Deliverable reference:	Date:	Responsible partner:
D04. 3	December 22nd, 2014	CNet

Bridging Resources and Agencies in Large-Scale Emergency Management

BRIDGE is a collaborative project co-funded by the European Commission within the Seventh Framework Programme (FP7-SEC-2010-1)
SEC-2010.4.2-1: Interoperability of data, systems, tools and equipment
Grant Agreement No.: 261817
Duration: 1 April 2011 – 31 March 2015

www.sec-bridge.eu

Title:

Information and Deployment View on the BRIDGE System Architecture

Editor(s):
Matts Ahlsén, Peeter Kool (CNet)

Approved by:
Evangelos Vlachogiannis

Classification:
Public

Abstract / Executive summary:

The BRIDGE systems architecture is intended to provide a framework for description and documentation of the overall BRIDGE system functionality and the ways in which the platform can be deployed for use under different circumstances and for different purposes. The architecture is structured and described following three architectural views: a functional view describing the BRIDGE middleware components and services, an information view and a deployment view. This document (D04.3) describes the latter two views, whereas *the Functional View* is described in the deliverable D04.2. *The Information View* of the BRIDGE architecture describes the data structures, message and event formats employed or recommended for use, in the BRIDGE system. The Information View also makes reference to standards and guidelines related to these models and structures. The specific concerns for the management of multiple media contents in the BRIDGE architecture are described in the accompanying deliverable D04.4. *The Deployment View* of the BRIDGE architecture describes the run-time components and their configurations as a system of systems. Adopting a system of systems view means that there are several alternative configurations of the components and tools of the BRIDGE project.

Document URL:
http://www.sec-bridge.eu/deliverables/...

ISBN number:
-

Table of Contents

Version History

Version[1]	Description	Date	Who
1	Initial ToC and objectives	2013-10-17	Matts Ahlsén, Peeter Kool (CNet)
2	First draft	2013-11-23	Matts Ahlsén, Peeter Kool (CNet)
3	Reworked, new structure	2014-05-10	Matts Ahlsén, Peeter Kool (CNet)
4	Description of deployment options	2014-05-28	Matts Ahlsén, Peeter Kool (CNet)
5	Revised structure, for input	2014-06-02	Matts Ahlsén, Peeter Kool (CNet)
6	Mesh network deployment issues	2014-06-19	Christian Raffelsberger (UNIKLU)
7	Master Table, Master Mobile server deployment and data management. WISE Connectivity deployment. Information Intelligence data management. LinkSmart architecture deployment. eTriage data management.	2014-06-23	Aslak Wegner Eide (SINTEF), Andreas A.C. Carlsson (SAAB), Daniela Pool (UNIKLU) Erion Elmasllari, Mark Vinkovits, Svetlana Matiouk (FIT)
8	Deployment descriptions	2014-08-15	Matts Ahlsén, Peeter Kool (CNet)
9	Version for internal review	2014-10-30	Matts Ahlsén, Peeter Kool (CNet)
10	Revision after internal review	2014-12-18	Matts Ahlsén, Jan H. Skjetne, Bernard van Veelen
11	Version for submission	2014-12-22	Matts Ahlsén, Peeter Kool (CNet)

[1] Note that the version number and description should correspond to the same information in the eRoom version control, thus version numbers are *integers*. See below for more information.

Contributing partners

	CNet Danderyd Sweden	**Matts Ahlsén** matts.ahlsen@cnet.se **Peeter Kool** Peeter Kool@cnet.se
	UNIKLU Alpen-Adria-Universität Klagenfurt Klagenfurt, Austria	**Hermann Hellwagner** hermann.hellwagner@aau.at **Daniela Pohl** daniela.pohl@aau.at **Christian Raffelsberger** christian.raffelsberger@aau.at
	FIT Fraunhofer-Institut für Angewandte Informationstechnik Sankt Augustin, Germany	**Andreas Zimmermann** andreas.zimmermann@fit.fraunhofer.de **Erion Elmasllari** erion.elmasllari@fit.fraunhofer.de **Svetlana Matiouk** svetlana.matiouk@fit.fraunhofer.de **Mark Vinkovits** Mark.vinkovits@fit.fraunhofer.de
	SAAB Group Sweden	**Andreas Carlsson** andreas.ac.carlsson@saabgroup.com
	SINTEF Oslo Norway	**Aslak Wegner Eide** Aslak.Eide@sintef.no **Antoine Pultier** Jan.H.Skjetne@sintef.no
	Thales R&T Nederland Delft The Netherlands	**Bernard van Veelen** bernard.vanveelen@d-cis.nl

List of Figures

List of Abbreviations

ASA	Advanced Situation Awareness
AQ	Architectural Qualities
CC	Concept Case
DAC	Device Application Catalogue
EDXL	Emergency Data eXchange Language
EDXL-DE	EDXL Distribution Element
EDXL-SitReP	EDXL Situation Reporting
EDXL-RM	EDXL Resource Messaging
EDXL-TEP	EDX Tracking Emergency Patients
GML	Geography Mark-up Language
HLS	HTTP Live Streaming
IOPE	Input, Output, Precondition, and Effects
IoT	Internet of Things
IPv6	Internet Protocol v6
OASIS	Organization for the Advancement of Structured Information Standards
QoS	Quality of Service
RTMP	Real-Time Messaging Protocol
S2D2S	Shared Secure Distributed Data Space
SDK	Software Development Kit
SOA	Service Oriented Architecture
SWARM	Situation aWAre Resource Management
6loWPAN	IPv6 over Low power Wireless Personal Area Networks
UAV	Unmanned Aerial Vehicle
UICDS	Unified Incident Command and Decision Support
UPnP	Universal Plug and Play

Executive Summary

The BRIDGE architecture is documented in the following views,

- The Functional View
- The Information View
- The Deployment View

This document (D04.3) describes the latter two views, whereas the Functional View is described in the deliverable D04.2.

The Information View of the BRIDGE architecture describes the data structures, message and event formats employed or recommended for use, in the BRIDGE system. The Information View also makes reference to standards and guidelines related to these models and structures. The specific concerns for the management of multiple media contents in the BRIDGE architecture are described in the accompanying deliverable D04.4.

The Deployment View of the BRIDGE architecture describes the run-time components and their configurations as a system of systems. Adopting a system of systems view means that there are several alternative configurations of the components and tools of the BRIDGE project. A central part of the BRIDGE design process was the development of a set of Concept Cases – these are applications and technologies that are used to drive the requirements process while in a later stage also being used as validation instruments for the implemented architectural design solutions. The Concept Cases include end-user centred applications, such as e-Triaging, as well as more general technical solutions such as the Mesh network.

1 Introduction

This document describes the information and deployment views of the BRIDGE system. These views and together with the functional view, form an overall description of the BRIDGE system architecture.

1.1 The BRIDGE project

The BRIDGE project has developed a platform for Emergency Management Systems and related applications. The goal of BRIDGE is to increase safety of citizens by developing technical and organisational solutions that significantly improve crisis and emergency management. The BRIDGE platform provides technical support for multi-agency collaboration in large-scale emergency relief efforts. The key to this is to ensure interoperability, harmonization and cooperation among stakeholders on the technical and organisational level. The developed platform should facilitate cross-border and cross-agency collaboration and support the creation of a common operational picture for the responders involved in an incident.

The BRIDGE system architecture is designed to support flexibility in the configuration of different combinations of BRIDGE middleware services with first responder devices and control applications.

1.2 Context and Scope of this deliverable

1.2.1 The functional view of the BRIDGE architecture

The functional view describes the BRIDGE platform in terms of its services and their structuring in a service-oriented architecture. The BRIDGE middleware services are enclosed by a physical communications layer at the bottom and an application layer at the top of the diagram respectively.

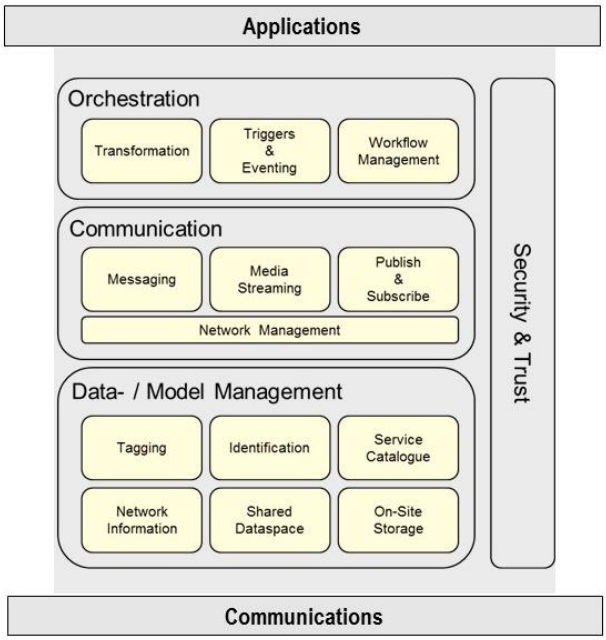


Figure 1: BRIDGE functional view - services

The communications layer realizes several network connection technologies like ZigBee, Bluetooth or WLAN. The application layer contains user applications which could comprise functions like workflow management, user interface, custom logic and configuration details.

The BRIDGE platform offers a large collection of reusable core software components for developers to develop run-time applications. Based on these software components, the BRIDGE middleware services provide programming abstraction and functionality for developers. The middleware services are logically clustered in four groups of services:

- Orchestration
- Communication
- Data- and Model Management
- Security & Trust

The Middleware Services represent globally available functionality shared by all BRIDGE applications, and possibly external systems/actors. The internal structure of each component is determined by a design derived from the related set of requirements and hence determined by specific project work packages.

A set of Architectural Qualities (AQs) has been analysed and described as part of the projects Domain Analysis task in Work package 2. AQs are non-functional properties of a system in use, as perceived by users of applications or of tools and mechanisms for design or configuration. A full list of AQs can be found in deliverable D04.2.

1.2.2 Information and Deployment View

The Deployment View of the BRIDGE architecture describes the run-time components and their configurations as a system of systems. Adopting a system of systems view means that there are several alternative configurations of the components and tools of the BRIDGE project.

The Information View of the BRIDGE architecture describes the data structures, message and event formats employed or recommended for use, in the BRIDGE system. The Information View also makes reference to standards and guidelines related to these models and structures.

We refer to deliverable D04.2 for a more detailed description of the architecture definition process.

2 Deployment View

2.1 BRIDGE System of systems

A Systems of systems approach

The vision for the BRIDGE project is to provide an infrastructure for a BRIDGE system of systems, i.e., a set of loosely coupled sub systems and services, with a minimum of global constraints and structures, but with a high degree of interoperability. This means that the middleware must be able to manage a very dynamic environment where services come and go in the BRIDGE network.

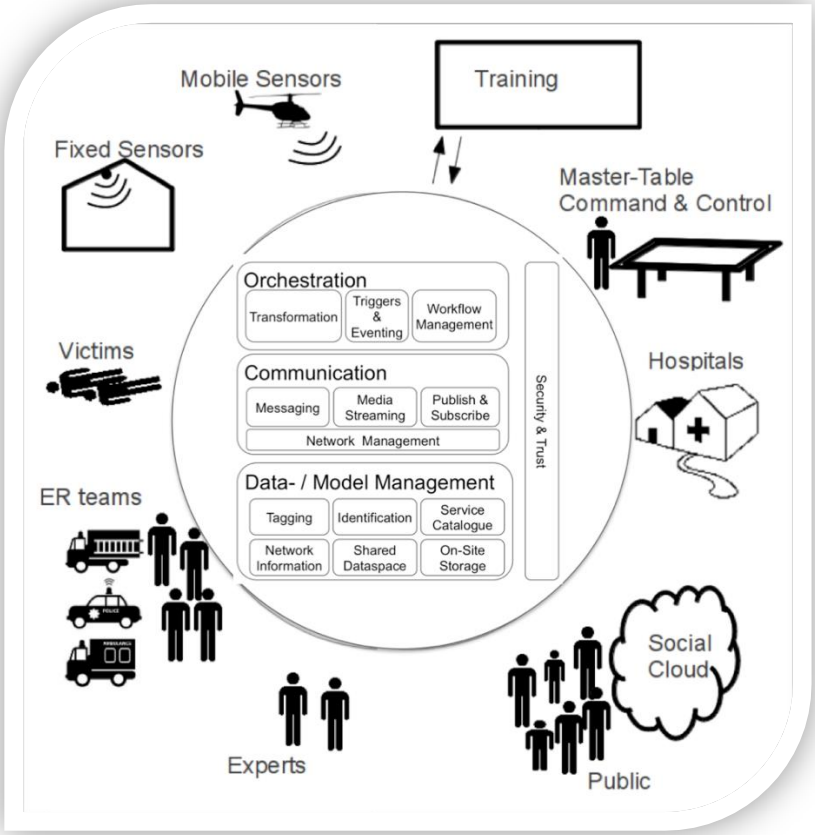


Figure 2: BRIDGE system of systems context

As BRIDGE is designed as a system of systems there is no specific BRIDGE system instance, rather the assembly of sub-systems and services in a BRIDGE configuration depends on the deployment context and scale, in real situations or in training.

The BRIDGE system of system approach to the architecture has several consequences. At run-time BRIDGE is composed of set of loosely coupled sub systems and services, with a minimum of global constraints and structures. At the same time there must be an openness for inclusion of new resources (services, devices), implying compliance to standards.

There are also a number of design time considerations for developers of BRIDGE client applications. A starting point is never to assume continuous resource and system availability,

hence the BRIDGE platform must be assumed to operate under potentially disruptive conditions considering, e.g.,

- Infrastructure failure, such as network component failure.
- Services and devices may become unavailable or operate with varying quality of service
- Congestion of communication channels and information/data overload

As a complement to the technical platform and the middleware, a set of Architectural Qualities (c.f., D04.2) were compiled to serve as an guidelines for the design of the BRIDGE system of systems from different quality perspectives.

These were then made concrete in terms a set of proof of concepts applications, referred to as BRIDGE Concepts Cases, representing emergency management applications and users (c.f. the perimeter of Figure 2). The Concept Cases were developed to show and capture (emergency) user requirements while at the same time placing technical requirements and constraints on the middleware. As Concept Cases were developed and validated, parts of their functionality was factored out and integrated with the middleware services (c.f. centre of Figure 2) thus becoming available to other existing as well as future applications.

The subsequent sections describe the deployment options for some of the different Concept Cases and middleware subsystems with respect to the BRIDGE platform.

2.2 Overview and control – the Master

The Master system is a map-based platform for synthesizing and presenting relevant information during emergency response. In this sense the Master functions as a window onto the BRIDGE system of systems. It is a means for commanding personnel to access, filter, visualize, and share various kinds of information in an efficient manner. The information is presented in a common operational picture that takes the form of an interactive 2D map. An overview screenshot of this interactive map is given in Figure 3.

