A Multi-Agency Command & Control System Prototype

Work Package 4

Deliverable Report 4.4

Acknowledgements

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under Grant agreement n° 603663 for the research project PEARL (Preparing for Extreme And Rare events in coastaL regions).

Disclaimer

The deliverable D 4.4 reflects only the authors' views and the European Union is not liable for any use that may be made of the information contained herein.

pearl
Preparing for Extreme And Rare events in coastaL regions

Authors:

Marios Moutzouris, Lefteris Voumvourakis, Antonis Kostaridis Dimitris Diagourtas (STWS)

Document Information

Project Number	***603663***	Acronym	**PEARL**
Full Title	Preparing for Extreme and Rare events in coastaL regions		
Project URL	http://www.pearl-fp7.eu/		
Document URL			
EU Project Officer	Denis Peter		

Deliverable	Number	D. 4.4	Title	Multi-Agency Command & Control system prototype
Work Package	Number	WP4	Title	Flood forecasting and early warning systems for coastal regions

Date of Delivery	Contractual	06/31/2016	Actual	06/31/2016
Status	version 1.0		final □	
Nature	prototype □ report □ dissemination □			
Dissemination level	public □ consortium □			

Abstract (for dissemination, 100 words)	This document presents the design and development of an alert routing software (ARS) that has been based on the Emergency Data Exchange Language (EDXL) specification. The alert router facilitates the implementation of standards for public alerting and hazard notification in disasters and emergency situations. As such it allows the automatic distribution based on various rules but also provides the ability to ad-hoc distribution to recipients from the systems and actors registered on the online alert routing system. For testing and validation purposes the alert routing system has been interfaced with an existing Command & Control system for Emergency Responders (ENGAGE IMS/CAD) and the Marbella Case Study Flood Early Warning System developed in the frame of PEARL from Hyds in order to demonstrate the ability to interlink Early Warning Service Providers and Emergency Responders.
Keywords	Multi-agency coordination, early warning, alert dissemination, common alerting protocol

Version Log				
Issue Date	Rev.No.	Author	Change	Approved by
06-06-2016	0.1	Lefteris Vouvmourakis	First draft	
10-06-2016	0.2	Lefteris Voumourakis	Design	
13-06-2016	0.3	Lefteris Voumourakis	Architectural strategies	
17-06-206	0.4	Marios Moutzouris	Technologies	
20-06-2016	0.5	Lefteris Voumourakis	CAD system description	
24-06-2016	0.6	Lefteris Voumourakis	EDXL standards	
27-06-2016	0.7	Marios Moutzouris	ARS architecture and design	
29-06-2016	0.8	Marios Moutzouris	Integration of C2 and ARS	
30-06-2016	0.9	Antonis Kostaridis	Ready for review	Dimitris Diagourtas

Abbreviations List	
Term	**Description**
API	Application Programming Interface
ARS	Alert Routing System
C2	Command & Control
CAD	Computer Aided Dispatch
CAP	Common Alerting Protocol
COTS	Commercial off-the-shelf
DoW	Description of Work
DC	Design Consideration
DS	Design Strategy
DE	Distribution Element
DPT	Design Policy and Tactic
EDXL	Emergency Description Exchange Language
EFAS	European Flood Awareness System
EMS	Emergency Management Service
GIS	Geographical Information System
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IMS	Incident Management System
JDBC	Java Database Connectivity
JEE	Java Enterprise Edition
JMS	Java Messaging Service
JNDI	Java Naming Directory Interface
JRE	Java Runtime Environment
OASIS	Organization for the Advancement of Structured Information Standards
ORM	Object Relational Mapping

OSGi	Open Services Gateway Initiative
RCP	Rich Client Platform
REST	Representational state transfer
RIA	Rich Internet Application
RMI	Remote Method Invocation
UML	Unified Modeling Language
URN	Uniform Resource Name
XML	Extensible Markup Language

Contents

1 Introduction

Around the globe, natural hazards have been on the increase in recent years resulting in loss of human lives, displacement of millions of people, and destruction of critical infrastructure. Preparedness and early warning are critical elements of managing these disasters and reducing loss of human lives to the bare minimum. With adequate warning, people can act to reduce the damage and loss of life caused by natural and manmade hazard events. The key is to get timely and appropriate alerts to everyone who needs them and to only those who need them. Nonetheless, appropriate and complete alerting is a complex challenge.

The PEARL deliverable 4-4 “A multi-agency command & control system prototype” is based on task 4.4 “Enabling first responders to receive and use efficiently the early warning information” of the PEARL DoW (European Commission Directorate-General for Research and Innovation 2013). Task 4.4 contributes to the objectives of WP4 for improving the state of art in flood forecasting and early warning for coastal areas across the whole chain of early warning from data to warning dissemination and aims to design and develop a multi-agency command & control system capable to receive and manipulate early warning and alerting messages.

Early warning is an essential aspect of effective preparedness in relation to emergency management. For instance the Early Warning component of the Copernicus Emergency Management Service (EMS) aims to improve the preparedness, and therefore the responsiveness, of national authorities in relation to floods and wildfires. It provides among others alerts related to floods via the European Flood Awareness System (EFAS) [1].

The work in Task 4.4 as described in the DoW is to design and develop a standards based alert routing software system able to interlink early warning service providers and emergency responders. Most Public Safety agencies in Europe operate COTS or in-house built Incident Management and Command & Control Systems that are not interoperable. In this respect work in this task aims to design and develop an alert routing system which will also serve as the interoperability layer between emergency management systems at local, regional, national and cross border level.

The design and development of the alert routing software has been based on the Emergency Data Exchange Language specification and more specifically on the Distribution Element [2] and Common Alerting Protocol [3] standard with the goal to effectively receive, decode and distribute early warning and alerting messages. The EDXL-CAP addresses the long-standing need to coordinate the information content across all of the mechanisms used for warnings and alerts and is maintained by the Emergency Management Technical Committee of the Organization for the Advancement of Structured Information Standards (OASIS) [4].

The alert router facilitates the implementation of standards for public alerting and hazard notification in disasters and emergency situations. As such it allows the automatic distribution based on various rules but also provides the ability to ad-hoc distribution to recipients from the systems and actors registered on the online alert routing system.

For testing and validation purposes the alert routing system has been interfaced with an existing Command & Control system for Emergency Responders (ENGAGE IMS/CAD) and the Marbella Case Study Flood Early Warning System developed in the frame of PEARL from Hyds in order to demonstrate the ability to interlink Early Warning Service Providers and Emergency Responders.

The rest of the document is structured as follows: Section 2 presents the system overview and Section 3 the key design considerations. In Section 4, the architectural strategies selected to meet the design considerations as well as the adopted emergency management standards are present. Section 5 describes the design and development of the Alert Routing System, section 6 presents in a high level manner the Incident Management System that served as a proof of concept for the realization of the Multi-agency C2 prototype. Finally in Section 7 the integration aspects are documented in more detail.

2 System Overview

Traditionally, during a disaster response Public Safety agencies rely on voice over radio communication along with pen and paper notes for situational awareness. Work in Task 4.4 of the PEARL project is centred around emergency data interoperability that seeks to connect early warning service providers, decision makers, operators/responders and additional stakeholders through the design and development of an alert dissemination system for multi-agency situational awareness based on standard data messaging formats.

