is given in the individual resource documentation.
Response Body	The data that will be returned in the body of the response, the layouts of the listed data types are found here API Types
Interface	A reference to the interface that will use this particular resource.

Summary Table

Method	Resource URI	Response HTTP Status Codes	Interface	Description
GET	/notify/new_clip	200 (Ok) 406 (Not Acceptable)	A	Notify Business Logic Component of a new clip event

- **GET /notify/new_clip**

Description: Used to notify the BLC that a new clip has been added to the system.
Supported HTTP Methods: GET
Parameters

Name	Data Type	Format	Required	Example Value	Description
clip	clip	Json	Yes	API Types	The clip object

HTTP Responses

Response	Description

200 (Ok)	The notification was successful and the response payload contains instructions.
406 (Not Acceptable)	The format of the supplied clip object was not valid.

Example Request

```
GET /clips HTTP/1.1
Host: api.p-react.eu
{
    "name": "clip_34.mjpeg",
    "algorithmName": "Fast",
    "dataType": "Video",
    "uri": "https://vcms.p-react.eu/videos/",
    "signature": "d41d8cd98f00b204e9800998ecf8427e",
    "validated": true,
    "startTime": "1410966187",
    "endTime": "1410966310"
}
```

Example Response

```
{
    "action": "analyze"
}
```

ANNEX II. VCMS Functions

Method	*Description*	*Parameters*	*Response*
addUser	Adds a new user Response formats: JSON Requires authentication: Yes Supported request methods: POST Resource family: user	*userName (Mandatory)* *userPassword (Mandatory)* any other User Attribute	userID (on addUser success)
updateUser	Updates a user Response formats: JSON Requires authentication: Yes Supported request methods: POST Resource family: user	userID (Required, Not updatable, First parameter)	userID (on updateUser success)
getUser	Retrieves the entire info from a specific user Response formats: JSON Requires authentication: Yes Supported request methods: GET Resource family: user	userID (Mandatory)	User Object Attributes

Method	Description	Parameters	Response
findUser	Searches the entire users Response formats: JSON Requires authentication: Yes Supported request methods: POST Resource family: user	any other User Attribute	List of userID
deleteUser	Deletes a user Response formats: JSON Requires authentication: Yes Supported request methods: POST Resource family: user	userID (Mandatory)	userID
connect	Connects to the VCMS Server. Response formats: JSON Requires authentication: Yes Supported request methods: POST Resource family: user	userName (Required) userPassword (Mandatory)	1 (connection and login established) -1 (connection failed due to wrong username) -2 (connection failed due to wrong password) -3 (connection failed due to wrong username and password) -4 (connection already exists) -5 (connection failed due to unknown reason.)
disconnect	Disconnect to the VCMS Server. Response formats: JSON Requires authentication: Yes Supported request methods: POST Resource family: user	userName (Required) userPassword (Mandatory)	1 (disconnection and logout succeeded) -4 (disconnection failed since no established connection) -5 (disconnection failed due to unknown reason)
addClipObject	Adds a new clip. Response formats: JSON Requires authentication: Yes Supported request methods: POST Resource family: repository	name (Required) clipData (Mandatory)	clipObjectID

Method	*Description*	*Parameters*	*Response*
updateClipObject	Updates a clip object. Response formats: JSON Requires authentication: Yes Supported request methods: POST Resource family: repository	any Clip Object Attribute clipObjectID (Mandatory, Not updatable, First parameter)	clipObjectID (On updateClipObject success.)
getClipObject	Retrieves the entire info from a specific clip object. Response formats: JSON Requires authentication: Yes Supported request methods: GET Resource family: repository	clipObjectID (Mandatory)	Clip Object Attributes
findClipObject	Searches the entire clip objects. Response formats: JSON Requires authentication: Yes Supported request methods: POST Resource family: repository	Any Clip Object attribute	List of clipObjectID
deleteClipObject	Deletes a clip object. Response formats: JSON Requires authentication: Yes Supported request methods: POST Resource family: repository	clipObjectID (Mandatory)	clipObjectID
playbackClipObject	Playback the video clip. Response formats: JSON Requires authentication: Yes Supported request methods: GET Resource family: player	clipObjectID (Mandatory)	clipObjectID
transcodeClipObject	Transcode clip object video clips from one format to another. Response formats: JSON Requires authentication: Yes Supported request	clipObjectID (Mandatory) videoFormat (The new videoFormat) compression (This parameter used alone with value "yes" or "no")	clipObjectID