Figure 3: Example screenshot of the Master system

The map has a number of informational overlays that can be shown or hidden, depending on which information the given user needs. Users can add, change and remove elements from some

of the overlays, and also add and edit additional information about the elements that have been added. The changes a user makes to the operational picture are instantly reflected on all running instances of the Master tool, enabling all involved personnel to share a single identical overview of the situation. Table 1 describes the overlays that are available in the current version of the tool, whether the overlays are editable, and which systems that provide the raw data. It should be emphasized that the number of available information overlays is expected to increase in the near future, as the Master tool will also receive data from the Advanced Situation Awareness Concept Case, which will provide sensor data, images, and video feeds from unmanned aerial vehicles (UAVs). Besides the common operational picture, the Master also provides a variety of other functionality. Most importantly, this comprises resource management functionalities that allows users to allocate and mobilize resources (i.e. personnel and vehicles) to specific tasks and locations, and chat communication with other Master users.

Overlay	Description	Editable	Source
Incidents	Overlay showing the location of the incidents that has occurred, as well as their type and description.	Yes	Master
Victims	Overlay showing the location of victims that has been tagged with eTriage bracelets and victims that have sent out a cry for help using the HelpBeacon app.	No	eTriage & HelpBeacon
Response	Overlay showing the location of response objects (e.g. command post, resource meeting place, inner and outer zones), as well as detailed information about these objects.	Yes	Master
Risks	Overlay showing the location of risks related to the incident, as well as detailed information about these risks.	Yes	Master
Resources	Overlay showing the real-time location of response personnel and vehicles, as well as detailed information about their type, status, assignment, and ownership.	No	SWARM[1]
Social media	Overlay showing geolocalized images, tweets, and videos from social media that is relevant to the incident.	No	Information aggregator
Object info	Overlay showing the availability of object plans and 3D models by associating these elements to locations in the map.	No	3D visualization

Table 1: Informational overlays in the Master

The Master system is intentionally designed to be used by all organizations that take part in emergency response efforts, on all levels of command within these organizations, and during any type of event, ranging from small day-to-day operations, to large-scale emergencies. To support users on the different levels in the command chains of these organizations (i.e. strategic, tactical, operational levels), and during various types of events, the tool has been tailored to run

[1] The BRIDGE Concept Case SWARM (Situation aWAre Resource Management) provides a mobile system for tracking the location and status of resources in real time, and for notifying resources of the assignments they have been allocated to perform (see deliverable D06.2).

on a number of different devices, including normal desktop PCs, collaborative multi-touch tables, and tablets (see Figure 4).

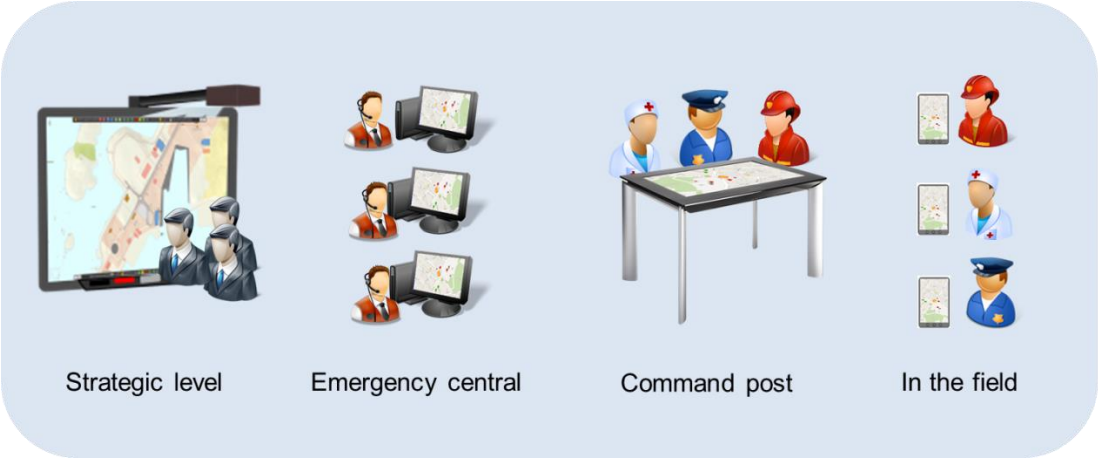


Figure 4: Master tool

The users on the strategic and tactical levels will typically run the tool on a desktop PC, preferably beamed to the wall using a projector. On the operational level, the tool will typically run on tablets that users can carry around or dock to the dashboard of a vehicle. In large-scale events, the tool will also be run on a collaborative multi-touch table placed in a local command post or in a mobile command centre vehicle. In order to receive and send data, the Master is dependent on access to LinkSmart and S2D2S. More detailed descriptions of the Master system is given in deliverables D06.2 and D06.3.

2.3 BRIDGE Services

2.3.1 Service Catalogue

The role of the service catalogue is to facilitate the deployment of different middleware services to be consumed via the BRIDGE platform.

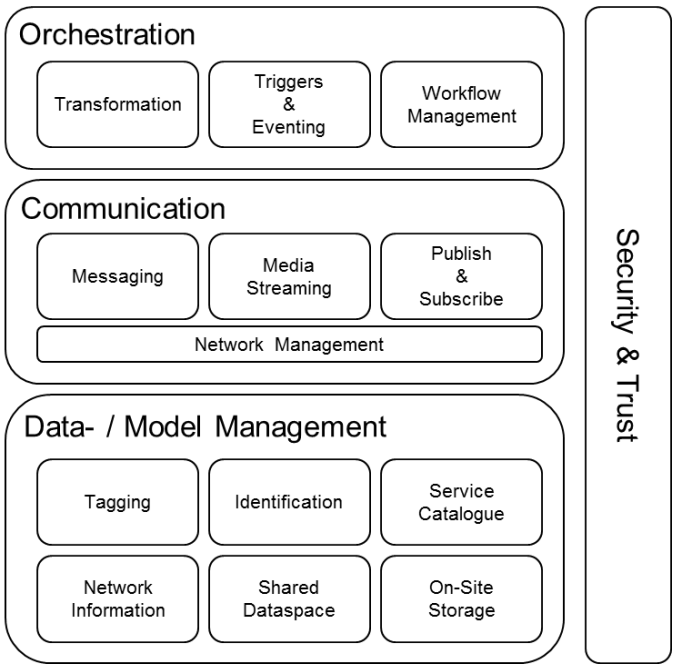


Figure 5: Structuring of the BRIDGE Middleware Services, including the service catalogue.

The service catalogue is deployed as a (cloud) service by itself, providing an entry point to all different kinds of BRIDGE services (Figure 5). Services can be various device services (such as environment sensors, triage tags), software agent services or human expert capabilities exposed as services. The service catalogue is a mechanism for managing the dynamic behaviour of resources in the BRIDGE network, e.g., new devices need to be automatically discovered and registered at the service proxy registry.

The service catalogue builds on the LinkSmart discovery mechanisms for IoT (Internet of Things) device and service (initially based on web services and the UPnP discovery protocol) extended with BRIDGE specific service attributes. The service catalogue also provides for Service Resolution, i.e., the functionality needed by BRIDGE clients in order to discover and get access to services.

To facilitate resolution, the catalogue also includes service descriptions, that may encode service semantics both formally (IOPEs[2]) and in natural language.

2.3.2 Quality of Service

The service descriptions may be extended with quality properties that are used to specify quality of service (QoS). The management of QoS in BRIDGE has been extensively analysed and elaborated by Work package 7 and is detailed in the deliverable D07.4[3] including an information meta-model and an information flow design for QoS. The design principles for BRIDGE the

[2] Input, Output, Preconditions and Effects

[3] D07.4 Scenario Bound Organisation, Coordination and Information Meta-Models

QoS are to a large extent based on requirements for the orchestration of services in the context of workflow generation and management, this is further elaborated in deliverable D07.3[4].

A BRIDGE Quality Descriptions defines the syntax and semantics for the representation of QoS properties (e.g., bandwidth). The descriptions can express a number of different qualities as well as possible dependencies between such qualities. The QoS framework further specifies a set of quality types and also proposes a set of default qualities.

The QoS descriptions for services included in the Service Catalogue are registered in a BRIDGE QoS Repository. This repository allows clients to register QoS descriptions using the specific BRIDGE QoS format and also provides for querying the repository for previously registered descriptions. The QoS descriptions are thus an integral part of the service semantics in the BRIDGE system.

We refer to deliverable D07.4 for details and usage of the overall BRIDGE QoS framework.

2.3.3 Storage options

The BRIDGE middleware provides a generic storage facility based on the S2D2S[5], a combined publish and subscribe eventing and storage service. As such it provides a persistent data space for sharing and distribution among multiple clients. This storage service is built on the AgentScape system (described in detail in deliverable D04.2) and exploits the AgentScape blackboard service for storing data items in topics, and provides a publish/subscribe mechanism). It also makes use of the AgentScape servlet service to provide access to the service via HTTP(s). Access to this service is done via LinkSmart or directly via remote processes via JSON-RPC. The machine that runs the S2D2S service also runs a LinkSmart proxy which communicates with LinkSmart on one side (using web services) and JSON-RPC on the other side.

Another storage access option in for BRIDGE middleware clients is to use On-Site Storage, i.e., the possibility of sharing locally stored data (e.g., disaster area imagery), over the BRIDGE network on demand.

Access to the On-Site Storage is usually achieved via reference links provided by the party producing or hosting the data. In this way actual communication over the network can be minimized and thus save bandwidth. The On-Site service uses a specific BRIDGE Link format for this.

2.4 BRIDGE Middleware subsystems

2.4.1 *LinkSmart*

The BRIDGE platform uses the LinkSmart[6] system as its core middleware component. The software architecture described here is an abstract representation of the software part of the LinkSmart middleware. The architecture is a partitioning scheme, describing components and their interaction with each other. Figure 6 shows the concrete deployment of LinkSmart managers on the specific network components according the LinkSmart terminology. Each Native Device is connected through a Network Manager. A Gateway further hosts a couple of

[4] D07.03 Workflow Configuration Concepts and Methods

[5] Secure Shared Distributed Data Space

[6] www.linksmart.eu

proxies for closed platform devices (e.g. commercial off the shelf Bluetooth and ZigBee devices).

The gateway must run a Network Manager that registers all services belonging to the devices in the LinkSmart network. Moreover, on the gateway a Device Application Catalogue (DAC) can keep track of devices available in the LinkSmart network.

The LinkSmart Application runs on a PC as a dedicated application (i.e. a centralized architecture). The application can access specific services through a Network Manager if it knows in advance which services it wants to use. Alternatively, the application can browse the devices on the DAC first, based on specific criteria and access their services.

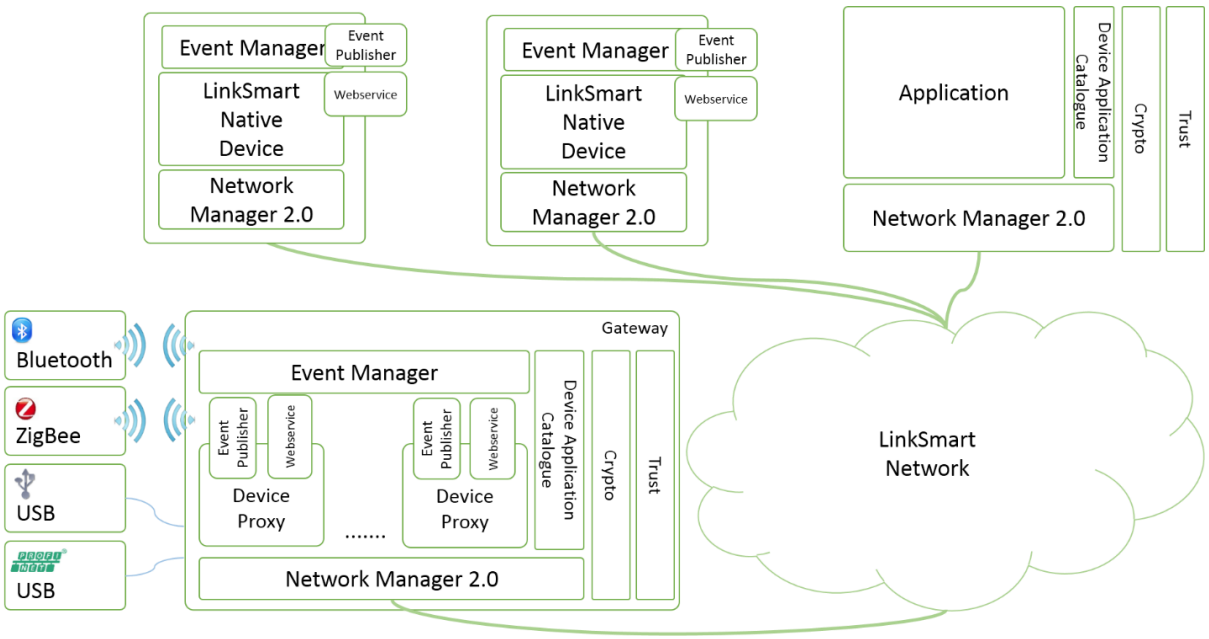


Figure 6: LinkSmart Configuration Example Deployment

The EventManager handles the publishing and the subscribing of events overarching specific deployment infrastructure. Applications can subscribe to an EventManager for a topic they are interested in or publish events, for example events containing sensor measurements. An event consists of a topic and key-value pairs.

In Figure 7, we see an example of a LinkSmart device network. LinkSmart distinguishes between powerful devices that are capable of hosting the LinkSmart middleware and smaller devices that are too constrained or closed to run the middleware. For the latter, proxies are used and once proxies are in place, all communication is based on the IP protocol.

The figure below illustrates these two device types. On the right, there are devices which can host the LinkSmart middleware and which are able to establish communication with services on the platform. On the left, devices are depicted which cannot operate the LinkSmart middleware, either because they have resource constraints or proprietary interfaces. For these devices proxies are created on a Business Area Network or a Personal Area Network node (in this case a mobile phone). For a service the communication with a proxy is not any different from communication with a LinkSmart enabled device.