The alert dissemination system is providing the necessary means for routing and exposure of alert emergency message data in ways that are complimentary to the current business processes of public safety agencies. Through this prototyping effort we aim to provide a common messaging platform that is able to connect early warning and emergency systems such as Incident Management, Computer Aided Dispatch (CAD) and Mobile Data Terminals (MDTs).

The overall system is depicted in the following Figure.

Figure 1: System overview

3 Design Considerations

The Multi-Agency Command & Control system prototype has been realized as an Enterprise Incident Management System that is able to exchange early warning and incident information not only between separate and distributed installations but also with Early Warning Service providers and potentially with 3rd party C2 systems of public safety agencies.

In order to meet this basic design objective a generic loose coupler in the form of an Alert Routing System (ARS) has been designed and developed that is based on open standards. As a proof of concept this software has been interfaced with the C2 system ENGAGE IMS/CAD developed by Satways Ltd. by means of a dedicated message agent that is able to interface both the Alert Routing software and the back-end services of ENGAGE. In addition user interface enhancements were required in order to couple the exchanged alerts and the internal incident object model of the C2 system.

This criteria that were taken into account in order to design the overall multi-agency command & control prototype are presented in the following table:

Table 1. Design Considerations

Code	Design Consideration	Description
DC.1	Reuse of existing software components	The Multi-Agency C2 system should be based on the existing incident management system (ENGAGE IMS/CAD) which is going to be interfaced with an Alert Routing Software (ARS). The latter will provide the mechanisms to route alerts from Early Warning Service Providers and facilitate the event information exchange between instances running in different Public Safety Agencies.
DC.2	Ability to distribute emergency messages between Public Safety Agencies	The C2 prototype should provide the ability to exchange event information with other instances of the same software or with 3rd party incident and crisis management systems.
DC.3	Ability for users to choose at runtime the message distribution endpoints	Public Safety operators or Early Warning Service providers need to have the ability to choose at runtime the emergency message or alert recipients according to their needs at runtime.
DC.4	Conformance to standards	The ARS system should be based on open standards wherever possible
DC.5	Platform Independence	The ARS is platform independent and should be installed in Linux and/or Windows Operating Systems.
DC.6	Distribution of Alert messages based on Geographic region	The ARS system should be capable to execute spatial rules for message distribution
DC.7	Encapsulation and dissemination of one or more alerts or notifications	The ARS system should be able to handle multiple message payloads in a single message transmission

DC.8	Modularity and Extensibility	The ARS design needs to be fully modular, allowing to be extended with additional services. In addition it needs to extensible in terms of the message payloads that can support.
DC.9	Web Services API	The ARS system should provide a well-defined and documented Application Programming Interface based on the Web Services programming paradigm. External Early Warning or C2 systems that need to interconnect with the software router should utilizes these Web Services.
DC.10	Development Simplicity	Provide the means for 3rd party software system to easily interface the ARS system
DC.11	Scalability and Performance	The overall system should be designed is focused on the maximization of performance without requesting too many CPU or memory resources. The ARS software should be run on a standard PC (2.5GHz processor with 4GB of RAM and a 500GB hard drive). In addition when many systems are integrated and many messages are exchanged the system should provide fail-over and load balancing capabilities.
DC.12	Event Driven Architecture	The event-driven architecture is a popular distributed asynchronous architecture pattern used to produce highly scalable applications. It can be used for both small and large, complex applications. The pattern is made up of highly decoupled, single-purpose event processing components that asynchronously receive and process events

4 Architectural Strategies and Technologies

In the section we describe the main decisions and/or strategies that affect the overall organization of the alert routing software and its higher-level structures. These strategies provide direction for the key abstractions and mechanisms used in the system architecture.

The following Table summarizes the decisions and strategies. The reasoning leading to each outcome is described by reference to the stated goals and principles of the previous Section. Additionally, any design goals or priorities that were balanced or traded-off in the selection process are also detailed.

Table 2. Architectural Decisions and Strategies

Code	Decision / Strategy	Description	Design Considerations Ref. Codes
ADS.1	Java Coding Language	The Java Language is one of the most popular programming languages in use; particularly for client-server web applications. This choice was informed by both platform independence and the modularity and extensibility offered by the JVM. In this decision we traded-off the potentially enhanced performance of other languages against the flexibility, modularity and extensibility offered by Java/OSGi which are the underlying technologies of the ENGAGE IMS/CAD system.	DC.1 – "Reuse of existing software components", DC.5 – "Platform Independence"
ADS.2	Java Enterprise Edition	The JEE is a widely used enterprise computing platform developed under the Java Community Process. The platform provides an API and runtime environment for developing and running enterprise software, including network and web services, and other large-scale, multi-tiered, scalable, reliable, and secure network applications.	DC.5 – "Platform Independence" DC.9 – "Web Services API"
ADS.3	Object Relational Mapping	ORM enables developers to more easily write applications whose data outlives the application process. An Object/Relational Mapping (ORM) framework is concerned with data persistence as it applies to relational databases (via JDBC).	DC.10 – "Development Simplicity"
ADS.4	Service Platform	The OSGi Service Platform is the de-facto standard for modularised Java Language. It is a framework that provides a dynamic environment for the deployment of services and modules (referred as bundles in OSGi terminology).	DC.8 – "Modularity and Extensibility"
ADS.5	Message Bus	The alert routing system will exchange alert messages with endpoints (publishers and recipients) via a Message Bus system and a reliable messaging scheme.	DC.12 – "Event Driven Architecture"

ADS.6	Emergency Management Standards	Adoption of the Distribution Element and Common Alerting Protocol standards of the Emergency Management Description Language family.	DC.2 – "Ability to distribute emergency messages between Public Safety Agencies", DC.3 – "Ability for users to choose at runtime the message distribution endpoints", DC.4 – "Conformance to standards", DC.6 – "Distribution of Alert messages based on Geographic region", DC.7 – "Encapsulation and dissemination of one or more alerts or notifications"

In the following subsections the main architectural strategies and technologies are presented in more detail.

4.1 Open Services Gateway Initiative

The OSGi [5] technology is a set of specifications that define a dynamic component system for Java. These specifications enable a development model where applications are (dynamically) composed of many different (reusable) components. The OSGi specifications enable components to hide their implementations from other components while communicating through services, which are objects that are specifically shared between components.

A practical advantage of OSGi is that every software component can define its API via a set of exported Java packages and that every component can specify its required dependencies. The components and services can be dynamically installed, activated, de-activated, updated, and de-installed. The OSGi specification defines a bundle as the smallest unit of modularization, i.e., in OSGi, a software component is a bundle.

The OSGi has a layered model that is depicted in the following figure.

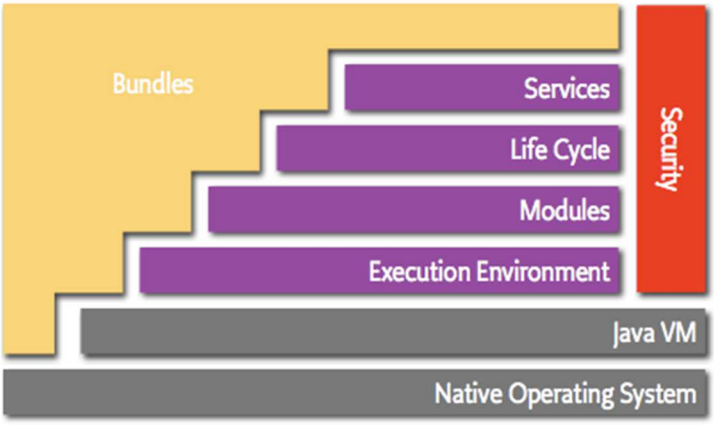


Figure 2: OSGi layered model

The following list contains a short definition of the terms:

- Bundles – Bundles are the OSGi components made by the developers.
- Services – The services layer connects bundles in a dynamic way by offering a publish-find-bind model for plain old Java objects.
- Life-Cycle – The API to install, start, stop, update, and uninstall bundles.
- Modules – The layer that defines how a bundle can import and export code.
- Security – The layer that handles the security aspects.
- Execution Environment – Defines what methods and classes are available in a specific platform.