Method	Description	Parameters	Response
	methods: POST Resource family: transcoder		
addAnalyticsObject	Adds a new analytics object to the analytics objects database. Response formats: JSON Requires authentication: Yes Supported request methods: POST Resource family: analytics	Analytics Object Attributes analyticsAlrgoID (Mandatory) analyticsAlgoName (Mandatory) analyticsAlgoData (Mandatory)	analyticsObjectID
updateAnalyticsObject	Updates an analytics object. Response formats: JSON Requires authentication: Yes Supported request methods: GET Resource family: analytics	Analytics Object Attributes	analyticsObjectID
getAnalyticsObject	Retrieves the entire info from a specific analytics object to another. Response formats: JSON Requires authentication: Yes Supported request methods: GET Resource family: analytics	analyticsObjectID (Mandatory)	Analytics Object Attributes
findAnalyticsObject	Searches the entire analytics objects. Response formats: JSON Requires authentication: Yes Supported request methods: POST Resource family: analytics	Analytics Object Attributes	List of analyticsObjectID
deleteAnalyticsObject	Deletes an analytics object with specific analyticsObjectID. Response formats: JSON Requires authentication: Yes	analyticsObjectID (Mandatory)	analyticsObjectID

Method	Description	Parameters	Response
	Supported request methods: POST Resource family: analytics		

ANNEX III. References

[1] http://www.cityofchicago.org/city/en/depts/oem/provdrs/tech.html.

[2] http://www.securitymagazine.com/articles/81995-look-mom-no-wires.

[3] http://www-03.ibm.com/press/us/en/pressrelease/22385.wss.

[4] http://www.cbsnews.com/news/big-brother-to-see-all-everywhere/

[5] http://www.naomiklein.org/articles/2008/05/chinas-all-seeing-eye

[6] https://tfl.gov.uk/corporate/privacy-and-cookies/cctv

[7] http://www.adt.com/

[8] http://www.tycois.com

[9] https://cpisecurity.com

[10] https://www.lorextechnology.com/

[11] http://www.q-see.com/

[12] http://www.guardianprotection.com/

[13] A. Pal, G. Schaefer, M.E. Celebi, "Robust codebook-based video background subtraction", IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), 2010.

[14] T.S.F. Haines, Tao Xiang, "Background Subtraction with Dirichlet Process Mixture Models,", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.36, no.4, pp.670,683, April 2014

[15] Dalal, N.; Triggs, B., "Histograms of oriented gradients for human detection," IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) 2005

[16] Xiaoqin Zhang; Weiming Hu; Wei Qu; Maybank, S., "Multiple Object Tracking Via Species-Based Particle Swarm Optimization", IEEE Transactions on Circuits and Systems for Video Technology, vol.20, no.11, pp.1590,1602, Nov. 2010

[17] Sakaino, H., "Video-Based Tracking, Learning, and Recognition Method for Multiple Moving Objects", IEEE Transactions on Circuits and Systems for Video Technology, vol.23, no.10, pp.1661,1674, Oct. 2013

[18] Wang, Heng; Kläser, Alexander; Schmid, Cordelia; Liu, Cheng-Lin, "Dense Trajectories and Motion Boundary Descriptors for Action Recognition", International Journal of Computer Vision, vol.103, no.1, pp.60,79, 2013

[19] Gorelick, L.; Blank, M.; Shechtman, E.; Irani, M.; Basri, R., "Actions as Space-Time Shapes", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, no. 12, pp.2247,2253, Dec. 2007

[20] Rui Zhao; Wanli Ouyang; Xiaogang Wang, "Unsupervised Salience Learning for Person Re-identification", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013

[21] Farenzena, M.; Bazzani, L.; Perina, A.; Murino, V.; Cristani, M., "Person re-identification by symmetry-driven accumulation of local features," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.2360,2367, 2010

[22] L. Bygrave, "Minding the machine : Article 15 of the EC Data Protection Directive and Automated," Computer Law & Security Report, vol. 17, pp. 17-24, 2001.

[23] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1999.

[24] N. Oliver, B. Rosario and A. Pentland, "A Bayesian computer vision system for modelling human interactions," IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000.

[25] R. C. Gonzalez and R. E. Woods, Digital Image Processing, Prentice Hall, 2007.

[26] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in Imaging Understanding Workshop, 1981.

[27] G. Farneback, "Two-Frame Motion Estimation Based on Polynomial Expansion," in Scandinavian Conference on Image Analysis, 2003.

[28] C. Harris and M. Stephens, "A combined corner and edge detector," in Alvey Vision Conference, 1988.

[29] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005.

[30] K. Fragkiadaki, W. Zhang, G. Zhang and J. Shi, "Two-Granularity Tracking: Mediating Trajectory and Detection Graphs for Tracking under Occlusions," in European Conference on Computer Vision, 2012.

[31] C. M. Bishop, Pattern Recognition and Machine Learning, Springer-Verlag New York, Inc., 2006.

[32] X. Cui, Q. Liu, M. Gao and D. N. Metaxas, "Abnormal detection using interaction energy potentials," in IEEE Conference on Computer Vision and Pattern Recognition, 2011.

[33] Nick Rozanski and Eóin Woods. 2005. Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives. Addison-Wesley Professional.

[34] http://www.microsoft.com/en-us/kinectforwindows/develop/

[35] Khoshelham, K. and Oude Elberink, S.J. (2012) Accuracy and resolution of Kinect depth data for indoor mapping applications. In: Sensors: journal on the science and technology of sensors and biosensors: open access, 12 (2012)2 pp. 1437-1454.