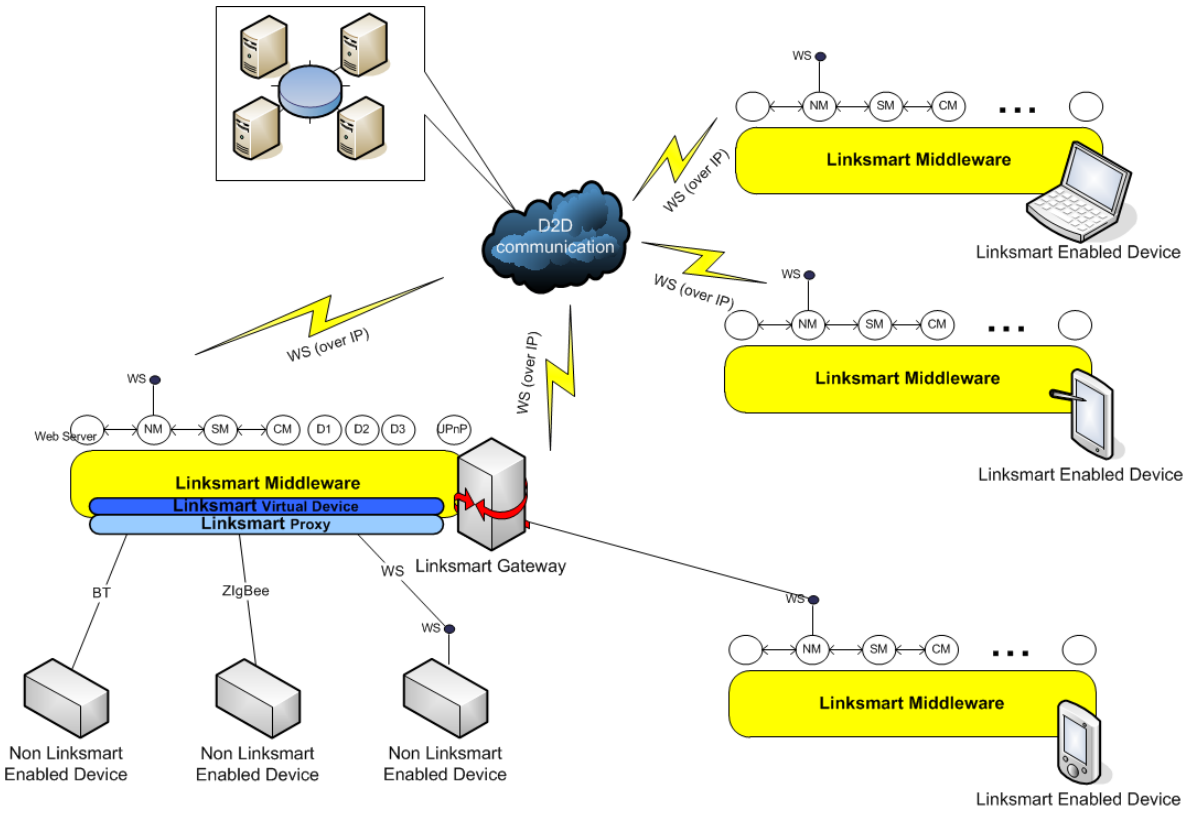


Figure 7: LinkSmart Example Device Network

The BRIDGE Mesh, which is the infrastructure that hosts LinkSmart at the disaster site, is a combination of LinkSmart enabled and constrained devices. The LinkSmart enables BRIDGE Mesh devices can act as gateways, as in the case of eTriage, and host proxies that enable the integration of these devices into the middleware.

LinkSmart proxies cannot only be used for the integration of devices with proprietary interfaces, but also for the integration of software services that use different protocols. This is also the approach that has been applied for the integration of the S2D2S, the Resource Manager and the Master Table. In these cases the machine hosting the specific application also acts like a LinkSmart Gateway hosting a dedicated proxy, registering the specific application and forwarding communication. This enables these services to discover and communicate with other applications without considering the physical deployment or the form of the connection between them.

2.4.2 Master Mobile Server

The Master system is available in several different versions, targeting different devices such as desktop PCs, multi-touch tables, and mobile units. The desktop PC and multi-touch table versions connect directly to Linksmart and S2D2S by running the Linksmart client locally on the device. In a mobile device however, running the client locally is not always feasible due to the wide variety of operating systems and hardware specifications available. This is particularly a challenge in terms of making the mobile Master application device-independent.

To overcome the abovementioned challenge, we developed a server component that acts as a bridge between LinkSmart and the mobile Master version (see Figure 8). Clients using the mobile Master version (which is built as a cross-platform HTML5 application), will connect to this server component, instead of connecting directly to Linksmart via a local client. As a

consequence, one will thus be able to use the mobile Master version on any device that supports HTML5, without having to install or set up the LinkSmart client locally on the device.

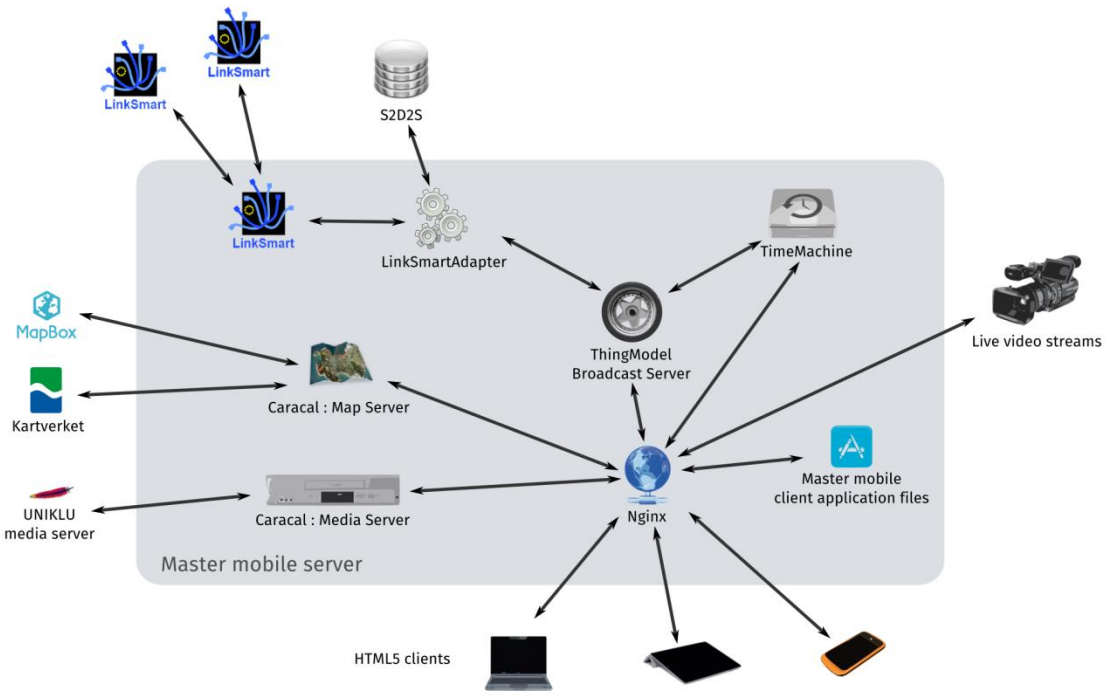


Figure 8: Master Mobile Server

The components of the Mobile Master server are described in detail in the list below:

- The main component of the mobile Master server is the ThingModel, which is a lightweight framework for data-synchronization across multiple components and devices. The developer API is easy to use, providing a simple syntax, without requiring compilation (the models are shared in runtime). The network synchronization is efficient, using small binary diff messages over WebSocket connections.

- The LinkSmartAdapter connects LinkSmart and S2D2S to the ThingModel. It translates in both directions EDXL documents and ThingModel objects.

- The TimeMachine is a ThingModel component for recording the model in real-time. Clients can fetch snapshots from the TimeMachine to display earlier states of the situation when necessary.

- The map server can host maps using WGS84/WebMercator map tiles, and operate as a reverse proxy for online map providers. In order to generate maps, we make use of TileMill[7], which is an Open-Source software for generation of map tiles.

[7] See https://www.mapbox.com/tilemill/ for more information about TileMill.

- The media server is a component for hosting any kind of document, such as pictures or recorded videos. It can also operate as a reverse proxy for fetching files from other servers. Online image resizing is supported.

- Live video streaming is supported via the nginx-rtmp-module[8] using the RTMP[9] or HLS[10] protocols. Online transcoding is possible with ffmpeg and x264 but not activated by default. One should note that the latency implied by these technologies can range from 10 seconds to 1 minute.

- The Mobile Master Client application files are compiled static files, easily hosted by any HTTP server. Currently these files are hosted by nginx.

- Nginx is the hub, providing access to the different components for the Mobile Master clients. The communication between Nginx and the clients uses HTTP and Websocket, encrypted by OpenSSL. Nginx is the only network point required by the clients. The communication is designed for slow and instable networks (e.g. 3G, low bandwidth Wi-Fi).

Deployment of the Mobile Master Server can be easily achieved using a cloud-based solution. Ready-to-use virtual machines are readily available, and can be installed in one step. In cases where internet is unavailable, one can setup a local server by bringing a preinstalled laptop.

2.4.3 Mesh Network

This section briefly describes the deployment of the BRIDE MESH network that is one of the possible networking platforms for the middleware. Details about the deployment of the MESH[11] network can be found in *D05.1 BRIDGE Network Infrastructure*. Additionally, D05.1 includes information about experiences that were gained during specific deployments of the BRIDGE mesh network during the demonstrations in Flums (cf. D05.1 Appendix F) and Risavika (cf. D05.1 Appendix G).

[8] See http://nginx-rtmp.blogspot.no/ for more information about the Nginx rtmp module.

[9] See http://www.adobe.com/devnet/rtmp.html for more information about RTMP.

[10] See http://tools.ietf.org/html/draft-pantos-http-live-streaming-13 for more information about HLS.

[11] The term Mesh refers to the topology of the resulting networks

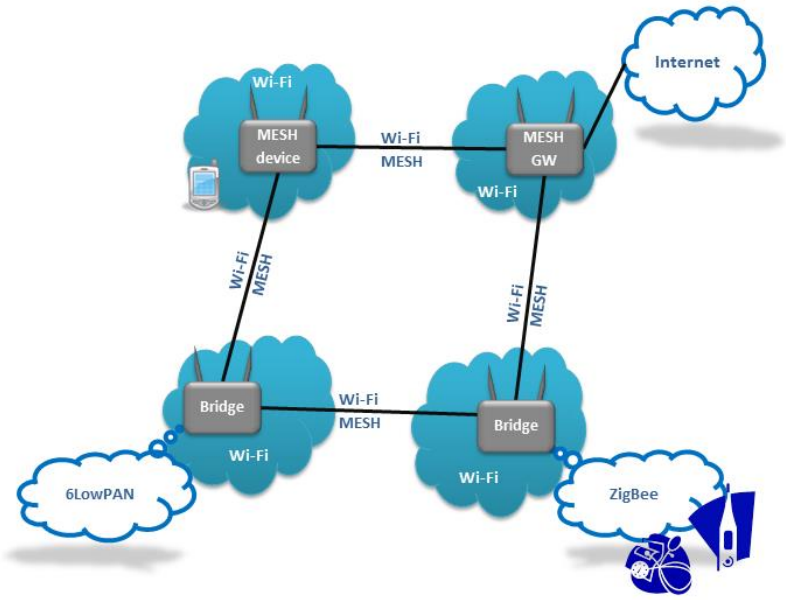


Figure 9: BRIDGE Mesh (from D05.1)

The main goal of the BRIDGE MESH is to create a deployable network infrastructure that interconnects heterogeneous sub networks, both IP-based (e.g. Wi-Fi or 6loWPAN[12]) and non-IP-based (e.g. ZigBee), in the disaster zone where existing infrastructure has been disrupted. Since the interconnection of different wireless technologies requires different communication electronics and antennas, it is not optimal to include all functions into one device. Especially, since not all emergency response operations require the use of all technologies at the same time. Thus, different types of mesh devices with different capabilities should be deployed, depending on the task at hand. For the demonstrations and exercises that were conducted within the BRIDGE project, several different mesh devices have been developed.

It is important to note that the focus within the BRIDGE project concerning networking was not to develop or improve networking hardware but to create a software stack for the BRIDGE MESH system that is practical in terms of implementation, deployment and use. For the MESH prototype only hardware that is mainly intended for home use has been used. However, several device categories have been defined which could be useful in real emergency response operations.

Gateway devices connect the MESH network with external networks such as the Internet. A gateway device usually is deployed close to the Incident Command Post to assure a stable connection to the higher level Emergency Operations Centre or other off-site locations. *Range extension devices* are deployed to increase the coverage of the wireless network. Hence, they have to be deployed throughout the area and are used to cover the distance between the gateway devices and the actual incident scenes. Compared to gateway devices, the hardware requirements for range extension devices are lower, since these devices are only used to forward traffic and do not perform further processing. Thus it is possible to reduce their physical size which fosters their integration into other first responder equipment. For instance, range extension devices could be integrated into door chocks that are used by fire fighters to prevent doors from closing and possibly block exits or jam a fire hose. Similarly, such range extension

[12] IPv6 over Low power Wireless Personal Area Networks

devices could be integrated into fire hose couplings. *Bridges* provide different network interfaces and deal with interconnecting networks, one of the main tasks of the BRIDGE MESH. Bridges have to be carefully placed at the locations where these network interfaces are actually needed. For instance, the eTriage bracelet prototypes only provide a ZigBee interface. Hence, a bridge device that forwards collected triage data to the WiFi-overlay needs to be deployed near the triage area. Finally, there may be situations where it is not feasible to cover all areas with network coverage. So called *delay-tolerant networking devices* offer storage capabilities to buffer messages which cannot be delivered instantly. These devices can be deployed in areas where large amount of information is generated but which are only intermittently connected to the rest of the network.

The BRIDGE MESH network has been successfully deployed in several real world tests including realistic first responder exercises. The experiences that have been gained there also led to the definition of several deployment guidelines. The guidelines include information about which locations are well suited for deployment in general (e.g. deploying routers some meters above the ground greatly improves the covered range) and suggested maximum distances between devices in order to create a stable wireless network. A more complete description and the guidelines itself can be found in D05.1.

2.4.4 Platform Integration: WISE Connectivity

A specific connectivity technology (WISE Connectivity) has been adapted for the integration of external training and simulation environments with the overall BRIDGE platform. This is used by the specific BRIDGE Concept Case FRITS, providing both methodology and technology for emergency training based on real as well as simulated incidents with the BRIDGE platform. The WISE Connectivity service can then be used for monitoring of BRIDGE systems activity, but also for injecting or providing access to simulated activity. This section describes the main building blocks of this connectivity solution.

Figure 10: Connectivity for external (training) systems

The WISE Connectivity Runtime Edition (CoRE) is a software used to execute configurations previously created using WISE Connectivity Designer Edition (CoDE).