Modules

The fundamental concept that enables such a system is modularity. Modularity, simply put, is about assuming less. Modularity is about keeping things local and not sharing. It is hard to be wrong about things you have no knowledge of and make no assumptions about them. Therefore, modularity is at the core of the OSGi specifications and embodied in the bundle concept. Though the code hiding and explicit sharing provides many benefits (for example, allowing multiple versions of the same library being used in a single Virtual Machine), the code sharing exists only to support the OSGi services model. The services model is concerned with bundles that collaborate.

Deployment

Bundles are deployed on an OSGi framework: the bundle runtime environment. It is a collaborative environment within which bundles run in the same VM and can actually share code. The framework uses the explicit imports and exports exposed by each bundle to wire up the bundles so they do not have to concern themselves with class loading. Moreover, the management of the framework is standardised. A simple API allows bundles to install, start, stop, and update other bundles, as well as enumerating the bundles and their service usage. Many management agents have used this API methodology to control OSGi frameworks.

4.2 Java Enterprise Edition

JEE (Java Enterprise Edition) [6] is a Java platform designed for the mainframe-scale computing typical of large enterprises. Sun Microsystems (together with industry partners such as IBM) designed JEE to simplify application development in a thin-client, tiered environment. Java Enterprise Edition, Java EE, or JEE is a widely used enterprise-computing platform developed under the Java Community Process. The platform provides an API and runtime environment for the development and execution of enterprise software; including network and web services as well as other large-scale, multi-tiered, scalable, reliable, and secure network applications. JEE simplifies application development and decreases the need for programming and programmer training by creating standardised, reusable, modular components and by enabling the enterprise tier to handle many aspects of programming automatically. Java EE includes several API specifications, such as RMI, e-mail, JMS, web services, XML, etc., and defines how these should be co-ordinated. Java EE also features some unique component specifications. These include Enterprise JavaBeans, connectors, servlets, Java Server Pages (JSP), and several web service technologies. This allows developers to create enterprise applications that are portable and scalable and that integrate with legacy technologies. A Java EE application server can handle transactions, security, scalability, concurrency and management of the components deployed within it. This enables developers to concentrate more on the business logic of the components rather than on infrastructure and integration tasks.

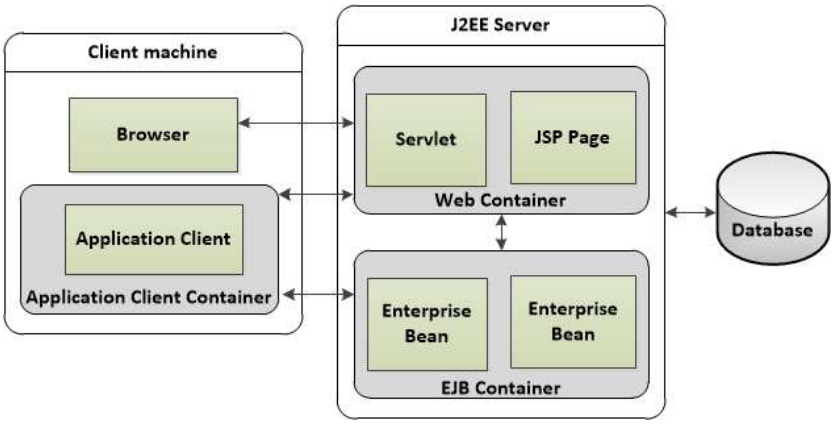


Figure 3: JEE Architecture

Normally, thin-client multi-tiered applications are hard to write because they involve many lines of intricate code to handle transaction and state management, multithreading, resource pooling, and other complex low-level details. The component-based and platform-independent JEE architecture makes JEE applications easy to write because business logic is organised into reusable components. In addition, the JEE server provides underlying services in the form of a container for every component type. Because developers are not developing these services themselves, they are free to concentrate on solving the business problem at hand.

Containers

Containers are the interface between a component and the low-level platform-specific functionality that supports the component. Before a Web, enterprise bean, or application client

component can be executed, it must be assembled into a JEE application and deployed into its container.

The assembly process involves specifying container settings for each component in the JEE application and for the JEE application itself. Container settings customize the underlying support provided by the JEE server, which includes services such as security, transaction management, Java Naming and Directory Interface (JNDI) lookups, and remote connectivity. Here are some of the highlights:

- The JEE security model lets you configure a Web component or enterprise bean so that system resources are accessed only by authorised users.
- The JEE transaction model lets you specify relationships among methods that make up a single transaction so that all methods in one transaction are treated as a single unit.
- JNDI lookup services provide a unified interface to multiple naming and directory services in the enterprise so that application components can access naming and directory services.
- The JEE remote connectivity model manages low-level communications between clients and enterprise beans. After an enterprise bean is created, a client invokes methods on it as if it were in the same virtual machine.

The fact that the JEE architecture provides configurable services means that application components within the same JEE application can behave differently based on where they are deployed. For example, an enterprise bean can have security settings that allow it a certain level of access to database data in one production environment and another level of database access in another production environment.

The container also manages non-configurable services such as enterprise bean and servlet life cycles, database connection resource pooling, data persistence, and access to the JEE platform APIs.

4.3 Object-relational Mapping

Object-relational mapping (ORM, O/RM, and O/R mapping) in computer science is a programming technique for converting data between incompatible type systems in object-oriented programming languages. Object-relational mapping (ORM) is a mechanism that makes it possible to address, access and manipulate objects without having to consider how those objects relate to their data sources. ORM lets programmers maintain a consistent view of objects over time, even as the sources that deliver them, the sinks that receive them, and the applications that access them change. Based on abstraction, ORM manages the mapping details between a set of objects and underlying relational databases, XML repositories, or other data sources and sinks while simultaneously hiding the more volatile details of related interfaces from developers and the code they create. ORM encapsulates and abstracts change in the data source itself, so that when data sources or their APIs change, only ORM needs to change to keep up—not the applications that use ORM to insulate themselves from this kind of effort. This capacity lets developers take advantage of new classes as they become available and also

makes it easy to extend ORM-based applications. In many cases, ORM changes can incorporate new technology and capability without requiring changes to the code for related applications.

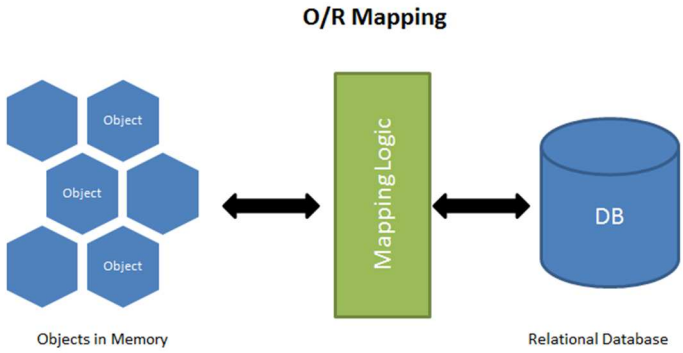


Figure 4: Object-relational Mapping

Many ORM frameworks exist and Hibernate is the framework of choice. In addition to its own "native" API, Hibernate is also an implementation of the Java Persistence API (JPA) specification. As such, it can be easily used in any environment supporting JPA including Java SE applications, Java EE application servers, Enterprise OSGi containers, etc.