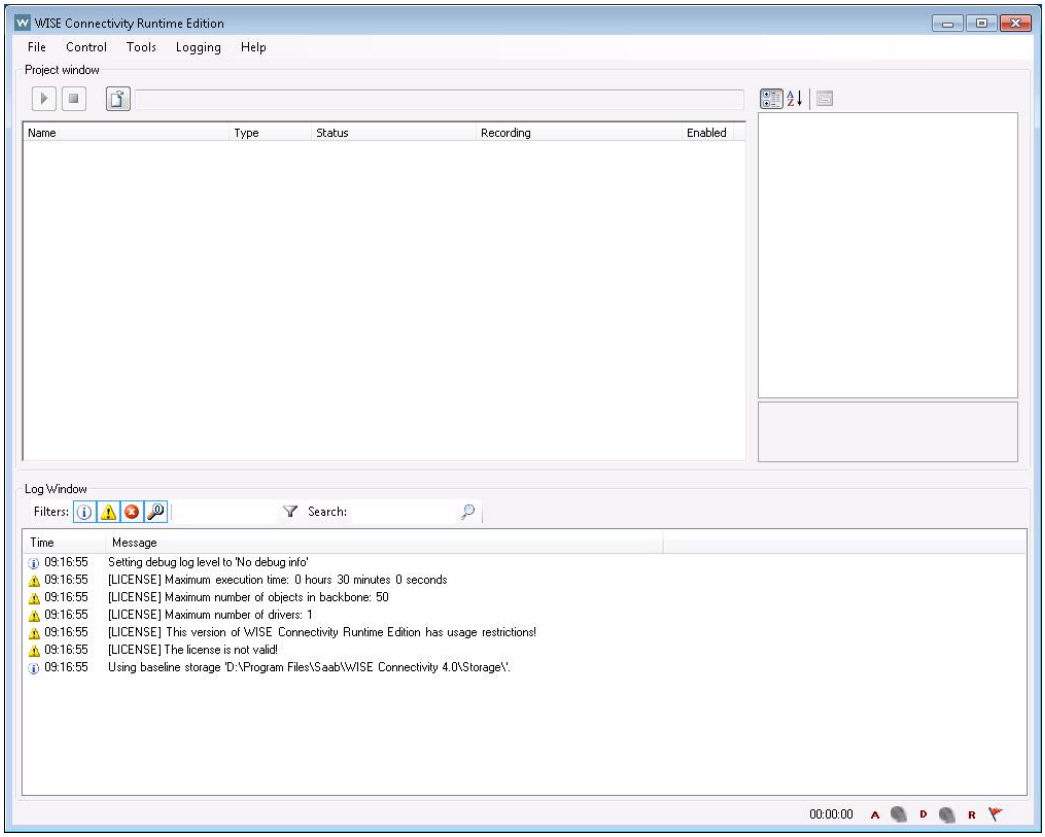


Figure 11: WISE Connectivity Runtime Edition (CoRE)

CoRE exists in two different forms; the standard version with a graphical user interface which is available on Windows platforms and a command-line version (CoRE Service) which is available on both Windows and Linux platforms.

Building blocks

A complete configuration project in WISE involves of the following parts;

- Binary components
  - Drivers
  - Services
  - Transformers
- Configuration components (XML files)
  - Object models
  - Object model transformations
  - Connectivity projects

Binary components

The following image shows how the binary components are related;

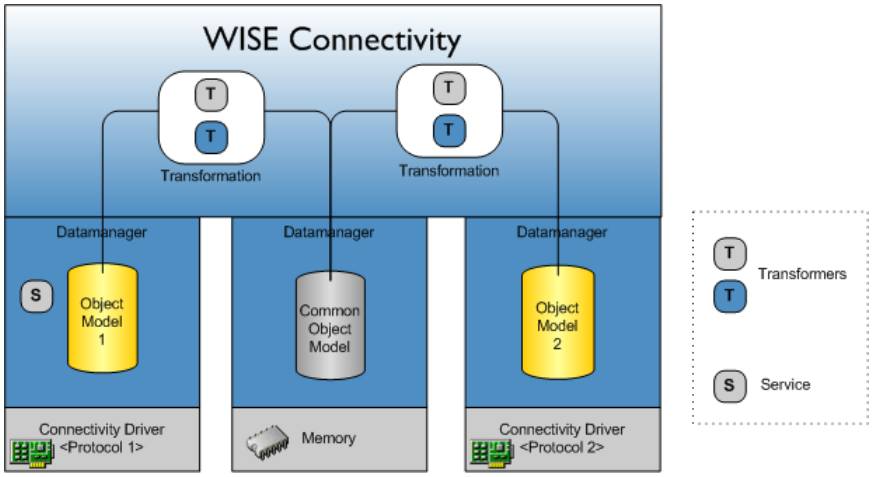


Figure 12: WISE Connectivity

The driver is the binary component responsible for synchronising the database of the connected system with the corresponding database in WISE using a specified protocol.

Services are attached to the WISE database and add new functionality that is not natively supported by the end system.

Some examples of features that a service could implement are;

- Dead-reckoning algorithms
- Aggregation/disaggregation

The last binary component is the transformer component which is used to transform data from one object model to another.

Binary components are developed using the WISE Connectivity SDK.

In order for CoRE to be able to know which binary components are available and the features of these components, each binary is described using an XML-formatted definition file. For instance a transformer component need to describe the input it requires and the output it generates.

Configuration components

A set of configuration files are used to define how the binary components are connected and how information is transformed between the connected systems;

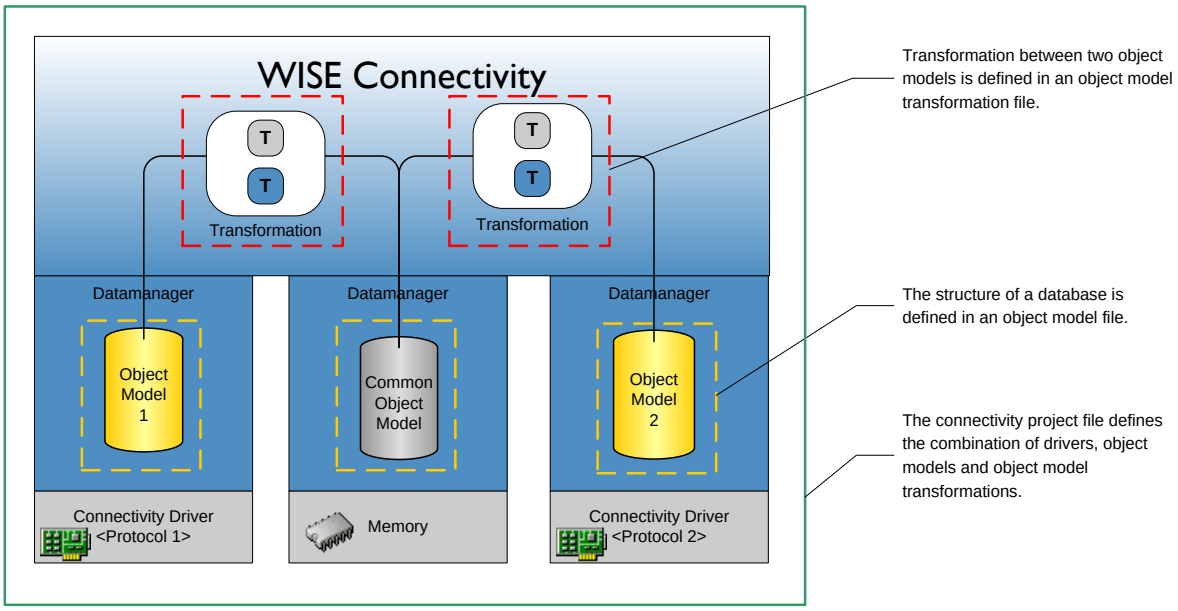


Figure 13: WISE Configuration components

These configuration files are defined using CoDE. The object model file is the foundation of a connectivity project. It defines the information model used by a system and is used to structure the runtime database of the connected system.

An object model transformation file defines how to convert between two object models. One file can define either a one- or two-way transformation.

The containing project file sets up the overall integration environment in terms of which drivers should be loaded, which databases to create and how to map between the databases by referencing the appropriate object model transformation files.

Specific run-time aspects of WISE is described in the Appendix 2: WISE Connectivity run time.

3 Information View

A central guideline for the BRIDGE system of systems design has been the adherence to standards for both information exchange and technical realization of such exchange. This section describes how standard formats for information exchange and information models are employed by main Concept Cases and by the BRIDGE middleware itself.

3.1 Information exchange standard

The BRIDGE project has adopted EDXL as the main standard for information exchange for its Concept Cases. The BRIDGE middleware does not prescribe the use of EDXL, but is open to other formats for client applications, while providing the same quality of service and connectivity[13].

A description of EDXL can be found in deliverable D04.1 which presents a comprehensive overview of systems, R&D projects and standards for emergency management. For completeness we provide a brief characterization of EDXL here.

The Emergency Data Exchange Language (EDXL) has been promoted by the OASIS International Open Standards Consortium[14] with the intent to cover a wide range of emergency data exchange standards (in areas like operations, logistics, planning and finance). The EDXL comprises a set of XML-based message standards should facilitate emergency information sharing between many different types of response organisations, including government entities and other emergency organizations. The different standards subsets are being applied to support interoperability within and across various emergency management systems efforts, among the most prominent are the XchangeCore[15] and BRIDGE.EDXL supports flexibility and extensibility thru its XML serialization vocabulary. BRIDGE uses several of the message subsets, such as

- EDXL Distribution Element (DE)
- EDXL Situation Reporting (SiTReP)
- EDXL Resource Messaging (RM)
- EDXL Tracking emergency Patients (TEP)

Sub sequent sections describe the models and formats used by the BRIDGE Concept Cases and the middleware and their mapping to the standard.

3.2 BRIDGE Link Format

The BRIDGE link format is intended to be used when distributing links to be used for access to data that could be retrieved via the BRIDGE network.

The format is designed using a very simple schema. This allows a client to have multiple link instances to the same target data resource but in possibly different resolutions or formats (depending on target data type).

[13] This depends also on the emergency situation and the available ICT infrastructure

[14] See https://www.oasis-open.org/org/tour for more information about OASIS.

15 Formerly the UICDS (Unified Incident Command and Decision Support), now XchangeCore www.xchangecore.org, middleware for emergency systems interoperability.

```
<xs:schema xmlns:bridge="urn:bridge:link" attributeFormDefault="unqualified"
  elementFormDefault="qualified"
  targetNamespace="urn:bridge:link" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Links">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="Link">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Url" type="xs:string" />
              <xs:element name="Size" type="xs:unsignedInt" />
              <xs:element name="Mimetype" type="xs:string" />
            </xs:sequence>
            <xs:attribute name="description" type="xs:string" use="optional" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figure 14: BRIDGELink schema

One Link consists of a *Url* which can be used to retrieve the data, *Size* in bytes of the data, *Mimetype* of the target data and an optional description attribute.

```
<bridge:Links xmlns:bridge="urn:bridge:link">
  <bridge:Link description="Full resulotion">
    <bridge:Url>http://127.0.0.1:8082/GRANDTunnel/0/122388338/0/x033282t7</bridge:Url>
    <bridge:Size>58937392</bridge:Size>
    <bridge:Mimetype>image/png</bridge:Mimetype>
  </bridge:Link>
  <bridge:Link description="Medium resulotion">
    <bridge:Url>http://127.0.0.1:8082/GRANDTunnel/0/122388338/0/x033232t7</bridge:Url>
    <bridge:Size>50883</bridge:Size>
    <bridge:Mimetype>image/png</bridge:Mimetype>
  </bridge:Link>
  <bridge:Link description="Small resolution">
    <bridge:Url>http://127.0.0.1:8082/GRANDTunnel/0/122388338/0/x033232t7</bridge:Url>
    <bridge:Size>3322</bridge:Size>
    <bridge:Mimetype>image/jpg</bridge:Mimetype>
  </bridge:Link>
</bridge:Links>
```

Figure 15: BRIDGELink instance

The On-site storage service (see D04.2) uses this link format, as is the Master when retrieving data from the BRIDGE Concept Case ASA (see section 3.6). However it can generally be used for all resources that can be retrieved using HTTP.

3.3 Overview and control: The Master

The Master system is a map-based platform that provides commanders and decision-makers with a common operational picture during emergency response. Besides the common operational picture, the Master also provides a variety of other functionality. Most importantly, this comprises resource management functionality that allows users to allocate and mobilize resources (i.e. personnel and vehicles) to specific tasks and locations, and chat-communication with other Master users.

In this section we describe the information models, formats and standards that the Master uses for outgoing data (i.e. data that is sent from the Master to other Master instances). Similar descriptions for incoming data (i.e. data that the Master receives from other BRIDGE components) will be given in the sections for the respective concepts that produce this data. The table below describes the various message types that are managed by the Master system and the respective formats and standards that are used to represent this data.

Data type	Description	Format
Incident Object Messages	These are messages describing geolocalized elements related to the incident itself, the response operation, or risks in the area. The message contains the element type, a description of the element, and the location of the element.	**EDXL-SitRep**
Resource Allocation Messages	These are messages describing a resource allocation command that is given from a Master user to a Swarm user. The message may contain the ID of the resource(s) that is being allocated (if no ID is present the SWARM system will automatically mobilize the best-suited resource(s) to the task), the task that the resource is assigned to perform, and the location where the task is going to be performed.	**EDXL-RM**
Chat Communication Messages	These are messages describing textual information that a Master user wish to distribute to all other Master users. The message contains the ID of the user that sends the message, the textual information (i.e. chat message), and a timestamp.	**EDXL-SitRep**

Table 2: Overview of data types produced by the Master system

All data that is published by the Master system is encoded in the EDXL) format. The data messages that are published in the Master system are also wrapped inside the EDXL-DE element), which is the root XML element or wrapper used by all BRIDGE information management components and client applications (see Figure 16).

```
<?xml version='1.0' encoding='utf-8' standalone='no'?>
<EDXLDistribution xmlns='urn:oasis:names:tc:emergency:EDXL:DE:1.0'>
  <distributionID>e1fcc2f4-d152-4d29-9b96-e8f9ec28dcc5</distributionID>
  <senderID>someID@bridgeproject.eu</senderID>
  <dateTimeSent>2013-10-23T11:25:41+02:00</dateTimeSent>
  <distributionStatus>Exercise</distributionStatus>
  <distributionType>Request</distributionType>
  <combinedConfidentiality>UNCLASSIFIED AND NOT SENSITIVE
  </combinedConfidentiality>
  <language>EN</language>
</EDXLDistribution>
```

Figure 16: EDXL distribution element wrapper

Incident object messages are encoded in EDXL-SitRep, which is a format for communicating situation reports related to the emergency. Figure 17 given an excerpt of an EDXL-SitRep message used to communicate the location of an incident object.

```
<contentDescription>MEXL-SitRep</contentDescription>
<xmlContent>
 <embeddedXMLContent>
  <mEXLSR:SitRep xmlns:mEXLSR='http://icnet.mitre.org/mEXL/SitRep/'>
   <ObservationLocation xmlns=''>
    <gml:Point xmlns:gml='http://www.opengis.net/gml'>
     <gml:pos>58.9198575325583 5.58128458555505</gml:pos>
    </gml:Point>
   </ObservationLocation>
   <MessageID xmlns=''>5495965d-3939-4456-bb79-8fd17e8fb31a</MessageID>
   <IncidentID xmlns=''>Resource meeting place</IncidentID>
   <SituationObservation xmlns=''>
     <ObservationText>This is where we will meet</ObservationText>
   </SituationObservation>
   </mEXLSR:SitRep>
 </embeddedXMLContent>
</xmlContent>
```

Figure 17: EDXL-SitRep message example

Resource allocation messages are encoded in EDXL-RM (Resource Messaging). The messages make use of the RequestResource element in the EDXL-RM standard. Figure 18 gives a short excerpt from using the element RequestResource element to describe a resource allocation.

```
<rmsg:RequestResource xmlns:rm='urn:oasis:names:tc:emergency:EDXL:RM:1.0' x
mlns:xpil='urn:oasis:names:tc:ciq:xpil:3' xmlns:xal='urn:oasis:names:tc:ciq
:xal:3' xmlns:gml='http://www.opengis.net/gml' xmlns:rmsg='urn:oasis:names:
tc:emergency:EDXL:RM:1.0:msg'>
 <rmsg:MessageID>111ccdc7-2a3e-457f-a70f-171f9cf1e0ca</rmsg:MessageID>
 <rmsg:SentDateTime>2013-04-05T10:16:59+02:00</rmsg:SentDateTime>
 <rmsg:MessageContentType>RequestResource</rmsg:MessageContentType>
 <rmsg:OriginatingMessageID>111ccdc7-2a3e-17</rmsg:OriginatingMessageID>
 <rmsg:ContactInformation>
  <rm:ContactRole>Sender</rm:ContactRole>
 </rmsg:ContactInformation>
 <rmsg:ResourceInformation>
   <rmsg:ResourceInfoElementID>7fc73efe-f84</rmsg:ResourceInfoElementID>
  <rmsg:Resource>
   <rm:ResourceID>30ea2645-fb2c-b5b6485f34d8</rm:ResourceID>
  </rmsg:Resource>
  <rmsg:AssignmentInformation>
   <rmsg:AnticipatedFunction>Hold traffic 2</rmsg:AnticipatedFunction>
  </rmsg:AssignmentInformation>
  <rmsg:ScheduleInformation>
   <rmsg:DateTime>2013-04-05T10:16:59+02:00</rmsg:DateTime>
   <rmsg:ScheduleType>RequestedArrival</rmsg:ScheduleType>
   <rmsg:Location>
    <rm:TargetArea>
     <gml:Point>
      <gml:pos>58.9193135368697 5.57863315838111</gml:pos>
     </gml:Point>
    </rm:TargetArea>
   </rmsg:Location>
  </rmsg:ScheduleInformation>
 </rmsg:ResourceInformation>
</rmsg:RequestResource>
```

Figure 18: EDXL-RM Request resource message example

In similarity with incident object messages, chat communication messages are also encoded using the EDXL-SitRep format. In this case however, we make use of a different element for storing the textual information, called ActionPlan. Figure 19 gives an example of using the ActionPlan element to describe a textual chat communication message. The MessageID element provides a unique ID for each message that is sent. The ID of the sender of the message is given from the EDXL-DE wrapper enclosing the SitRep messages.

```
 <mEXLSR:SitRep xmlns:mEXLSR="http://icnet.mitre.org/mEXL/SitRep/">
      <MessageID xmlns="">
          4b1f0f98-3b96-4da5-9e00-a963d1a72f25
      </MessageID>
      <ActionPlan xmlns="">
          Some important message here
      </ActionPlan>
</mEXLSR:SitRep>
```

Figure 19: EDXL-SitRep chat communication message example

After the data has been encoded into the respective EDXL formats, it is published to S2D2S, using specific topics for each data type. As soon as a new message has been published on S2D2S, it will be pushed to all running Master instances that are currently subscribed to the specific topics on S2D2S. The received data will then be decoded, and shown to the user.

3.4 Environment Tagging: eTriage

The BRIDEG eTriage Concept Case makes heavy use of a subset of the EDXL-TEP standard for emergency data exchange. As such, all the eTriage-generated data can be used in EDXL-compatible systems, both within and outside of the BRIDGE constellation.

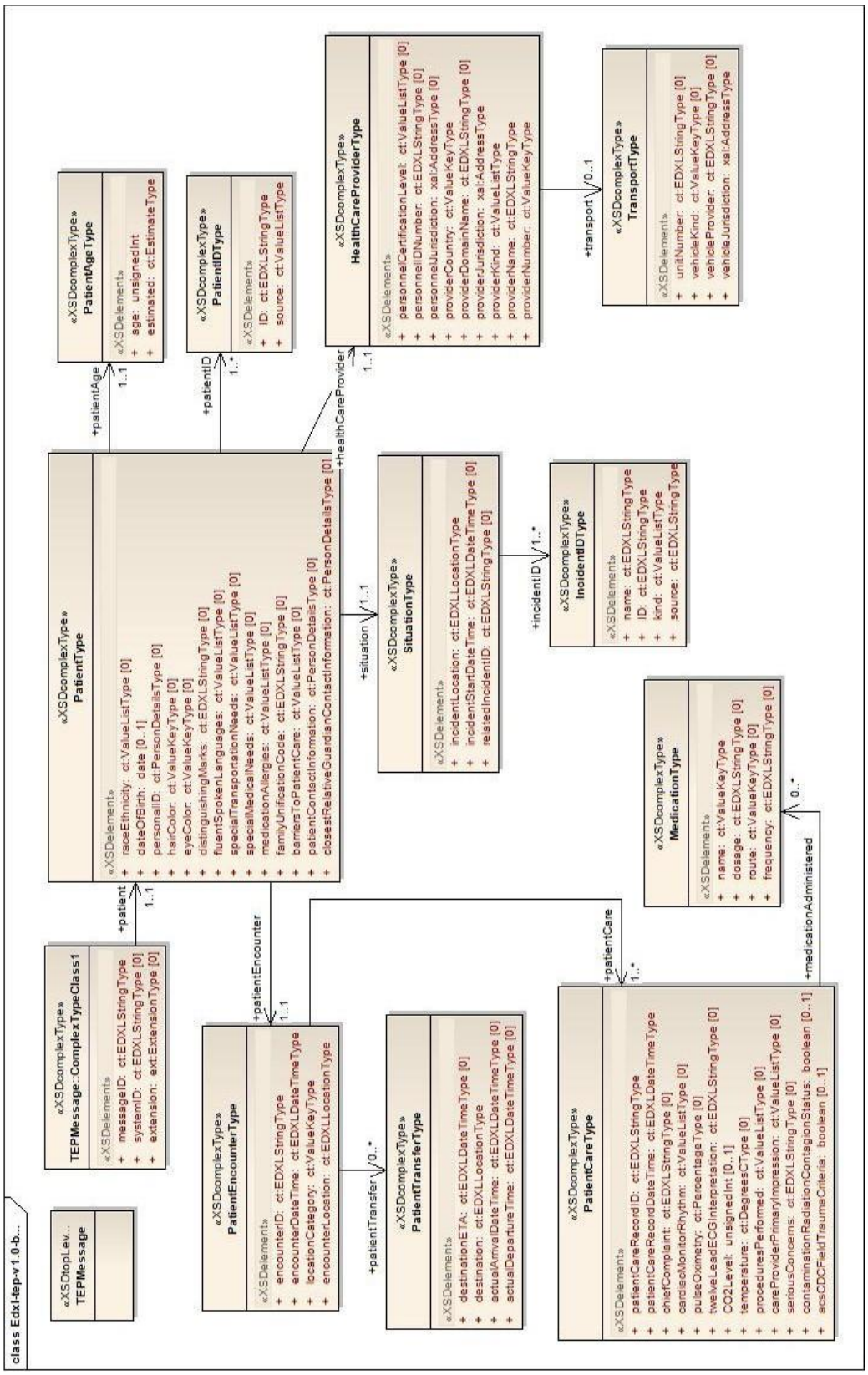


Figure 20 EDXL-TEP fields diagram. Source: OASIS EDXL website, docs.oasis-open.org

In particular:

We set the Provider Info field of the EDXL message to the Bracelet ID, so that the data can be traced back not only to BRIDGE but to the particular bracelet that generated them.

The eTriage system does not know the gender of the patient at the beginning; this can be input into the system during re-triage or in the gathering place. The newer messages will override the older ones with an empty gender field.

The eTriage system allows the patient ID to be independent of the bracelet ID, so that the triage bracelets can be exchanged and the data still traced back to the same patient.

We report the priority as defined in the EDXL standard (Delayed, Immediate, Minor, Deceased). The priority is reported with every message, as is the timestamp. This is to make sure that even if a single message makes it to the command post, we know how old it is and what the state of the patient is. Other sensor values, like position, breathing, and heart rate, are also reported in the EDXL message if available. A new EDXL message is generated each time one of the vital values or GPS position changes.

Given that the bracelets do not have an internal clock, there is a need to synchronize the timestamps. EDXL-TEP does not provide a facility or tag for this function. We therefore rely on the MESH to have a correct clock and on the middleware to do the translation from bracelet timestamp to real-time timestamp.

Even though the eTriage system messages are represented in EDXL, they originate as simple key-value pairs from the bracelets, because the latter have very limited resources. The translation of the key-value pairs to EDXL happens within the MESH network routers and in the middleware, where the resources are considerable. The bracelets do not need these components to function, but if eTriage data is requested in EDXL format, this request can only be fulfilled if the Middleware does the conversion.

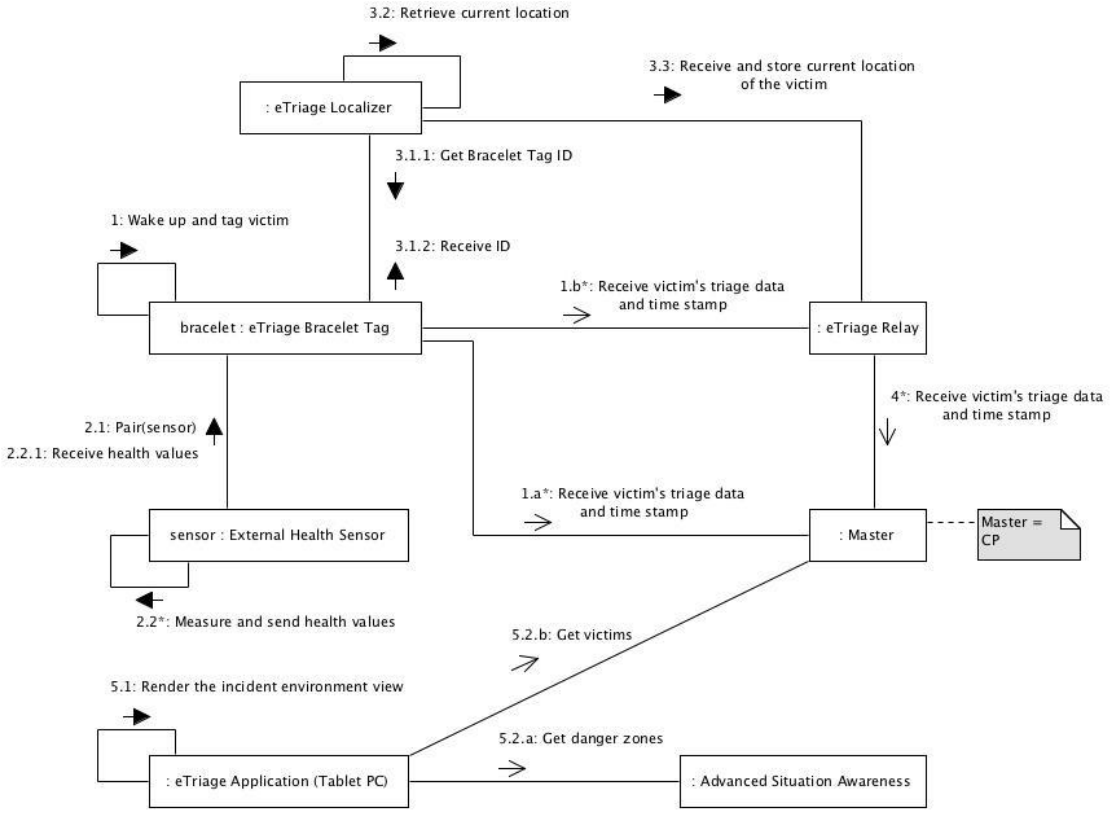


Figure 21 UML Communication Diagram for eTriage

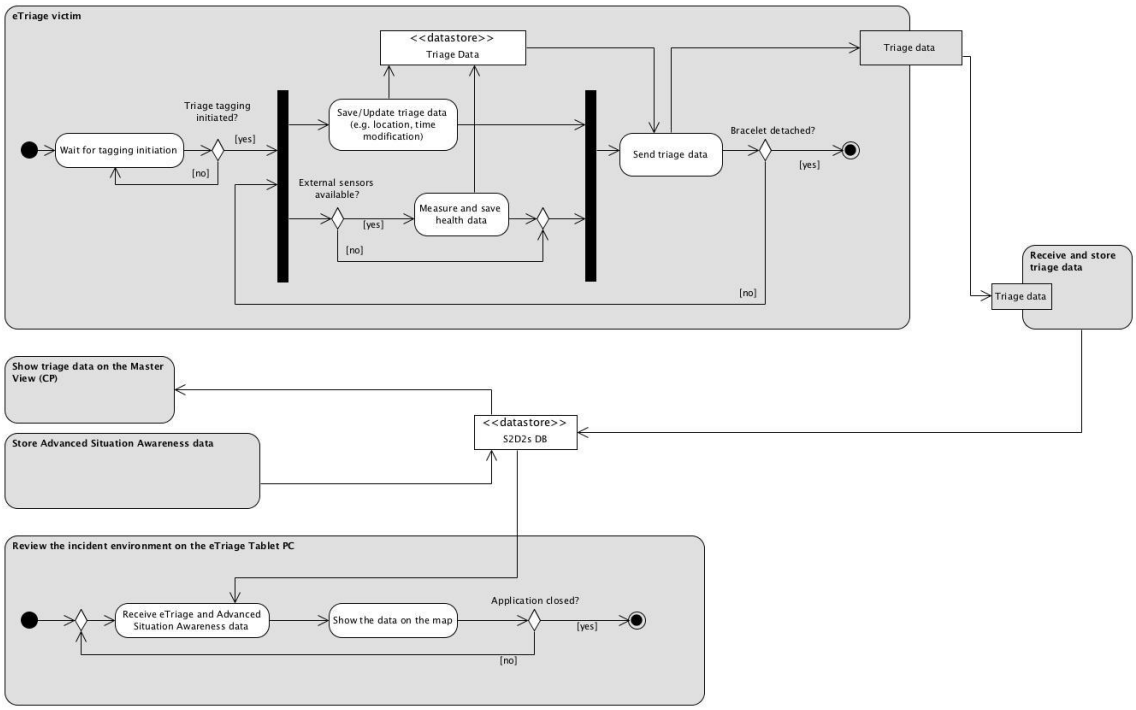


Figure 22 UML Activity Diagram for eTriage

3.5 External sources: Social Media Integration

The BRIDGE middleware also supports the integration media and event streams from (external) social media sources (such Twitter, Flickr). This functionality has been analysed and elaborated within the BRIDGE Concept Case Information Intelligence. The main information model is described below, whereas a more comprehensive description of social and multi media management in BRIDGE with this Concept Case can be found in the accompanying architecture deliverable D04.4.

Selected results of the social media analysis are sent to the master to improve and support situational awareness. Through the social media analysis, sub-events are identified. Sub-events are hotspots (e.g., flooding in specific area etc.) of a crisis where emergency management must be aware of or react on it. These sub-events are identified through the information propagated through social media related to the crisis at hand.

The results of the analysis are shown to a responsible person who decides on the importance of sub-events. Sub-events important for the on-site activities are communicated to the Master. We created a XML-Schema (see Figure 23) to describe the identified (important) sub-events in a list and to transmit them to interested components (e.g., the Master Table). A graphical representation of the schema can be found in Figure 24.

There, a *SubeventList* is composed of one or more *Subevents*. Subevents describe the important hotspots of the analysis. A specific Subevent contains a date-time attribute, when it was available (= identified via the aggregation), coordinates (longitude and latitude information) and keywords describing the content. It consists of a list of *Mediaitems* the Subevent originates from. The Mediaitem element is used to represent simple text (tweets) messages, pictures or videos (e.g., gained from Flickr or YouTube). We decided to use one tag-representation for all kinds of Mediaitems. Therefore, some of the attributes of media items are optional.

The following tags are important for describing the different types of Mediaitems:

- Text:
  - Type = 1 indicates a text message (e.g., a tweet)
  - Description includes the text (e.g., tweet-text).
  - Tags: Additionally include the hashtags (if there is one used)
- Picture:
  - Type = 2 means a picture (e.g., from Flickr)
  - All tags/fields are important - except the user (of the social media platform where the picture originates) does not enter it (e.g., user has not assigned tags to the picture). LocalFile includes a link to the picture/video.
- Video:
  - Type = 3 denotes a video (e.g., from YouTube)
  - All tags equal to the picture case