4.4 Adopted Emergency Management Standards

The design and development of the multi-Agency C2 prototype from the interoperability point of view is driven from the data model and implementation of existing EDXL standards namely the Emergency Data Exchange Language – Distribution Element (EDXL-DE) and the Emergency Data Exchange Language – Common Alerting Protocol (EDXL-CAP).

The ARS prototype leverages the encapsulation and routing capabilities of the EDXL-DE as the system's top-level loose coupler that allows for processing of message payloads in the form of the EDXL-CAP specification.

The EDXL-DE standard was developed by OASIS and "allows an organization to wrap separate but related pieces of information into a single package for easier and more useful distribution. This standard provides a set of differentiated features when compared with other standards that implement the compositional design pattern.

The EDXL-DE header provides sender and recipient information which enables the receiving system to "reply" back to the sender by merely "swapping" the fields. The diagram below illustrates the EDXL-DE object module.

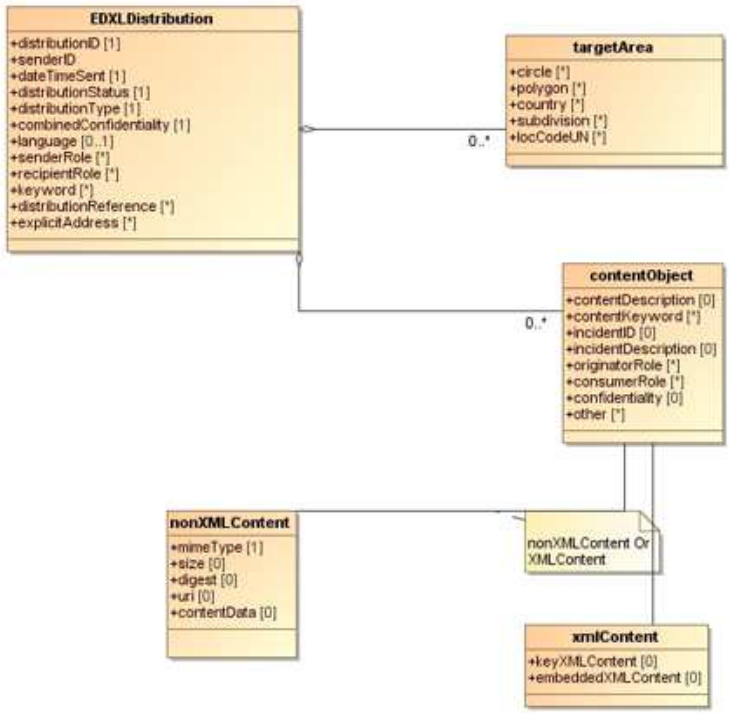


Figure 5: The Distribution Element Class Diagram

The EDXL-CAP (Emergency Management Description Language - Common Alerting Protocol) is an XML-based format for exchanging public warnings and emergencies between alerting systems.

The alert is made up of a number of information blocks which are illustrated in the following Figure. These blocks contain all required information fields describing the alert's properties as explained in the next paragraphs analytically.

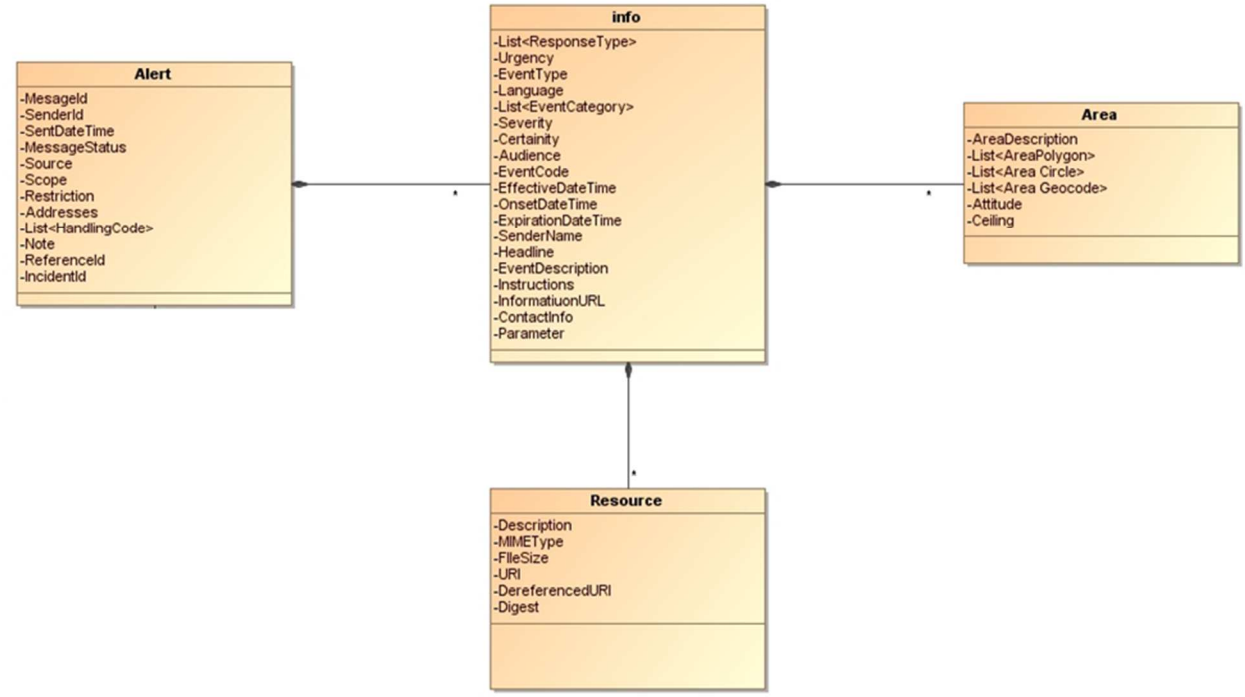


Figure 6: CAP Alert Class Diagram

Each EDXL-CAP Alert Message consists of an <alert> segment, which may contain one or more <info> segments, each of which may include one or more <area> and/or <resource> segments. Under most circumstances EDXL-CAP messages with a <msgType> value of “Alert” should include at least one <info> element.

The <alert> segment provides basic information about the current message: its purpose, its source and its status, as well as a unique identifier for the current message and links to any other, related messages. An <alert> segment may be used alone for message acknowledgements, cancellations or other system functions, but most <alert> segments will include at least one <info> segment.

The <info> segment describes an anticipated or actual event in terms of its urgency (time available to prepare), severity (intensity of impact) and certainty (confidence in the observation or prediction), as well as providing both categorical and textual descriptions of the subject event. It may also provide instructions for appropriate response by message recipients and various other details (hazard duration, technical parameters, contact information, links to additional information sources, etc.) Multiple <info> segments may be used to describe differing parameters (e.g., for different probability or intensity “bands”) or to provide the information in multiple languages.

The alert resource segment provides an optional reference to additional information related to the <info> segment within which it appears in the form of a digital asset such as an image or audio file.

The <area> segment describes a geographic area to which the <info> segment in which it appears applies. Textual and coded descriptions (such as postal codes) are supported, but the preferred representations use geospatial shapes (polygons and circles) and an altitude or

altitude range, expressed in standard latitude / longitude / altitude terms in accordance with a specified geospatial datum.

5 The Alert Routing Software

The Alert Routing Software has been designed as enterprise and cross-platform software application which enables data messages to be routed according to their header information. Each agency or organization is “seen” by the ARS as an endpoint.

I consists of the Web Module, the EJB module, the Message Bus and backend database. The Web module provides the REST application programming interface [8] to the client (endpoints). Via this interface endpoints can register themselves and query information from the router. The Enterprise Java Beans module contains stateless session beans for persisting data into the backend database.

An ARS endpoint can a message publisher, a message consumer of both. The actual message exchange between a client and the ARS is accomplished asynchronously via the Message Bus.

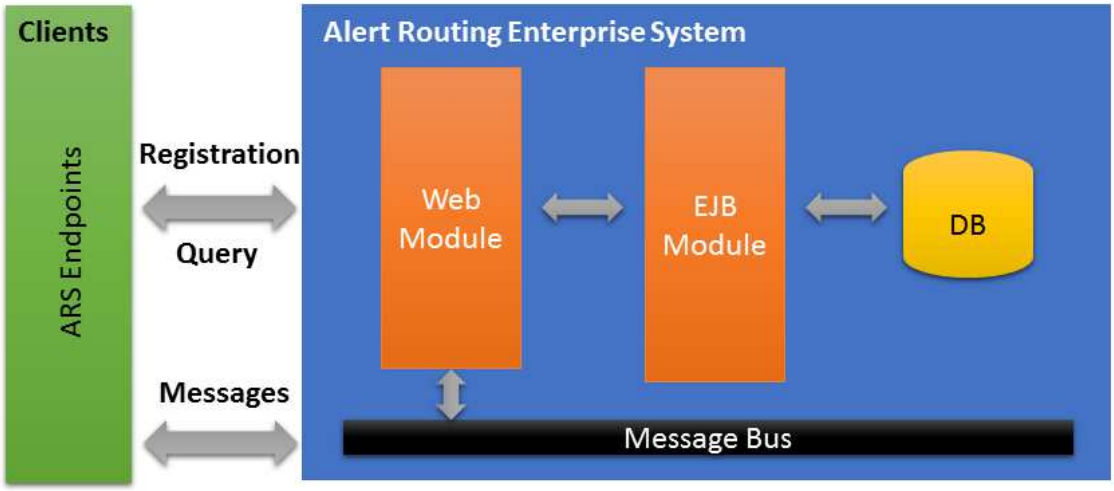


Figure 7: The ARS high level architecture

5.1 Provided Services

The Web Module of the Alert Routing Software device provides the following services:

Registration Service – Each endpoint has a responsibility to “announce” itself to the ARS along with their reachability attributes:

- Uniform resource name (URN)
- Descriptive Name
- Attributes and Method of communicating with the end point (currently only via a Message Broker)
- The alert messages they wish to receive based on
    - Type
    - Geographical areas of interest
    - Keywords

The Registration service upon receiving a registration request will check the validity of the information provided by the endpoint. For example whether the URN has been pre-registered etc. If the information is valid it will be entered as a valid endpoint.

Message Service – Each endpoint or any generator of information contacts the receiver and hands over the message for delivery specifying the recipients. The following process will be followed on reception:

1. Check if the publisher and consumer are valid
2. Check if the consumer endpoint is available
3. Check if there is geographical information and keywords about the message
4. Send message to endpoint(s) which match
5. Cache message should the endpoint not be available

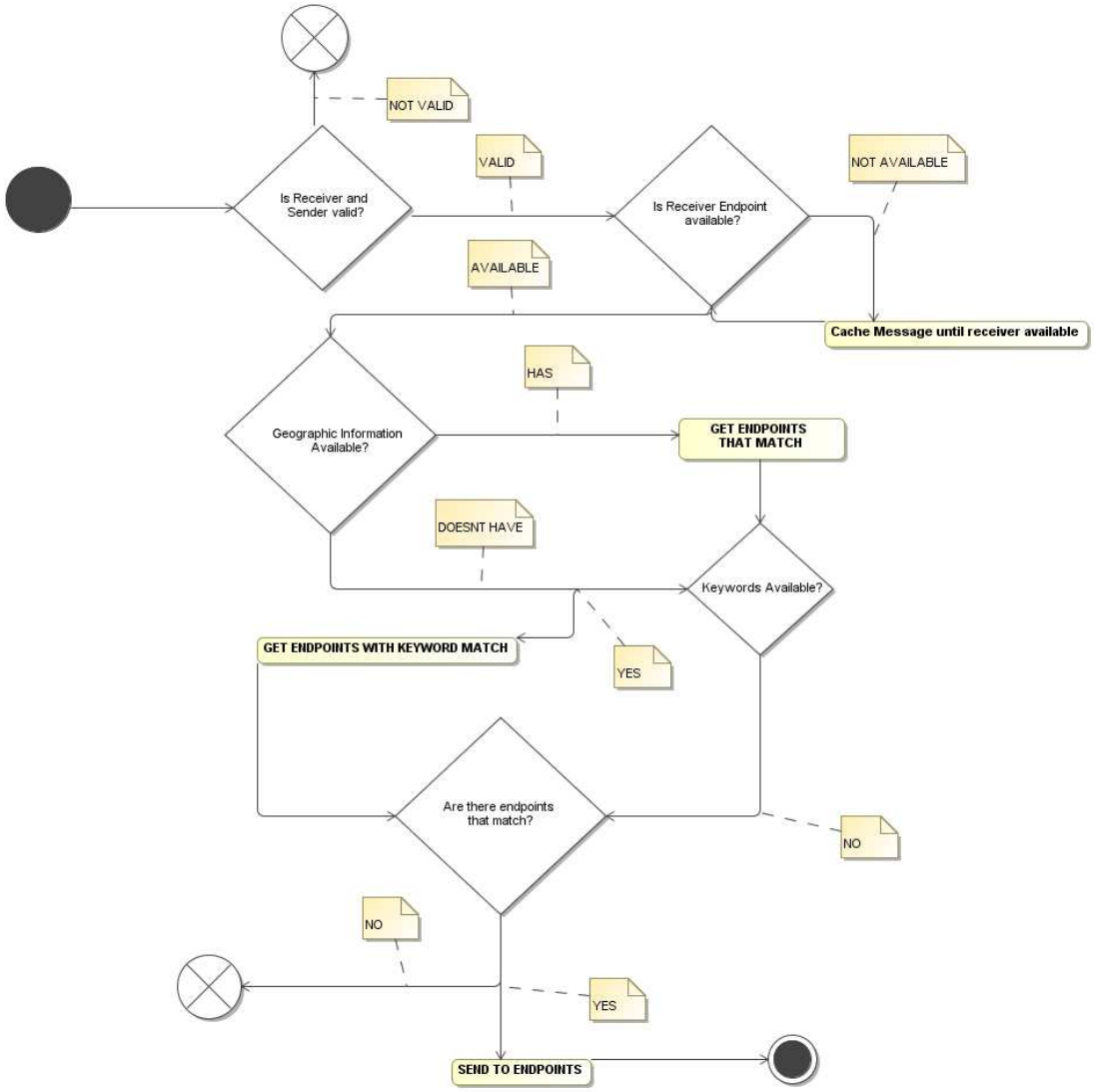


Figure 8: The ARS State Machine Diagram

Query Service – The Query service allows endpoints to query which other endpoints have been registered and are available in the ARS. This allows the endpoints to insert valid recipient information. The Query Service has a lookup table which is populated by data from the Registration service.