```
<?xml version="1.0" encoding="UTF-8"?>
<!--<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">-->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:subEvents="www.bridge.eu/subEvents" targetNamespace="www.bridge.eu/subEvents">
   <xs:element name="SubeventList">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="Subevent" minOccurs="1" maxOccurs="unbounded">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="Mediaitem" minOccurs="1" maxOccurs="unbounded">
                        <xs:complexType>
                           <xs:sequence>
                              <xs:element name="Title" type="xs:string"/>
                              <xs:element name="Available" type="xs:dateTime"/>
                              <xs:element name="Tags" type="xs:string"/>
                              <xs:element name="LocalFile" type="xs:string"/>
                              <xs:element name="Type" type="xs:byte"/>
                              <xs:element name="Longitude" type="xs:double"/>
                              <xs:element name="Latitude" type="xs:double"/>
                              <xs:element name="Description" type="xs:string"/>
                              <xs:element name="Author" type="xs:string"/>
                              <xs:element name="URL" type="xs:anyURI"/>
                           </xs:sequence>
                        </xs:complexType>
                     </xs:element>
                  </xs:sequence>
                  <xs:attribute name="available" type="xs:dateTime" use="required"/>
                  <xs:attribute name="longitude" type="xs:double" use="required"/>
                  <xs:attribute name="latitude" type="xs:double" use="required"/>
                  <xs:attribute name="keywords" type="xs:string" use="required"/>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
         <xs:attribute name="source" type="xs:string" use="required"/>
      </xs:complexType>
   </xs:element>
</xs:schema>
```

Figure 23: Schema for communicating social media data

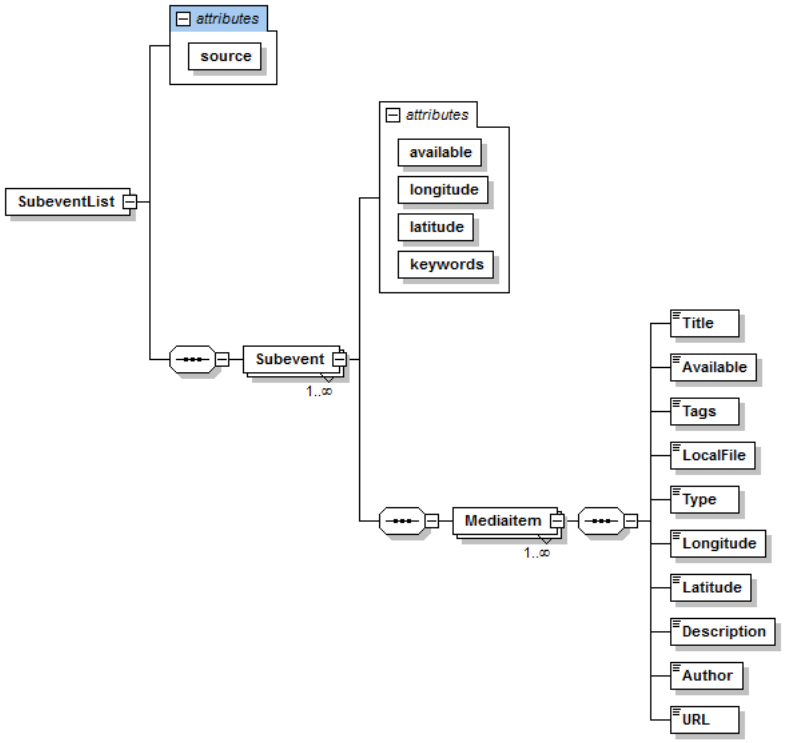


Figure 24: Graphical representation of the XSD Schema

An example for a SubeventList with two Subevents and with different kinds of Mediaitems can be found in Figure 25. The first Subevent contains two simple text Mediaitems. The second Subevent contains two Mediaitems describing a picture and a video, respectively. All Mediaitems include longitude and latitude information.

```
<subEvents:SubeventList xmlns:subEvents="www.bridge.eu/subEvents">
    <Subevent available="2013-07-29T14:27:36.821+02:00" longitude="5.579741954803467"
latitude="58.9215202331543" keywords="fire, injured">
        <Mediaitem>
            <Available>2013-07-29T14:27:36.821+02:00</Available>
            <Tags>fire in harbor</Tags>
            <Type>1</Type>
            <Longitude>5.577575</Longitude>
            <Latitude>58.921331</Latitude>
            <Description>Some fire here #fire #harbor</Description>
        </Mediaitem>
        <Mediaitem>
            <Available>2013-07-29T14:27:36.821+02:00</Available>
            <Tags>fire harbor</Tags>
            <Type>1</Type>
            <Longitude>5.579399</Longitude>
            <Latitude>58.921497</Latitude>
            <Description>Smoke and fire; injured people here #fire #harbor </Description>
        </Mediaitem>
    </Subevent>
    <Subevent available="2013-07-29T14:27:36.821+02:00" longitude="5.581801891326904"
latitude="58.91911697387695" keywords="explosion, fire, harbor">
        <Mediaitem>
            <Title>Fire Explosion</Title>
            <Available>2013-07-29T14:27:36.821+02:00</Available>
            <Tags>fire harbor</Tags>
            <LocalFile>http://www.bridgeproject.eu/notready/pic1.jpg</LocalFile>
            <Type>2</Type>
            <Longitude>5.583605</Longitude>
            <Latitude>58.918318</Latitude>
            <Description>Here is a big fire. pictures shows the harbor in...</Description>
        </Mediaitem>
        <Mediaitem>
            <Title>Fire @ Ferry Harbor</Title>
            <Available>2013-07-29T14:27:36.821+02:00</Available>
            <Tags>fire ferry injured</Tags>
            <LocalFile>http:// www.bridgeproject.eu/notready/v1.mp4</LocalFile>
            <Type>3</Type>
            <Longitude>5.58324</Longitude>
            <Latitude>58.917366</Latitude>
            <Description>the video shows an explosion near harbor ferry help. wounded
here...</Description>
        </Mediaitem>
    </Subevent>
</subEvents:SubeventList>
```

Figure 25: Example of a SubeventList with different Subevents

The EDXL-DE format is used to enclose the event lists. Figure 26 shows the SubeventList within the EDXL message envelope. To gain a better overview, the Subevents in the SubeventList are collapsed (the structure is similar to the example given in Figure 25). The schema allows communicating important sub-events through the S2D2S to BRIDGE clients, the Master Table as well as other external services.

```
<EDXLDistribution xmlns="urn:oasis:names:tc:emergency:EDXL:DE:1.0">
    <distributionID>BRIDGE Important SubEvents</distributionID>
    <senderID>admin@icnet.mitre.org</senderID>
    <dateTimeSent>2013-07-29T14:27:36.837+02:00</dateTimeSent>
    <distributionStatus>Test</distributionStatus>
    <distributionType>Report</distributionType>
    <combinedConfidentiality>Unclass</combinedConfidentiality>
    <language>en-US</language>
    <contentObject>
        <contentDescription>SubEvent List</contentDescription>
        <contentKeyword>
            <valueListUrn>http://icnet.mitre.org/ValueLists/ContentKeywords</valueListUrn>
            <value>SubEvent List</value>
        </contentKeyword>
        <xmlContent>
            <embeddedXMLContent>
                <subEvents:SubeventList xmlns:subEvents="www.bridge.eu/subEvents">
                    + <Subevent available="2013-07-29T14:27:36.821+02:00"
longitude="5.579741954803467" latitude="58.9215202331543" keywords="fire, injured">
                    + <Subevent available="2013-07-29T14:27:36.821+02:00"
longitude="5.581801891326904" latitude="58.91911697387695" keywords=" explosion, fire, harbor">
                </subEvents:SubeventList>
            </embeddedXMLContent>
        </xmlContent>
    </contentObject>
</EDXLDistribution>
```

Figure 26: Embedding the SubeventList into EDXL

3.6 Advanced Situation Awareness: UAV

Unmanned Aerial Vehicles (UAVs) represent one of the many different sources that can be integrated by the BRIDGE platform to help build situation awareness and in the creation of common operational views. This functionality has been developed and demonstrated by the BRIDGE Concept Case Advanced Situation Awareness (ASA) Concept Case and we refer to the accompanying deliverable D04.4 for a description of the UAV/ASA sub-systems, operational requirements and run-time capabilities. In this section we describe the formats used for data from the UAV and how these data are structured as payload content in EDXL messages for distribution over the BRIDGE platform.

There are four types of content emanating from the UAV/ASA, with corresponding EDXL message types,

- UAV status, in general resource deployment status
- Image data, i.e., meta data for static and streamed images
- UAV sensor data results, including measurements and expert advice

3.6.1 UAV status

A UAV is treated as a resource in BRIDGE and its status or characteristics can be reported using EDXL-RM messages. In BRIDGE a *ReportResourceDeploymentStatus* message is used to compile UAV resource status, such as its position.

```
.....
<ResourceInformation>
  <ResourceInfoElementID>1</ResourceInfoElementID>
  <Resource>
   <ResourceID>0ec33f9b-b8f1-44e2-99de-0a8dde04a68b</ResourceID>
   <Name>UAV</Name>
   <TypeStructure>
    <rm:Value>UAV</rm:Value>
    <rm:ValueListURN>urn:x-hazard:vocab:resourceTypes</rm:ValueListURN>
   </TypeStructure>
   <ResourceStatus>
    <DeploymentStatus>
     <rm:Value>Unknown</rm:Value>
     <rm:ValueListURN>urn:x-hazard:vocab:deploymentStatusTypes</rm:ValueListURN>
    </DeploymentStatus>
   </ResourceStatus>
  </Resource>
  <ScheduleInformation>
   <ScheduleType>Current</ScheduleType>
   <Location>
    <rm:TargetArea>
     <gml:Point>
      <gml:pos>18.9827839 69.7031209 120</gml:pos>
     </gml:Point>
    </rm:TargetArea>
   </Location>
  </ScheduleInformation>
 </ResourceInformation>
</ReportResourceDeploymentStatus>
```

Figure 27: UAV resource position information

In this case the position is simply given by a GML[16] point. Additional data of the UAV such as other aspects of its deployment status could also be conveyed in the same message. As in all EDXL messages external (namespace) standard vocabularies are referenced for encoding such properties (omitted in the message above).

3.6.2 UAV image meta data

UAV Image (meta) data is transmitted as embedded XML content in an EDXL-DE message. The image meta data includes image position, a possible observation text message, and, a BRIDGELink (see above) referring to the actual image object or stream

[16]Geography Markup Language, http://www.opengeospatial.org/standards/gml

```
<EDXLDistribution xmlns="urn:oasis:names:tc:emergency:EDXL:DE:1.0">
 <distributionID>3F2504E0-4F89-41D3-9A0C-0305E82C3301</distributionID>
 <senderID>ASA@bridgeproject.eu</senderID>
 <dateTimeSent>2014-06-18T22:23:12.573Z</dateTimeSent>
 <distributionStatus>Exercise</distributionStatus>
 <distributionType>Request</distributionType>
 <combinedConfidentiality>UNCLASSIFIED AND NOT SENSITIVE</combinedConfidentiality>
 <language>EN</language>
 <contentObject>
  <contentDescription>MEXL-UAVStaticImageUpdate</contentDescription>
  <contentKeyword>
   <valueListUrn>http://icnet.mitre.org/ValueLists/ContentKeywords</valueListUrn>
   <value>MEXL-UAVStaticImageUpdate</value>
  </contentKeyword>
  <xmlContent>
   <embeddedXMLContent>
    <UAVStaticImageUpdate xmlns="urn:BRIDGE:ASA">
     <ImagePosition xmlns="">
      <gml:Point xmlns:gml="http://www.opengis.net/gml">
       <gml:pos>18.9827839  69.7031209  120</gml:pos>
      </gml:Point>
     </ImagePosition>
     <SituationObservation xmlns="">
      <ObservationText>UAV1:Nice image of hexacopter, optional start with UAV id followed by :</ObservationText>
      <TimeStamp>2014-05-22T23:02:01.122Z</TimeStamp>
      <bridge:Links xmlns:bridge="urn:bridge:link">
       <bridge:Link description="UAV Image">
        <bridge:Url>http://127.0.0.1:8082/GRANDTunneling/0/0//StaticImage/image2.jpg?Description=Asa:Webserver</bridge:Url>
        <bridge:Size>58937392</bridge:Size>
        <bridge:Mimetype>image/png</bridge:Mimetype>
       </bridge:Link>
      </bridge:Links>
     </SituationObservation>
    </UAVStaticImageUpdate>
   </embeddedXMLContent>
  </xmlContent>
 </contentObject>
</EDXLDistribution>
```

Figure 28: Image meta data DE message

A similar structure is used for streaming images but with the content element *UAVStreamUpdate* as parent to a BRIDGELink with a URI to a streaming server.