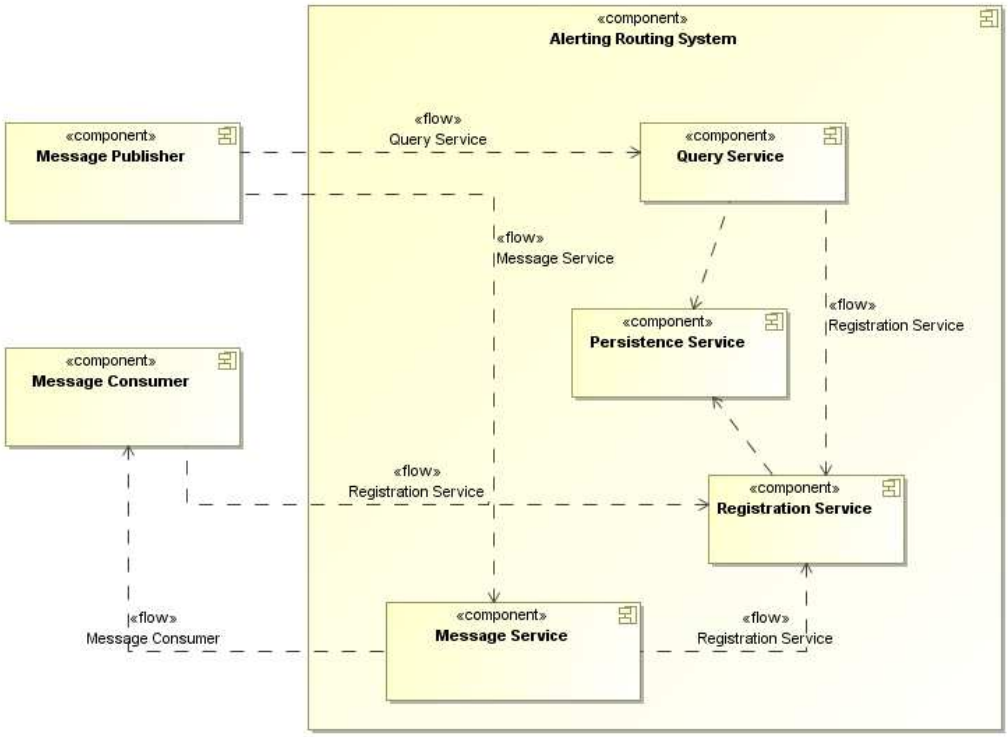


Figure 9: The ARS component diagram

Each of the aforementioned services accepts messages using the HTTP Post method. All services will ensure that a reply is provided back to the request.

- Registration – Whether or not the information has been saved
- Messaging – Whether the message has been sent to the recipient or if there is an error during processing or transmission.
- Query – The available endpoints

Persistence Service – The persistence service persists the information of endpoints and statistics information to a relational database.

Statistics Service – The statistics service can be accessed by following a link in a web browser. This page provides a list of registered endpoints and their current incoming and outgoing messages.

The ARS uses the EDXL-DE to route payloads (EDXL-CAP messages) to the proper recipient(s). The EDXL-DE header provides sender and recipient information which enables the consuming system to “reply” back to the sender by merely “swapping” the fields.

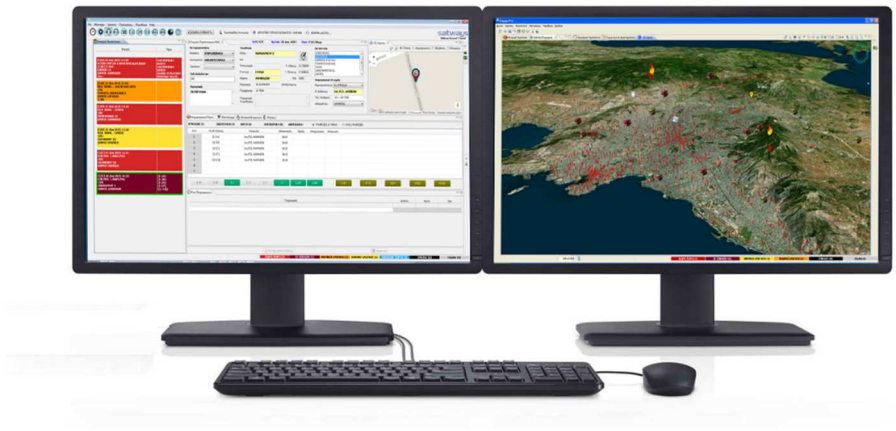
6 The ENGAGE IMS/CAD

The ENGAGE Incident Management & Computer Aided Dispatch constitutes an integrated Call-Center solution for public & private safety organizations providing all the tools for call & incident management, computer aided dispatch, operational resource management and disparate crucial information data integration.

Combining advanced searching; filtering in current and historical data and geo-correlation of data operations are enhanced with situational awareness, decision support and electronic logging of incident information and related actions of the involved organizations.

ENGAGE in general is designed for operational use by professional organizations employing mobile resources, such as Police, Fire Departments, Rescue Services, Emergency services, Security Departments etc.

Based on a highly modular and reconfigurable S/W platform and a reliable, distributed Event Driven architecture, ENGAGE supports comprehensive incident control and dispatching for Public Safety offering an unmatched combination of speed, reliability, and features adaptive to highly complex communication environments.



6.1 Main Modules

The ENGAGE IMS/CAD is a modular and extensible enterprise software system based on the OSGi technology. The system has been used as a proof of concept in the frame of Task 4.4 towards a Multi-Agency and interoperable C2 system during emergency operations related to floods. The main modules of the system are described in the following paragraphs while section 8 describes the adaptations and developments that took place in order to integrate the software with the ARS in order to extend it with multi-agency characteristics and the ability to link with Flood Early Warning Service providers.

Call Taking

The trigger source of a call taking procedure can be a regular telephone call, an emergency call or an alarm event that has been occurred by the interconnected physical security and safety systems. For instance if an alarm is received from an interconnected physical security or safety system the alarm type, location and triggered date/time is automatically entered in the relevant fields.

CTI Integration

Any regular or emergency call is processed through ENGAGE. Even before the call is answered, the system displays the caller's number, the number called and the service the call should be directed to- if the system covers multiple emergency or service

Call Assist

It allows the call-taker role to ensure rapid assistance to the caller or decisions to take as regards the disposition of a call. Every answer pulls an associated Action in this field, such as setting the procedure or increasing the Priority of the selected process.

Integrated Mapping

ENGAGE provides easy access to publicly available maps by supporting different online mapping engines. In addition it supports the interaction with privately held data via ArcGIS Server or via the embedded 3D Enterprise GIS for maximum performance and massive spatial data visualization and management.

Incident Management

Allows personnel to manage the whole incident life-cycle, assign and dispatch resources and execute measures according to Standard Operating Procedures (SOPs). Intuitive views, forms and filters provide an easy to use user interface with less required user actions and complete audit control.

Resource Management & Rostering

ENGAGE incorporates a versatile and fully configurable Operational Resource Management Module that manages one or more organization's resources for incidence response. The module provides the capability to register, plan, allocate and assign resources to specific incidents either manually or automatically through the Resource Allocation functionality that combines and analyzes the suitability of one or more resources based on resource responsibility area, capabilities and incident location proximity.

Computer Aided Dispatch

In the event of an alert, ENGAGE, taking into consideration the nature of the incident, its location, the time constraints this imposes, and other factors, identifies the closest resource available. It is also possible for several alert routes to be simultaneously activated for one or

more resources. Alerts can vary depending on the time of day, and can also be predefined for any given time intervals.

Resource Tracking

Monitors resources, such as vehicles vessels and aircrafts by offering seamless integration with multiple tracking technologies over multiple communication bearers.

6.2 Architecture

From the system architecture point of view the ENGAGE IMS/CAD system is N-Tier Enterprise software application that consists on the following subsystems:

- **Application Server**: The ENGAGE Application Server is the core of the system, running all server side Business Logic. JBoss Application server is the main component. It stands between workstations or mobile data terminals and the ENGAGE Database Server, handling requests and storing and retrieving data, doing all necessary validations and actions. Communication with the Operator Workstations uses Enterprise Java Beans remote method invocations (RMI) while the ENGAGE Mobile Data Terminal utilizes the Web Services SDK for request/response based data exchanges and the Message Broker for push notifications. Connector Services are standalone Java services, their primary task is to communicate and exchange data with external systems. Finally the Message Broker is used to provide asynchronous message exchange capabilities between the internal components and the Operator Client RIA application.

- **Database Server**: The Database Server stores all configuration and runtime data of the system. PostgreSQL is the Relational Database System used. It is also extended with PostGIS to support geographical data structures and spatial queries.

- **3D GIS Server**: The GIS Server accesses geographical data (3D terrain databases and vector layers), used for visually presenting geographical information to the EOC client application. It will comprise the basic component of the Common Operational Picture. The Terrain Streaming Server streams effectively terrain raster data while the Streaming Feature Server (SFS) streams vector layers from the Geodatabase.

- **ENGAGE Client**: The Operator Workstation is the hosting device of the ENGAGE IMS/CAD Client application. The latter is be a multi-screen Rich Internet Application for the various EOCs and the prime system the operators interact with.

- **ENGAGE Mobile Data Terminal**: This is a special version of the ENGAGE Client that runs on smartphones and can be used by public safety personnel on the field. The mobile data terminal is able exchange real time information with operators of Emergency Operation Centers.

6.3 Developed Extensions

In order to interface the ENGAGE IMS/CAD with the ARS and thus with Early Warning Service providers additional OSGi client plugins, server component and services have developed on the ENGAGE system.

The objective is to be able to receive CAP alerts from Early Warning Service providers or other public Safety agencies and disseminate CAP alerts back to them without altering the internal incident object model. Thus appropriate mechanisms that translate CAP alerts into incidents (and vice versa) as well as keeping track of incident associated CAP alerts that are updated by the message originator need be established. For this reason the following component have been designed and developed.

- **EDXL-CAP Alert Plugin**: This is a Client-Side Plugin that provides a User Interface for sending, receiving and managing CAP Alerts. It acts in cooperation with the main Incident Management Plugin and its primary job is to disseminate Incident Information via CAP Alerts/Updates. It also manages the receiving CAP Alerts/Updates in order to create new or update existing incidents. Its user interface contains tables with incoming and outgoing CAP Alerts, views showing CAP Information in detail and wizards for creating and sending CAP Alerts. It is also integrated with Map Control Bundle so it can show the Areas and Locations of incoming CAP Alerts on the Map so the application user can add them directly to any related Incidents.

- **EDXL-CAP Object Model Plugin**: This plugin contains the formal structure of objects as defined by the OASIS EDXL-CAP Standard. The objects have been generated from the provided XSDs into formal structures and then annotated to ensure compatibility with the underlying persistence framework (Hibernate).

- **EDXL-DE Object Model Plugin**: This plugin contains the formal structure of objects as defined by the OASIS EDXL-DE Standard. The objects have been generated from the provided XSDs into formal structures and then annotated to ensure compatibility with the underlying persistence framework (Hibernate).

- **ARS Access Plugin**: This plugin allows the client application to query the ARS for available endpoints and consequently queue a message for the ARS Message Agent to send.

- **ARS Message Agent**: The agent acts as a go-between the ARS and an ENGAGE Server instance. Its job is to register with the ARS, sends messages to the ARS and receive messages from the ARS destined

- **EDXL-CAP EJB**: Manages the interaction between client application and the database server by providing functions for selecting and updating CAP Alert Objects from/to DB. It also provides the business logic that is related with the application.

- **EDXL-DE EJB**: This programming module acts as the go-between the client application and the underlying database. It provides the business logic for database persistence operations.

7 Integration with the ARS

The diagram depicted in the following Figure depicts how an EDXL-CAP Alert is propagated between two ENGAGE instances. The sequence commences when the operator at ENGAGE #1 creates an incident and decides to disseminate the incident as an alert to another ENGAGE instance.