```
.....
<xmlContent>
    <embeddedXMLContent>
     <UAVStreamUpdate xmlns="urn:BRIDGE:ASA">
      <SituationObservation xmlns="">
       <ObservationText>Computer Webcam</ObservationText>
       <TimeStamp>2014-05-22T23:02:01.122Z</TimeStamp>
       <bridge:Links xmlns:bridge="urn:bridge:link">
        <bridge:Link description="UAV Stream">
         <bridge:Url>rstp://124.22.34.88:8554/visual</bridge:Url>
         <bridge:Mimetype>application/x-rtsp</bridge:Mimetype>
        </bridge:Link>
       </bridge:Links>
      </SituationObservation>
     </UAVStreamUpdate>
    </embeddedXMLContent>
   </xmlContent>
  </contentObject>
</EDXLDistribution>
```

Figure 29: Stream meta data element

3.6.3 UAV Sensor data results

A central function of the UAV/ASA Concept Case is to process and model UAV environmental sensor inputs (in addition to image, e.g., temperature, gas concentration) for the purpose of feeding situation data to the BRIDGE platform and its clients. In addition to the sensor measurements this includes the provision of expert advice on the consequences of an incident, as well as models of physical phenomena such as toxic plume dispersion and explosions (see D04.4 for details).

Advice message content is based on a vocabulary containing terms for characterizing hazards by severity, response actions and safety for first responders. The vocabulary has a corresponding XML schema defining the structure of the specific advice messages. A graphical subset of the schema is depicted below (for the full XML serialization see appendix)

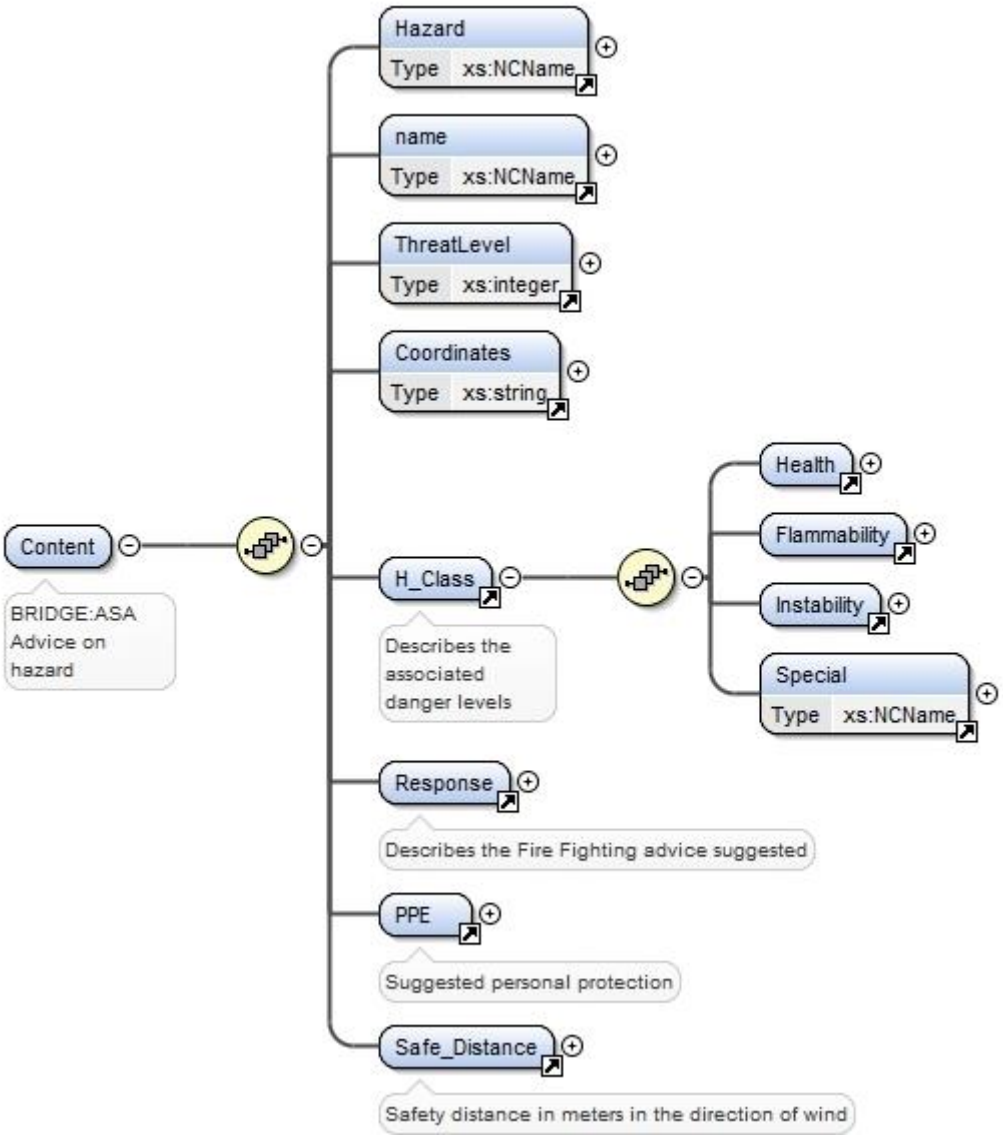


Figure 30: BRIDGE-ASA XML vocabulary for hazards advice

The advice XML is sent embedded in EDXL-DE messages. An example of an advice message for a chemical incident is shown below (the DE element omitted).

```
<xmlContent>
  <embeddedXMLContent>
   <AdviseUpdate xmlns="urn:BRIDGE:ASA">
    <ObservationText>Advise on chemical Acrolein</ObservationText>
    <TimeStamp>2014-05-22T23:02:01.122Z</TimeStamp>
    <Position>
     <gml:Point xmlns:gml="http://www.opengis.net/gml">
      <gml:pos>18.9827839 69.7031209 120</gml:pos>
     </gml:Point>
    </Position>
    <Content>
     <ns:Hazard xmlns:ns="urn:BRIDGE:ASA">Chemical</ns:Hazard>
     <ns:name xmlns:ns="urn:BRIDGE:ASA">Acrolein</ns:name>
     <ns:ThreatLevel xmlns:ns="urn:BRIDGE:ASA">3</ns:ThreatLevel>
     <ns:Coordinates xmlns:ns="urn:BRIDGE:ASA">(18.9827839, 69.7031209, 120)</ns:Coordinates>
     <ns:H_Class xmlns:ns="urn:BRIDGE:ASA">
      <ns:Health>
       <ns:level>4</ns:level>
       <ns:comment>Can be lethal.</ns:comment>
      </ns:Health>
      <ns:Flammability>
       <ns:level>3</ns:level>
       <ns:comment>Can be ignited under almost all ambient temperature
        conditions.</ns:comment>
      </ns:Flammability>
     </ns:H_Class>
     <ns:Response xmlns:ns="urn:BRIDGE:ASA">
      <ns:Agents> Alcohol foam, dry chemical, carbon dioxide </ns:Agents>
      <ns:Precaution>
       <ns:ol>
        <ns:li>Unsafe levels of Acrolein detected take extreme care and wear appropriate
         personal protection before entering the region</ns:li>
        <ns:li>Withdraw immediately in case of rising sound from venting safety device or
         any discolouration of tank due to fire.</ns:li>
       </ns:ol>
      </ns:Precaution>
      <ns:Evacuation_radius>10 km</ns:Evacuation_radius>
     </ns:Response>
     <ns:PPE xmlns:ns="urn:BRIDGE:ASA">
      <ns:Body>
       <ns:ol>
        <ns:li> Fully encapsulating, vapor protective clothing should be worn to deal with
         spills or leaks with no fire. </ns:li>
        <ns:li> Fire Fighters should wear additional protection for heat. </ns:li>
       </ns:ol>
      </ns:Body>
      <ns:Respirator>
       <ns:ol>
        <ns:li> Any supplied-air respirator that has a full facepiece and is operated in a
         pressure-demand or other positive-pressure mode. </ns:li>
        <ns:li> Carry an auxiliary self-contained positive-pressure breathing apparatus.
        </ns:li>
       </ns:ol>
      </ns:Respirator>
     </ns:PPE>
     <ns:Safe_Distance xmlns:ns="urn:BRIDGE:ASA">
      <ns:Isolate>800</ns:Isolate>
      <ns:Protect>9300, 11000</ns:Protect>
     </ns:Safe_Distance>
    </Content>
   </AdviseUpdate>
  </embeddedXMLContent>
</xmlContent>
```

Figure 31: BRIDGE-ASA Chemical hazard advice message

4 Demonstration and Simulation Platform

The demonstration and simulation platform is designed to be used as a test bench for the middleware as well as for the various BRIDGE Concept Cases.

4.1 Simulation platform

A set of simulation applications has been developed in order to support validation of the BRIDGE middleware as well as existing and future Concept Cases. The simulation apps are used to feed the platform and the Concept Case with incident and other context data. The simulation platform is deployed in a laboratory setting.

Another important usage of the simulation platform is to be able to test components with regards to stability and scalability in a controlled environment.

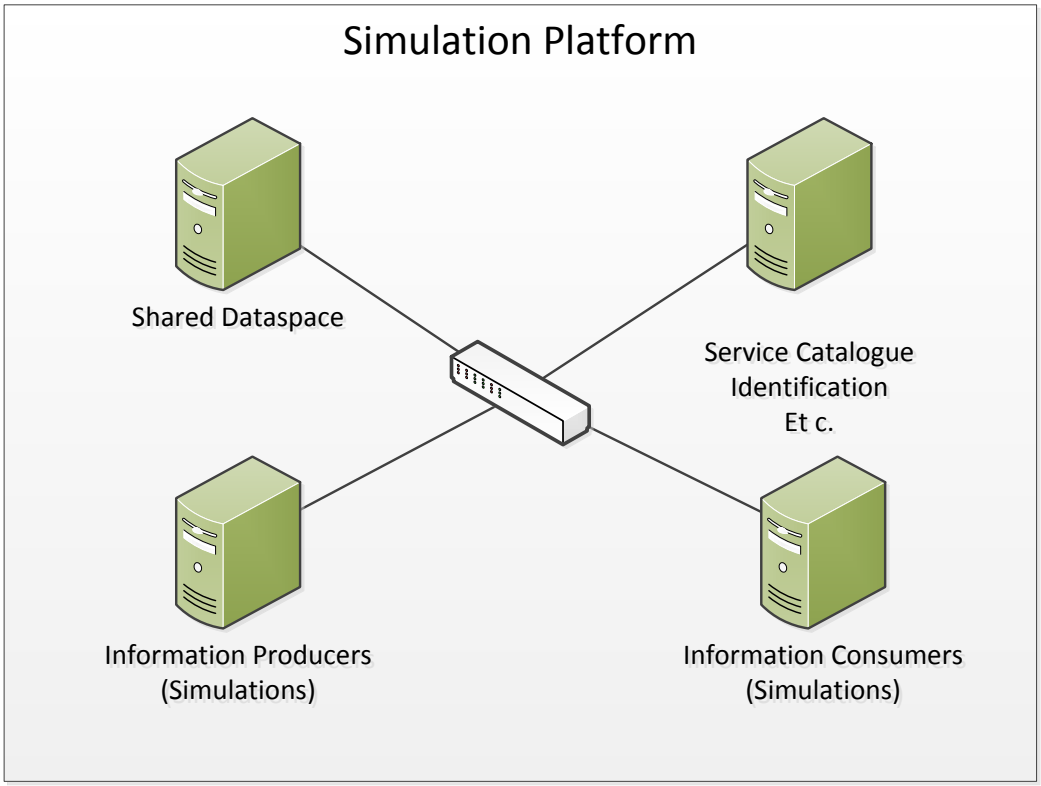


Figure 32: Simulation Platform Deployment

Figure 32 shows the configuration of the simulation platform setup used for testing the stability and scalability of components. The setup uses a closed network consisting of four computers:

1. Shared Dataspace: This computer hosts the storage and messaging parts of the BRIDGE middleware.
2. Service Catalogue, Identification etc.: This computer hosts the other components of the BRIDGE middleware
3. Information Producers: This computer hosts the information producing simulations.
4. Information Consumers: This computer hosts simulations that consume information.

One can note that the MESH network is not part of the simulation platform. However, components on the platform can access generated MESH data acquired from tests with the MESH platform mimicking the bandwidth and messages sizes. The reason for this is that having all the hardware available for testing is not possible.

The special components developed to support the Simulation Platform are listed below in the table:

Simulation Component	Description
MESH	Feeding the middleware with simulated data from the MESH at the rate that is the maximum for the actual MESH
Help Beacons	Creates simulated beacons that are posted to S2D2S (the shared dataspace).
SWARM	Creates simulated resources with location and other information.
Advanced Situational Awareness	Provides simulation of data feeds from the Concept Case Advanced Situational Awareness. This includes images and video streams.
Information Intelligence	Twitter feed simulations published to the shared dataspace
Master	Provide simulation application for sending incident object messages, resource allocation messages, and chat messages to S2D2S.
eTriage	Provides simulated triage data.
Consumers pub/Sub	Receive data from the middleware using subscriptions using the publish/subscribe mechanism, pure simulation used for measuring the throughput in the system.
Consumers Query	Queries Data from the shared dataspace, measuring performance.
Service Catalogue	Register and query services in the service catalogue, measuring performance and scalability.
Identification service	Create/Search/Remove identities, measuring performance and scalability.

Table 3: BRIDGE simulators

4.2 Demonstration configurations

The BRIDGE platform has been demonstrated in several different configurations, in lab setups as well as in emergency exercise settings. At the Risavika exercise[17] the BRIDGE platform was deployed with all main middleware components, network infrastructure and Concept Cases as part of a major incident exercise.

[17] http://www.bridgeproject.eu/en/news/BRIDGE-Conducts-Its-Third-Successful-Demonstration-in-Stavanger--Norway_126

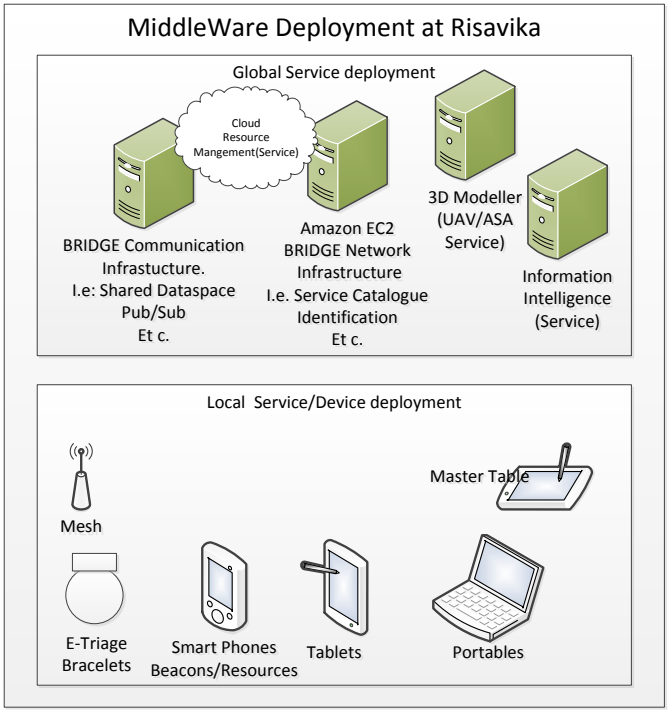


Figure 33: Service and device deployment at Risavika

The configuration involved locally deployed network infrastructure (Mesh), and first responder devices (triage tags and smart phones). The service provision involved both locally and cloud based services. The WISE training subsystem was also part of the overall setup providing tools for exercise data capture and post exercise analysis.

Another deployment was done during the Alpine Validation where the Advanced Situational Awareness (ASA) and help beacon Concept Cases were demonstrated. From a middleware point of view the ASA Concept Case including the usage of the sensor equipped hexacopter (UAV) is the most interesting since it provides video streams and images that are published on the BRIDGE network. The deployment is described below in Figure 33.

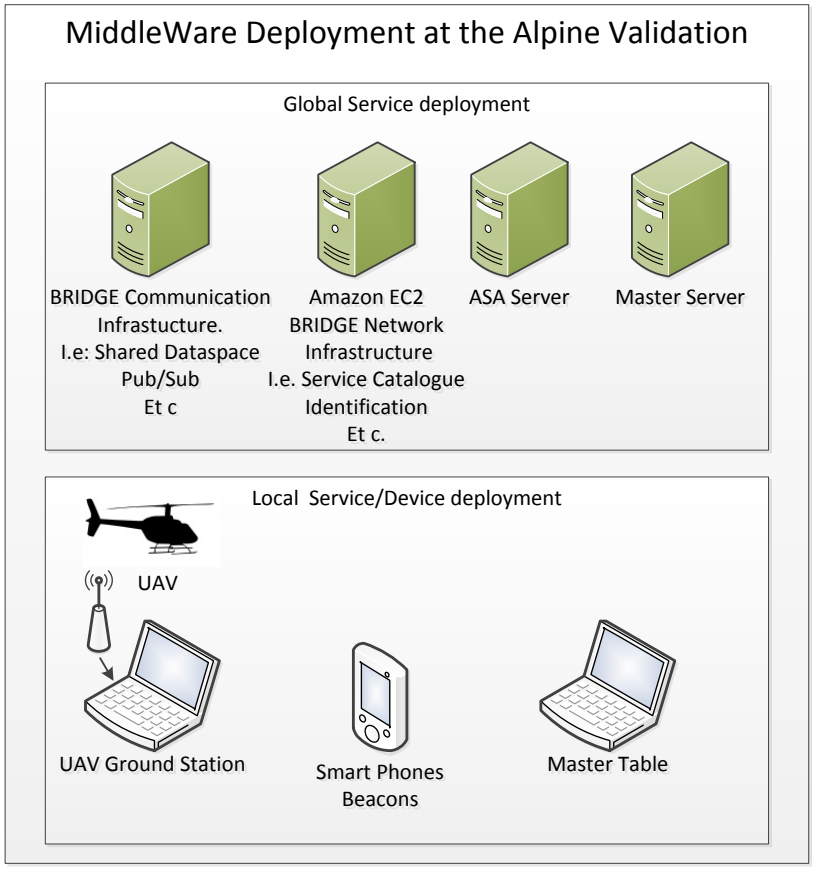


Figure 34: Middleware deployment at the Alpine Validation

The integration of the BRIDGE ASA provided data and streams at the BRIDGE ASA Server was done using the On-Site storage mechanism together with transformation in-between ASA internal formats and the formats used in the BRIDGE network. An overview of the integration is shown in Figure 34.

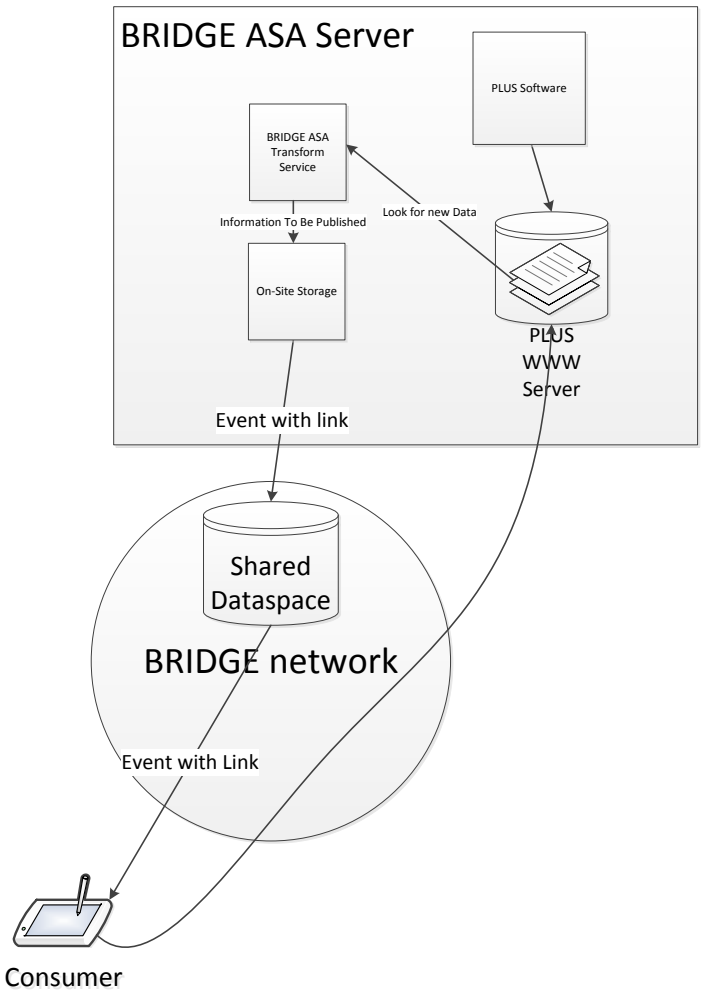


Figure 35: BRIDGE ASA Concept Case Server integration

The BRIDGE ASA uses a web server to publish the information that it creates, for instance the advice data. The integration with the BRIDGE network was done using the following steps:

1. The BRIDGE ASA publishes new information
2. The BRIDGE ASA Transform service monitors the web server file system to see any newly created file.
3. If a new file is found the BRIDGE ASA Transform service will determine the type of information published and transform it into a BRIDGE compliant format.
4. The BRIDGE ASA Transform service informs the BRIDGE On-Site Storage Service that there is new information available.
5. The BRIDGE On-Site Storage Service creates an event with links following the BRIDGE Link Format, see 3.2 for a complete description of the format.
6. The Event is published to the shared data space
7. Finally the consumer e.g., the MASTER receives the event and can retrieve the information from the ASA using the provided links. Note that the link uses BRIDGE Network addressing so that the data can be retrieved even if it is stored locally.

5 Summary and conclusion

This report addresses facets of data representation and sub system deployment of the BRIDGE platform, in the context of the Information and Deployment View of the BRIDGE architecture description.

We speak of BRIDGE as a system of systems in the sense that BRIDGE Concept Cases and middleware services should be assembled (or orchestrated) on demand and in context of incident. Hence there is no beforehand determined configuration, although there is a BRIDGE core comprising a middleware instance (LinkSmart, S2D2S), a service catalogue, a Master instance and one or more of the current Concept Cases.

BRIDGE does not provide or prescribe a global information model, but recommends the use of existing interoperability standards, notably the EDXL. The middleware is SOA-based and event driven, and can run on current network infrastructures. To this end the Concept Cases are model implementations of and serve as guidelines for a set of basic EMS functions (visualization, resource management, triaging etc.).

We conclude that the adoption of the EDXL (de facto) message standard has contributed to syntactic and (to some extent) semantic interoperability on the Concept Case and service levels in BRIDGE. However, its applicability for inter-organisational cross-border emergency systems interoperability remains to be further validated.

The architecture descriptions of BRIDGE comprise the following deliverable documentation

- State of the art (D04.1[18])
- Functional View and SOA (D04.2[16])
- Information and Deployment View (D04.3[16])
- Multimedia management (D04.4)

However, the architecture deliverables do not describe the specific innovations and technical realizations of the individual BRIDGE sub systems (Concept Cases, SOA middleware, network infrastructures), these are to be found in the respective technical Work Packages and deliverables.

[18] Publicly available at http://www.bridgeproject.eu/en/bridge-results/deliverables

Appendix 1: BRIDGE ASA Advice Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="Content">
    <xs:annotation>
      <xs:documentation>BRIDGE:ASA Advice on hazard</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Hazard"/>
        <xs:element ref="name"/>
        <xs:element ref="ThreatLevel"/>
        <xs:element ref="Coordinates"/>
        <xs:element ref="H_Class">
          <xs:annotation>
            <xs:documentation>Describes the associated danger levels</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element ref="Response">
          <xs:annotation>
            <xs:documentation>Describes the Fire Fighting advice suggested</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element ref="PPE">
          <xs:annotation>
            <xs:documentation>Suggested personal protection</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element ref="Safe_Distance">
          <xs:annotation>
            <xs:documentation>Safety distance in meters in the direction of wind</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Hazard" type="xs:NCName"/>
  <xs:element name="name" type="xs:NCName"/>
  <xs:element name="ThreatLevel" type="xs:integer"/>
  <xs:element name="Coordinates" type="xs:string"/>
  <xs:element name="H_Class">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Health"/>
        <xs:element ref="Flammability"/>
        <xs:element ref="Instability"/>
        <xs:element ref="Special"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Health">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="level"/>
        <xs:element ref="comment"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Flammability">
    <xs:annotation> </xs:annotation>
    <xs:complexType>
      <xs:sequence>
```

```
        <xs:element ref="level"/>
        <xs:element ref="comment"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Instability">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="level"/>
        <xs:element ref="comment"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Special" type="xs:NCName"/>
  <xs:element name="Response">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Agents">
          <xs:annotation>
            <xs:documentation>
              The following section describes the preferable Fire Fighting Agent if other than water
            </xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element ref="Precaution">
          <xs:annotation>
            <xs:documentation>Precautions suggested</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element ref="Evacuation_radius">
          <xs:annotation>
            <xs:documentation>
              If evacuation is suggested the following section describes evacuation radius
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Agents" type="xs:string"/>
  <xs:element name="Precaution" type="ol"/>
  <xs:element name="Evacuation_radius" type="xs:string"/>
  <xs:element name="PPE">
    <xs:annotation/>
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="category"/>
        <xs:element ref="APF">
          <xs:annotation>
            <xs:documentation>Assigned protection factor suggested</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element ref="Body">
          <xs:annotation>
            <xs:documentation>PPE description for covering the body</xs:documentation>
          </xs:annotation>
        </xs:element>
        <xs:element ref="Respirator">
          <xs:annotation>
            <xs:documentation>
              Respirators suggested, e.g., type of supplied-air respirator
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
```