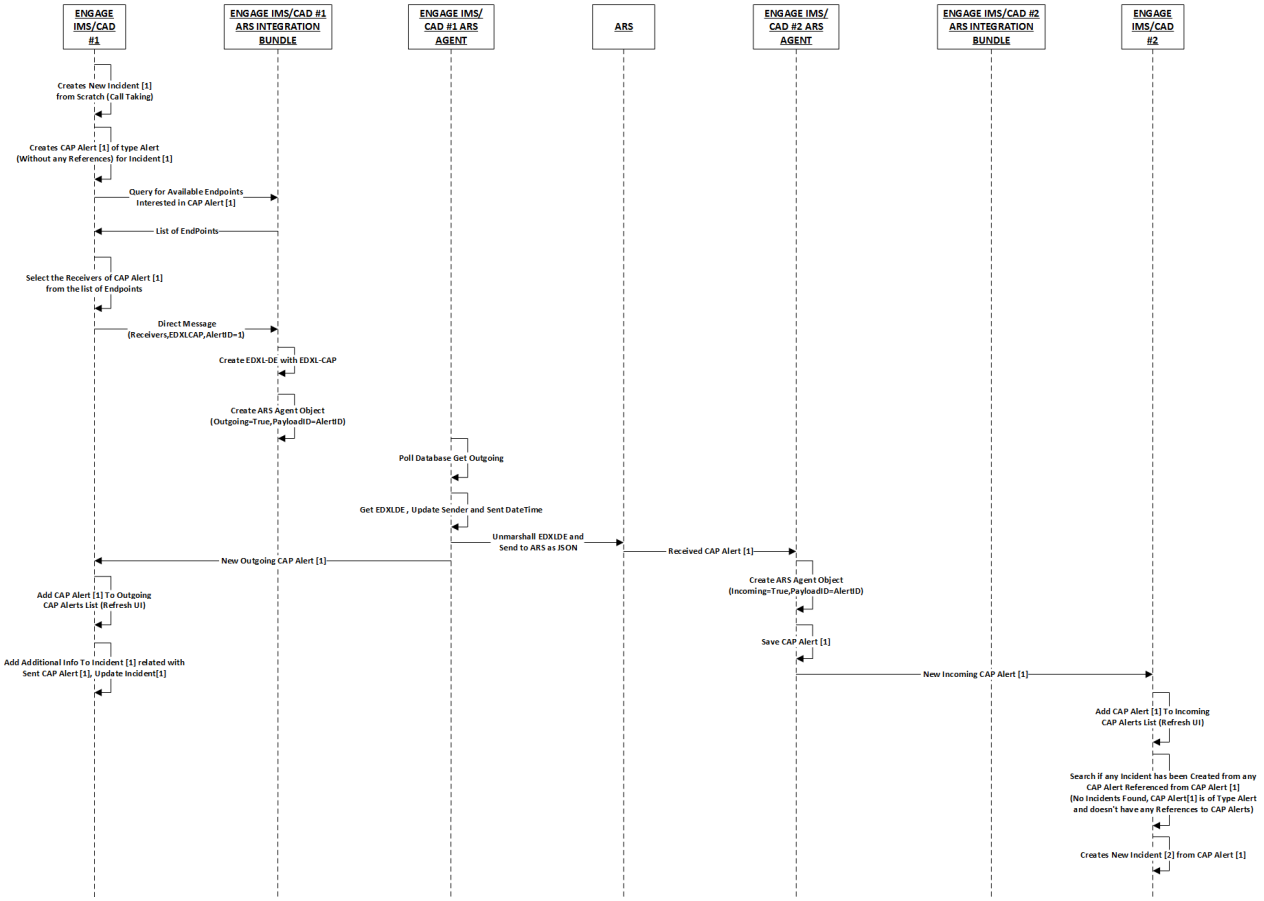


Figure 10: Flow of EDXL messages and incidents between C2 applications

The diagram illustrated in Figure 11 on the other hand illustrates the flow when the originator of the first EDXL-CAP decides to disseminate new updated information about their incident.

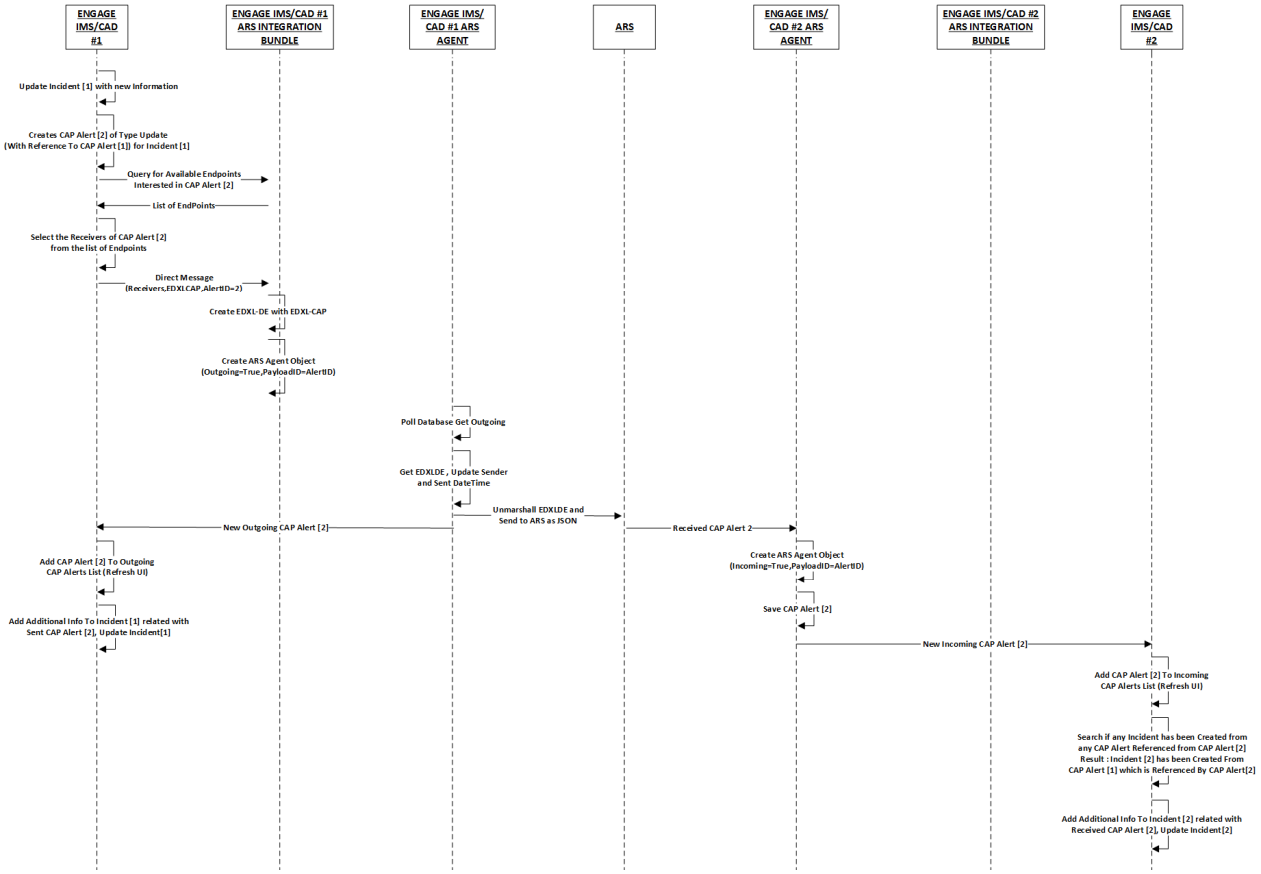


Figure 11: Dissemination of an Alert updates

In each C2 installation there is an ARS Agent service which communicates directly with the ARS Web/App Server. The ARS Agent Service has the following characteristics:

- Register with the ARS Web/App Server (publishing the existence of a C2 instance)
- Identify the specific C2 installation
- Publish outgoing messages to the ARS Web/App Server that have been selected by the specific C2 Operator
- Listen for incoming messages from the ARS Message Bus destined for the specific C2 instance.
    - Translate incoming messages so that they correspond to internal mechanisms and identification (e.g. match external incidents with local incidents).

8 Conclusions and Future Work

The multi-agency C2 prototype and the associated ARS system is a work in progress prototype designed to bring focus on the need for interoperable systems and data standards for operational-level early warning systems and first responder systems such as CAD software. This work has been based on a reliable and flexible messaging platform that captures comprehensive alert and incident data and routes it to the appropriate parties.

During this prototyping effort we were able to successfully create a software router based on open standards that is able to route, deliver and expose messages based on payload content type, sender role, geographical areas of interest and even message keyword information.

Our future objectives is to extend the ARS system capabilities in order to support additional emergency management messages such as resource management, unit tasking and situational reporting.

9 References

[1] European Flood Awareness System - https://www.efas.eu/

[2] EDXL-DE - https://docs.oasis-open.org/emergency/edxl-de/v1.0/EDXL-DE_Spec_v1.0.pdf

[3] EDXL-CAP - OASIS Standard - http://docs.oasis-open.org/emergency/cap/v1.2/CAP-v1.2-os.pdf

[4] Emergency Management Technical Committee of the OASIS - https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=emergency

[5] OSGi – Open Services gateway Initiative - https://www.osgi.org/

[6] Java Enterprise Edition – http://www.oracle.com/technetwork/java/javaee/overview/index.html

[7] Hibernate Object Relational Mapping - http://hibernate.org/orm/

[8] REST - http://www.service-architecture.com/articles/web-services/representational_state_transfer_rest.html