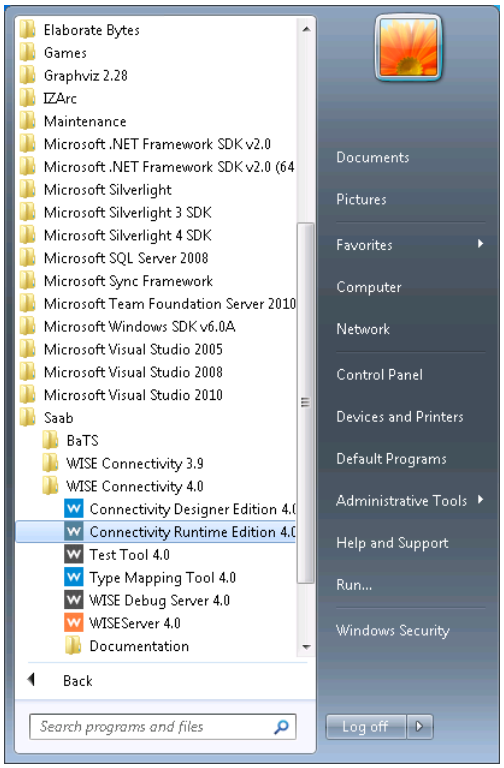
```
    </xs:complexType>
  </xs:element>
  <xs:element name="category" type="xs:string"/>
  <xs:element name="APF" type="xs:integer"/>
  <xs:element name="Body" type="ol"/>
  <xs:element name="Respirator" type="ol"/>
  <xs:element name="Safe_Distance">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Isolate"/>
        <xs:element ref="Protect"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Isolate" type="xs:integer"/>
  <xs:element name="Protect" type="xs:string"/>
  <xs:element name="level" type="xs:integer"/>
  <xs:element name="comment" type="xs:string"/>
  <xs:complexType name="ol">
    <xs:sequence>
      <xs:element ref="ol"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="ol">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="li"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="li" type="xs:string"/>
</xs:schema>
```

Appendix 2: WISE Connectivity run time

Connectivity Runtime Edition

Starting CoRE

CoRE is started using the shortcut found under the Saab – WISE Connectivity X.Y folder of the Start menu (X.Y indicates the product version number);

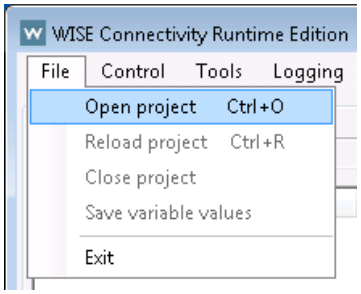


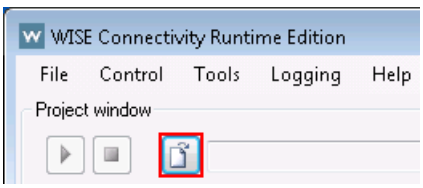
Execution control

In order for a connectivity project to be executed, CoRE first loads and validates the project files before starting the execution. The following chapters describe how this is done.

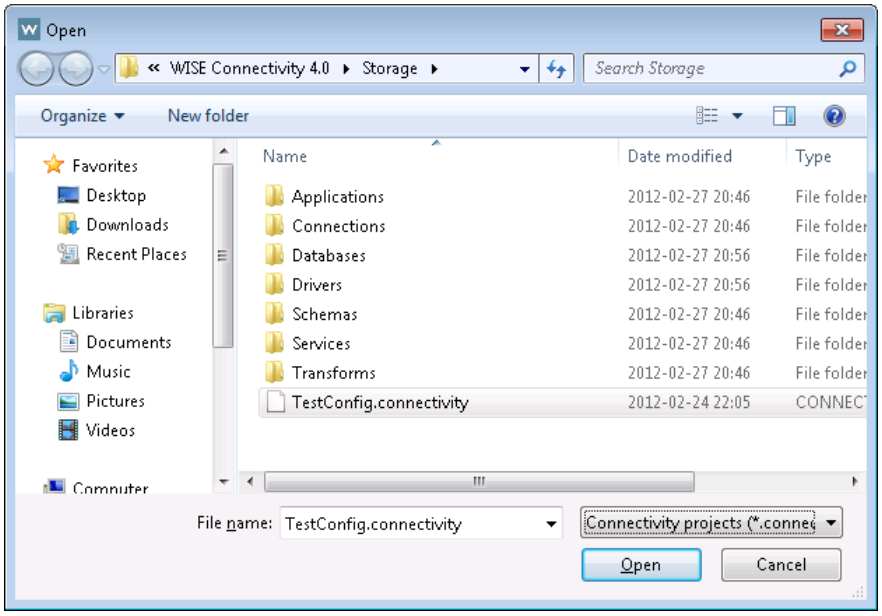
Opening a connectivity project

To open a connectivity project, select "File" – "Open project" from the menu or click the "Open project" button;





A file selection dialog is now displayed;



Select the WISE Connectivity project file you wish to load and press “Open”.

CoRE will proceed to load and validate the selected connectivity project.

If validation generates any error or warnings, the user is notified in the “Log Window”.

If validation is successful, CoRE enables the “Start” command to indicate that the connectivity project is ready for execution.

Ready mode

“Ready-mode” is used in redundancy scenarios where a two CoRE instances are started and where one of them is running as standby.

The purpose of “ready-mode” is that the drivers are instructed to disable their input or output dataflow with the goal to keep the standby CoRE updated with the latest changes while output to target systems are disabled to avoid duplicate entities. Should the primary CoRE fail, the standby CoRE instance can resume by exiting “ready-mode”.

“Ready-mode” can be exited either manually by unselecting the “Control” – “Toggle ready mode” menu item or by external software using the administration socket interface.